PTO/SB/81 (01-09)
Approved for use through 11/30/2011. OMB 0651-0035
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

POWER OF ATTORNEY OR REVOCATION OF POWER OF ATTORNEY WITH A NEW POWER OF ATTORNEY AND CHANGE OF CORRESPONDENCE ADDRESS

Application Number	
Filing Date	
First Named Inventor	Craig S. Etchegoyen
Title	Remote Update of Computers Based on Physica
Art Unit	
Examiner Name	
Attorney Docket Number	UNLP-P-10-037

I hereby revoke all previous powers of attorney given in the above-identified application.

☐ A Power of Attorney is submitted herewith.

OR

☒ I hereby appoint Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith:

96051

OR

☐ I hereby appoint Practitioner(s) named below as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith:

Practitioner(s) Name	Registration Number

Please recognize or change the correspondence address for the above-identified application to:

☒ The address associated with the above-mentioned Customer Number.

OR

☐ The address associated with Customer Number:

OR

☐	Firm or Individual Name				
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the:

☒ Applicant/Inventor.

OR

☐ Assignee of record of the entire interest. See 37 CFR 3.71.
Statement under 37 CFR 3.73(b) (Form PTO/SB/96) submitted herewith or filed on _____

SIGNATURE of Applicant or Assignee of Record

Signature		Date	June 16, 2010
Name	Craig S. Etchegoyen	Telephone	949-788-1740
Title and Company	Founder, Uniloc USA, Inc.		

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required. Submit multiple forms if more than one signature is required, see below*.

☐ *Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 3 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Doc Code: Oath
Document Description: Oath or declaration filed

PTO/SB/01 (04-05)
Approved for use through 07/31/2006. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION (37 CFR 1.63)

☒ Declaration Submitted With Initial Filing OR ☐ Declaration Submitted After Initial Filing (surcharge (37 CFR 1.16(f)) required)

Attorney Docket Number	UN-NP-AD-007
First Named Inventor	Craig S. Etchegoyen
COMPLETE IF KNOWN	
Application Number	
Filing Date	
Art Unit	
Examiner Name	

I hereby declare that: (1) Each inventor's residence, mailing address, and citizenship are as stated below next to their name; and (2) I believe the inventor(s) named below to be the original and first inventor(s) of the subject matter which is claimed and for which a patent is sought on the invention titled:

Remote Update of Computers Based on Physical Device Recognition

(Title of the Invention)

the application of which

☒ is attached hereto

OR

☐ was filed on (MM/DD/YYYY) _____ as United States Application Number or PCT International Application Number _____ and was amended on (MM/DD/YYYY) _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified application, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56, including for continuation-in-part applications, material information which became available between the filing date of the prior application and the national or PCT international filing date of the continuation-in-part application.

Authorization To Permit Access To Application by Participating Offices

☒ If checked, the undersigned hereby grants the USPTO authority to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the World Intellectual Property Office (WIPO), and any other intellectual property offices in which a foreign application claiming priority to the above-identified patent application is filed access to the above-identified patent application. See 37 CFR 1.14(c) and (h). This box should not be checked if the applicant does not wish the EPO, JPO, KIPO, WIPO, or other intellectual property office in which a foreign application claiming priority to the above-identified patent application is filed to have access to the above-identified patent application.

In accordance with 37 CFR 1.14(h)(3), access will be provided to a copy of the above-identified patent application with respect to: 1) the above-identified patent application-as-filed; 2) any foreign application to which the above-identified patent application claims priority under 35 U.S.C. 119(a)-(d) if a copy of the foreign application that satisfies the certified copy requirement of 37 CFR 1.55 has been filed in the above-identified patent application; and 3) any U.S. application-as-filed from which benefit is sought in the above-identified patent application.

In accordance with 37 CFR 1.14(c), access may be provided to information concerning the date of filing the Authorization to Permit Access to Application by Participating Offices.

This collection of information is required by 35 U.S.C. 115 and 37 CFR 1.63. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 47 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

PTO/SB/01 (04-05)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

DECLARATION — Utility or Design Patent Application

Claim of Foreign Priority Benefits

I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or (f), or 365(b) of any foreign application(s) for patent, inventor's or plant breeder's rights certificate(s), or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent, inventor's or plant breeder's rights certificate(s), or any PCT international application having a filing date before that of the application on which priority is claimed.

Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached? YES	NO
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐
			☐	☐	☐

☐ Additional foreign application number(s) are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.

PTO/SB/01 (04-09)
Approved for use through 04/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number

DECLARATION — Utility or Design Patent Application

Direct all correspondence to:	☒	The address associated with Customer Number:	95051	OR ☐	Correspondence address below

Name		
Address		
City	State	Zip
Country	Telephone	Email

WARNING:

Petitioner/applicant is cautioned to avoid submitting personal information in documents filed in a patent application that may contribute to identity theft. Personal information such as social security numbers, bank account numbers, or credit card numbers (other than a check or credit card authorization form PTO-2038 submitted for payment purposes) is never required by the USPTO to support a petition or an application. If this type of personal information is included in documents submitted to the USPTO, petitioners/applicants should consider redacting such personal information from the documents before submitting them to the USPTO. Petitioner/applicant is advised that the record of a patent application is available to the public after publication of the application (unless a non-publication request in compliance with 37 CFR 1.213(a) is made in the application) or issuance of a patent. Furthermore, the record from an abandoned application may also be available to the public if the application is referenced in a published application or an issued patent (see 37 CFR 1.14). Checks and credit card authorization forms PTO-2038 submitted for payment purposes are not retained in the application file and therefore are not publicly available. Petitioner/applicant is advised that documents which form the record of a patent application (such as the PTO/SB/01) are placed into the Privacy Act system of records DEPARTMENT OF COMMERCE, COMMERCE-PAT-2, System name: *Patent Application Files*. Documents not retained in an application file (such as the PTO-2038) are placed into the Privacy Act system of COMMERCE/PAT-TM-10, System name: *Deposit Accounts and Electronic Funds Transfer Profiles*.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

NAME OF SOLE OR FIRST INVENTOR:		☐ A petition has been filed for this unsigned inventor
Given Name (first and middle [if any])	Family Name or Surname	
Craig Stephen	Etchegoyen	
Inventor's Signature	Date	

Residence: City	State	Country	Citizenship
Irvine	CA	USA	USA

Mailing Address			
2161 Michelson Drive Suite 100			
City	State	Zip	Country
Irvine	CA	92612	USA

☐ Additional inventors or a legal representative are being named on the ____ supplemental sheet(s) PTO/SB/02A or 02LR attached hereto.

REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

[0001] This application claims priority to U.S. Provisional Application No. 61/220,092 which was filed June 24, 2009 and which is fully incorporated herein by reference.

BACKGROUND

Field of the Invention

[0002] The present invention relates to computers and, in particular, to methods, apparatus and systems for maintaining appropriate configuration updates to software/hardware configuration through the use of physical device recognition to tailor configuration updates.

Description of the Related Art

[0003] Monitoring changes and updates to the plurality of computer programs resident on a client device is a difficult task for the typical user to consistently perform. In addition, the latest update from a vendor may not be appropriate considering the hardware, software or physical/geo-location of the client device. Thus, there is a need in the art for a tool that will automate the program configuration update process and optimize the suggested updated program configuration to match the environment of the client device.

[0004] The present invention is directed toward a system, method and apparatus for remote updating of the configuration of a computer. One embodiment of the invention is system for remote updating a computer configuration, comprising: a client device configured to load a computer program to perform a remote update; a processor, at the client device, configured to perform physical device recognition on the client device to determine machine parameters, wherein unique device identifiers are generated for the client device, at least in part, based on the determined machine parameters; a transceiver configured to send the unique device identifiers to at least one of an auditing server and an update server via Internet; an update server configured to collect the unique device identifiers from at least one client device; a processor, at the update server, configured to analyzed the unique identifiers at the update server, wherein the analyzed unique identifiers determine an updated program configuration; and a transceiver, at the update server, configured to deliver the updated program configuration to the client device via Internet.

APPLE EXHIBIT 1102, Page 5 of 1048

[0005] In accordance with one aspect of the embodiments described herein, there is provided an apparatus for remote update of a program, comprising: means for loading a client device with a computer program configured to perform a remote update; means for performing physical device recognition on the client device to determine machine parameters; means for generating unique device identifier based at least in part on the determined machine parameters; means for sending the unique device identifier to at least one of an auditing server and an update server; and means for receiving an updated program configuration from the update server.

[0006] In accordance with another aspect of the embodiments described herein, there is provided a method for remote update of a program, comprising: collecting unique identifiers from at least one of an audit server and client device at an update server; analyzing the unique identifiers; determining an updated program configuration for the client device from the analyzed unique identifiers; and delivering the updated program configuration to the client.

[0007] In accordance with another aspect of the embodiments described herein, there is provided a tangible computer readable medium having stored thereon, computer-executable instructions that, if executed by a computing device, cause the computing device to perform a method comprising: loading a client device with a computer program configured to perform a remote update; performing physical device recognition on the client device to determine machine parameters; generating unique device identifier based at least in part on the determined machine parameters; sending the unique device identifier to at least one of an auditing server and an update server; and receiving an updated program configuration from the update server.

[0008] In accordance with another aspect of the embodiments described herein, there is provided an apparatus for remote updating of a program, comprising: means for collecting unique identifiers from at least one of an audit server and client device; means for analyzing the unique identifiers; means for determining an updated program configuration for the client device from the analyzed unique identifiers; and means for delivering the updated program configuration to the client.

[0009] In accordance with another aspect of the embodiments described herein, there is provided a tangible computer readable medium having stored thereon, computer-executable instructions that, if executed by a computing device, cause the computing device to perform a method comprising: collecting unique identifiers from at least one of an audit server and client

APPLE EXHIBIT 1102, Page 6 of 1048

device; analyzing the unique identifiers; determining an updated program configuration for the client device from the analyzed unique identifiers; and delivering the updated program configuration to the client.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. 1 is a schematic diagram of a system for remote updating of a client device by an update server in accordance with an embodiment of the invention.

[0011] FIG. 2 is a flow diagram of a method for remote updating of a client device in accordance with an embodiment of the invention implemented on the client device.

[0012] FIG. 3 is a block diagram of an apparatus according to the invention that may be configured as a client device, or as a processor or similar device for use within a client device.

[0013] FIG. 4 is a flow diagram of a method for remote updating of a client device in accordance with an embodiment of the invention implemented on the update server.

[0014] FIG. 5 is a block diagram of an apparatus according to the invention that may be configured as an update server, or as a processor or similar device for use within an update server.

[0015] FIG. 6 is a block diagram of memory allocation for a unique device identifier used in the various exemplary embodiments of the invention.

DETAILED DESCRIPTION

[0016] In accordance with the present technology, there is provided a system, method and apparatus for the remote update of computer software licenses through the use of physical device recognition. In particular, FIG. 1 shows an exemplary schematic diagram for a system for remote updating of at least one client device 100 by an update server 120 in accordance with an exemplary embodiment of the invention. In particular, FIG. 1, shows an exemplary system having at least one computing/network client device 100 that is in operative communication via the Internet 102 with an audit server 110 and an update server 120. While only one client device 100 is illustrated in FIG. 1, it will be understood that a given system may comprise any number of client devices and use any number of apparatuses and methods of the invention as described herein. Further details regarding the system of FIG. 1 are provided below.

APPLE EXHIBIT 1102, Page 7 of 1048

[0017] FIG. 2 provides an exemplary flow diagram of a method for remotely updating a client device. In particular, in step 210 of FIG. 2, the loading of at least one client device with a computer program for remote updating is performed. Physical device recognition of at least one of a software, hardware and geo-location environment of the client device is performed to determine machine parameters in step 220. Step 230 involves generating unique device identifiers, at least in part, from the determined machine parameters. The unique device identifiers are sent to at least one of an audit server 110 and an update server 120, as shown in FIG. 1, in step 240. In step 250, the client device 100 receives an updated program configuration from the update server 120, as shown in FIG. 1.

[0018] FIG. 3 illustrates an exemplary apparatus that may be configured as a client device, comprising: a transceiver 304, a processor 306 and a memory 308; or as a processor 306; or as a similar device for use within a client device 100, as shown in FIG. 1, which provides the means for implementing the method, as disclosed in FIG. 2, on the client device 100. In particular, apparatus 300 may comprise means for loading 320 a client device with a computer program for performing a remote update. In addition, the apparatus 300 may comprise means for performing 320 physical device recognition of one or more machine parameters of the client device. The machine parameters may comprise a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the client device. Further, the apparatus 300 may comprise a means for generating 340 a device identifier for the device based at least in part on the collected one or more machine parameters. Furthermore, apparatus 300 may comprise means for sending unique identifiers to at least one of an audit server and an update server. Moreover, apparatus 300 may comprise means for receiving 360 an updated program configuration from the update server.

[0019] In addition, apparatus 300 may further comprise a means for determining the geo-location code for the device and for associating the geo-location code with a unique device identifier; and a software identifier to generate an audit number. The geo-location code may comprise, but is not limited to an Internet protocol (IP) address.

[0020] The apparatus 300 may further comprise a means for generating a device identifier by implementing or executing at least one irreversible transformation such that the machine

APPLE EXHIBIT 1102, Page 8 of 1048

parameters cannot be derived from the device identifier. Additionally, at least one of the irreversible transformations may comprise, but is not limited to a cryptographic hash function.

[0021] It is noted that apparatus 300 may optionally include a processor module 306 having at least one processor, in the case of apparatus 300 configured as computing device, rather than as a processor. Processor module 306, in such case, may be in operative communication with means for determining the geo-location code; means for generating a device identifier by implementing or executing at least one irreversible transformation and components thereof, via a bus 302 or similar communication coupling. Processor 306 may effect initiation and scheduling of the processes or functions performed by means for generating a device identifier by implementing or executing at least one irreversible transformation, and components thereof.

[0022] In related aspects, apparatus 300 may include a transceiver module 304 for communicating with means for generating a device identifier by implementing or executing at least one irreversible transformation, and components thereof. A stand alone receiver and/or stand alone transmitter may be used in lieu of or in conjunction with the transceiver 304.

[0023] In addition, apparatus 300 may optionally include a means for storing information, such as, for example, a computer readable medium or memory device/module 308. Further, the memory device/module 308 may be operatively coupled to the other components of apparatus 300 via bus 302 or the like. The computer readable medium or memory device 308 may be adapted to store computer readable instructions and data for effecting the methods of FIG. 2; and, as shown in FIG. 3, the processes and behavior of means 320-360; means for determining the geo-location code; means for generating a device identifier by implementing or executing at least one irreversible, and components thereof; or processor 306 (in the case of apparatus 300 being configured as a computing device) or the methods disclosed herein.

[0024] In yet further related aspects, the memory module 308 may optionally include executable code for the processor module 304 configured to: (a) determine machine parameters of a client device, the machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the device; (b) generate a device identifier for the device based, at least in part, on the machine parameters; and (c) determine whether an Internet connection is available for the client device. One or more of

APPLE EXHIBIT 1102, Page 9 of 1048

steps (a)-(c) may be performed by a processor module in lieu of or in conjunction with the means described above.

[0025] FIG. 4 shows an exemplary flow diagram of a method for remote updating of a client device in accordance with an embodiment of the invention on the update server 120, as shown in FIG. 1. In particular, in step 410 the unique identifiers are collected by update server 120 from at least one of the audit server 110 and the client device 100, as shown FIG. 1. The unique identifiers are then analyzed on the update server in step 420. Step 430 involves determining an updated program configuration for the client device from the analysis of the unique identifiers. The updated program configuration is delivered to the client device in step 440.

[0026] FIG. 5 shows an exemplary apparatus that may be configured as either an update server, or as a processor or similar device for use within the update server. an exemplary apparatus diagram that may be configured as an update server comprising: a transceiver 504, a processor 506 and a memory 508; or as a processor 506; or as a similar device for use within an update server 120, as shown in FIG. 1, which provides the means for implementing the method, as disclosed in FIG. 4, on the update server 120, as disclosed in FIG. 1. In particular, apparatus 500 may comprise means for collecting 520 unique identifiers from at least one of an audit server and at least one client device with a computer program for performing a remote update. In addition, the apparatus 300 may comprise means for analyzing 530 the unique identifiers that are determined, at least in part, from the machine parameter. The machine parameters may comprise a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the client device. Further, the apparatus 500 may comprise a means for determining 540 an updated program configuration for the device based, at least in part, on the collected one or more machine parameters. Furthermore, apparatus 500 may comprise means for delivering the updated program configuration for the client device 100 from the update server 120, as shown in FIG. 1.

[0027] It is noted that apparatus 500 may optionally include a processor module 506 having at least one processor, in the case of apparatus 500 configured as a computing device, rather than as a processor. In related aspects, apparatus 500 may include a transceiver module 504 for communicating with means for generating a device identifier by implementing or executing at

least one irreversible transformation, and components thereof. A stand alone receiver and/or stand alone transmitter may be used in lieu of or in conjunction with the transceiver 504.

[0028] In addition, apparatus 500 may optionally include a means for storing information, such as, for example, a computer readable medium or memory device/module 508. Further, the memory device/module 508 may be operatively coupled to the other components of apparatus 500 via bus 302 or the like. The computer readable medium or memory device 508 may be adapted to store computer readable instructions and data for effecting the methods of FIG. 4; and, as shown in FIG. 5, the processes and behavior of means 520-550, and components thereof; or processor 506 (in the case of apparatus 300 being configured as a computing device).

[0029] In yet further related aspects, the memory module 508 may optionally include executable code for the processor module 504 configured to: (a) collect unique identifiers from at least one of an audit server and client device; (b) analyze the collected unique identifiers; (c) determine an updated program configuration for the client device; and (d) deliver the updated program configuration to the client device(s). One or more of steps (a)-(d) may be performed by a processor module in lieu of or in conjunction with the means described above.

[0030] FIG. 6, discloses, for one or more embodiments described herein, an exemplary format for a unique device identifier 600, which may further include two components: (1) a variable key portion; and (2) a system key portion. The variable key portion may be generated at the time of registration of client device 100 by reference to a variable platform parameter, such as, but not limited to: a reference to system time information, location and/or other parameters that are variable in nature may be utilized in other embodiments. The system key portion may include the above described parameters expected to be unique to the client device 100, that are for example, but not limited to: hard disk volume name, user name, computer name, user password, hard disk initialization date, or combinations thereof. The variable key portion and/or system key portion may be combined with the IP address and/or other platform parameters of the client device 100. It is noted that unique device identifiers, or portions thereof, may be encrypted to add an additional layer of specificity and security.

[0031] With respect to the system, method and apparatus of the invention, the following paragraphs provide additional detail regarding the implementation of each of the embodiments discussed above.

[0032] The machine parameters may further include, but are not limited to: user account information, program information (e.g., serial number); location of a user within a given application program, and features of the software/hardware the user is entitled to use. As shown in FIG. 1, block 107, the updated program configuration delivered to the client device may include, but is not limited to: binary, executables, paths, dlls, miss or assets.

[0033] The client device 100 may be, but is not limited to, a personal computer, a server computer, a laptop computer, a tablet computer, a personal digital assistant, a mobile phone, a wireless communication device, an onboard vehicle computer, a game console, or any other machine/device capable of communication with a computer network, such as but not limited to the Internet. In related aspects, in wireless communications, Over The Air (OTA) Push or the like may be implemented to download onto or upgrade (e.g., configuration/settings, etc.) client network devices. OTA Push involves the use of wireless phone numbers (MS-ISDN) rather than IP.

[0034] The client device 100 may comprise software (e.g., an operating system or other applications) that requires a license to be authorized for use. The client device 100 may further comprise an auditing tool or application. The auditing application may be any program or application that collects identifying information regarding the client device 100 and/or software on the client device 100. The auditing application may comprise a stand alone application or an applet running within a web browser on the client device 100 (e.g., an applet comprising executable code for a Java Virtual Machine).

[0035] The auditing application may be embedded in or associated with another software application, including, but not limited to software. For example, the auditing application may be embedded in or associated with a tool bar of a software application, for example, but not limited to a web browser. The auditing application may prompt the user to register with an online software registration service, or may run in the background with little or no interaction with the user of the client device 100.

[0036] The auditing application may include a registration routine that collects information regarding client device 100 by checking a number of parameters which are expected to be unique to the client device environment. The parameters checked may include, but are not limited to: hard disk volume name, user name, device name, user password, hard disk initialization date, etc.

The collected information may include, but is not limited to: information that identifies the hardware comprising the platform on which the web browser runs, such as, CPU number, or other unique parameters associated with the firmware in use. The system information may further include, but is not limited to: system configuration information, amount of memory, type of processor, software or operating system serial number, etc.

[0037] In the alternative, or in addition, the parameters checked may include, but are not limited to virtual machine specifications. Examples of virtual machine specifications may include, but are not limited to: information relating to virtual processors, virtual BIOS, virtual memory, virtual graphics, virtual IDE drives, virtual SCSI devices, virtual PCI slots, virtual floppy drives, virtual serial (COM) ports, virtual parallel (LPT) ports, virtual keyboard, virtual mouse and drawing tablets, virtual Ethernet card, virtual networking, virtual sound adapter, etc.

[0038] Based on the collected information, the auditing application may generate a device identifier that is unique for the client device 100. In the alternative, or in addition, the auditing application may gather and send the device parameters to a remote server, such as audit server 110, which in turn generates the device identifier. The device identifier may be stored in a hidden directory of the client device 100 and/or at a remote location, such as the audit server 110. The device identifier may incorporate the device's IP address and/or other geo-location code (e.g., GPS data, cell site triangulation data, or the like, or combinations thereof) to add another layer of specificity to client device's unique identifier.

[0039] An application (e.g., auditing application) running on the client device 100 or otherwise having access to the hardware and file system of the client device 100 may generate a device identifier (e.g., a unique device identifier) using a process that operates on data indicative of the configuration and hardware of the client device 100. The device identifier may be generated using a combination of user-configurable and non-user-configurable machine parameters as input to a process that results in the device identifier, which may be expressed in digital data as a binary number.

[0040] Each machine parameter is data determined by a hardware component, software component, or data component specific to the client device 100. Machine parameters may be selected based on the target device system configuration such that the resulting device identifier

APPLE EXHIBIT 1102, Page 13 of 1048

has a very high probability (e.g., greater than 99.999%) of being a unique identifier of the client device 100.

[0041] In addition, the machine parameters may be selected such that the device identifier includes at least a stable unique portion up to and including the entire identifier that has a very high probability of remaining unchanged during normal operation of the client device 100. As a result, the device identifier should be highly specific, unique, reproducible and stable as a result of properly selecting the machine parameters.

[0042] The application for generating the unique device identifier may also operate on the collected parameters with one or more algorithms to generate the device identifier. This process may include at least one irreversible transformation, such as, but not limited to a cryptographic hash function. As a result, the input machine parameters cannot be derived from the resulting device identifier. Thus, each device identifier, to a very high degree of certainty, cannot be generated except by the suitably configured application operating or otherwise having had access to the same client device for which the device identifier was first generated. Conversely, each device identifier, again to a very high degree of certainty, can be successfully reproduced by the suitably configured application operating or otherwise having access to the same client device 100 on which the device identifier was first generated.

[0043] The auditing application may operate by performing a system scan to determine a present configuration of the client device. The auditing application may then select the machine parameters to be used as input for generating the unique device identifier. Selection of parameters may vary depending on the system configuration. Once the parameters are selected, the application may generate the device identifier.

[0044] Further, generating the device identifier may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties as well as a variety of additional parameters such as uniquely generated hashes and time sensitive values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms. Physical device parameters available for sampling may include, but are not limited to: unique manufacturer characteristics, carbon and silicone degradation and small device failures.

APPLE EXHIBIT 1102, Page 14 of 1048

[0045] The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as, but not limited to: heat and additional stresses placed on a chip during the sampling process. This approach allows for each and every benchmark to reproduce the expected values.

[0046] In addition to the chip benchmarking and degradation measurements, the process for generating a device identifier may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

[0047] Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further extended by the use of complex encryption algorithms to convolute the device identifier values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

[0048] The device identifier may be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag. Further, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

[0049] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: memory model; memory slots; memory total; and memory details. Further, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: video model; video details; display model; display details; audio model; and audio details.

APPLE EXHIBIT 1102, Page 15 of 1048

[0050] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: network model; network address; Bluetooth address; BlackBox model; BlackBox serial; BlackBox details; BlackBox damage map; BlackBox volume name; NetStore details; and NetStore volume name. Furthermore, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

[0051] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag. Moreover, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

[0052] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller. Further, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

[0053] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type. Furthermore, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

[0054] The device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD. Moreover, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

APPLE EXHIBIT 1102, Page 16 of 1048

[0055] In one example, the device identifier may also be generated by utilizing machine parameters associated with, but not limited to, one or more of the following: machine model, processor model, processor details, processor speed, memory model, memory total, network model of each Ethemet interface, network MAC address of each Ethernet interface, BlackBox Model, BlackBox Serial (e.g., using Dallas Silicone Serial DS-2401 chipset or the like), OS install date, nonce value, and nonce time of day.

[0056] Further, with reference once again to FIG. 1 the auditing application may also include a registration routine that collects or receives information regarding the software on the client device 100 by checking information which is expected to be unique to software, for example, but not limited to the software serial number. The collected software identifier may include, but is not limited to: the software serial number, product identification number, product key, etc. The collected software identifier may include, but is not limited to: information regarding where the software was sold or distributed, who the buyers, sellers, and/or distributors were, which stores the software was sold in, etc.

[0057] The software identifier may be unique to particular copy of software, such as when the software is licensed to a single user. In the altemative, or in addition, the software identifier may be unique to particular type or group of software, such as when the software is licensed to a defined group of users.

[0058] The embodiments described herein comprise an auditing application that collects the software identifier for software on the client devices. However, it will be understood that the systems, methods and components described herein can be adapted to collect one or more types of software identifiers for a plurality of software applications. The software identifier may be stored in a hidden directory of the client device 100 and/or at a remote location, such as the audit server 110. For example, in one approach, the software identifier, device identifier, and/or combinations thereof may be hidden in multiple locations on the client device 100 and may be crosschecked for tampering, corruption, etc. In another approach, the software identifier, device identifier, and/or combinations thereof may be hidden in multiple locations, including one or more remote locations/servers, and may be crosschecked with each other to verify the integrity of the identifiers.

[0059] The auditing application may also include a registration routine that collects or receives information regarding the geo-location code of the client device 100. The geo-location code may comprise, but is not limited to: the IP address, GPS data, cell site triangulation data, or the like for the client device 100.

[0060] Auditing application may electronically send the device identifier and the software identifier to the auditing server 110 or directly to the update server 120 via the Internet 102. In the alternative, or in addition, a geo-location code may be associated with the device identifier and/or the software identifier and may be sent to the auditing server 110 or directly to the update server 120, via a secured network connection or via the Internet 102. Further, the client device 100 or the auditing server 110 may encrypt and store the data, such as the device identifier, the software identifier, and/or the geo-location code, received from the client device 100. In addition, the auditing server 110 may receive such data from a plurality of client devices and store the received data in an audit database.

[0061] In one embodiment, the auditing application may generate an audit number by associating the software identifier with the device identifier and/or geo-location code, and may send the generated audit number to the audit server 110 or store the audit number in the client device 100.

[0062] In another embodiment, the auditing application may send the device identifier, the software identifier, and/or the geo-location code to the audit server 110 in a piecemeal manner. The audit server 110 may in turn generate the audit number. The audit server 110 may receive or generate audit numbers from a plurality of client devices 110 and store the received audit numbers in the audit database.

[0063] It is noted that the audit number may be generated from the device identifier, the software identifier, and/or the geo-location code via any number of suitable approaches. For example, the software identifier may be concatenated or linked with the device identifier and/or geo-location code. It is also noted that the audit number may be stored in a hidden directory of the client device 100 and/or at a remote location, such as the audit server 110. It is further noted that the device identifier, the software identifier, and/or the geo-location code may at a later time be extracted from the audit number.

APPLE EXHIBIT 1102, Page 18 of 1048

[0064] When a user of a client device, including but not limited to client device 110, installed with auditing application, attempts to run software, the auditing application in response may transmit the software identifier associated with the device identifier and/or the geo-location code (or an audit number generated from such data) to the audit server 110, which in tum may store the received data in the audit database.

[0065] With reference to the embodiment of FIG. 1, the audit server 110 may be in operative communication with an upgrade server 120, which may be any device, for example, but not limited to: personal computer, a server computer, a laptop computer, a tablet computer, a personal digital assistant, a mobile phone, or a wireless communication device, that is capable of communication with a computer network, such as the Internet. The upgrade server 120 may comprise a remote update application, which may be any program or application, such as a stand alone application or an application that is embedded or associated with another software application, such as an applet running within a web browser on the upgrade server 120.

[0066] The remote update application may be adapted to allow a user, for example, but not limited to a software manufacturer or distributor, to view the data collected and stored in the audit database of the client device 100, audit server 110 or that is collected from the client device 100. The present embodiment will be described in the context of a software manufacturer utilizing the remote update application. However, it will be understood that any user of the remote update server 120 may utilize the remote update application.

[0067] The remote update application may present the data in the audit database or that which is collected from the client device 100 in a manner that allows its user to better understand how its software is being used, legitimately or otherwise.

[0068] While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

APPLE EXHIBIT 1102, Page 19 of 1048

[0069] As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0070] It is understood that the specific order or hierarchy of steps in the processes disclosed herein in an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure. The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

[0071] Moreover, various aspects or features described herein can be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer-readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips, etc.), optical discs (e.g., compact disc (CD), digital versatile disc (DVD), etc.), smart cards, and flash memory devices (e.g., Erasable Programmable Read Only Memory (EPROM), card, stick, key drive, etc.). Additionally, various storage media described herein can represent one or more devices and/or other machine-readable media for storing information. The term "machine-readable medium" can include, without being limited to, wireless channels and various other media capable of storing, containing, and/or carrying instruction(s) and/or data.

APPLE EXHIBIT 1102, Page 20 of 1048

[0072] Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 21 of 1048

CLAIMS

What is claimed is:

1. A system for remotely updating a program configuration, comprising:

a client device configured to execute a computer program to perform a remote update, the client device comprising:

a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates unique device identifiers for the client device, the unique device identifiers based at least in part on the determined machine parameters; and

a first transceiver configured to send the unique device identifiers to at least one server via Internet; and

an update server configured to collect the unique device identifiers from at least one client device, the update server comprising:

a second processor coupled to memory and configured to analyze the unique device identifiers at the update server, and to determine based on the analyzed unique device identifiers an updated program configuration; and

a second transceiver configured to deliver via the Internet data representing the updated program configuration to the client device for storage therein.

2. The system of claim 1 wherein the unique device identifier comprises a hash code.

3. The system of claim 1 wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.

4. The system of claim 3 wherein the at least one irreversible transformation comprises a cryptographic hash function.

5. The system of claim 1 wherein the unique identifiers further comprise software identifiers and geo-location identifiers.

APPLE EXHIBIT 1102, Page 22 of 1048

6. The system of claim 5 wherein at least one of the geo-location identifiers comprises an Internet Protocol address of the client device.

7. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

8. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.

9. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model.

10. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.

11. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.

12. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.

13. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.

14. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type.

APPLE EXHIBIT 1102, Page 23 of 1048

15. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.

16. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.

17. The system of claim 1 wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.

18. A method for remote update of a program, comprising:

collecting, at an update server, unique identifiers from at least one of an audit server and client device;

analyzing the unique identifiers;

determining an updated program configuration for the client device based on the analyzed unique identifiers; and

delivering the updated program configuration to the client device.

19. The method of claim 18 wherein the determining step comprises the update server comparing each analyzed unique identifier to known identifiers stored in a database to determine whether a match exists.

20. The method of claim 19 wherein the determining step further comprises the update server generating the updated program configuration as data representing all matches yielded by the comparing step.

APPLE EXHIBIT 1102, Page 24 of 1048

ABSTRACT

A system for remotely updating a program configuration includes an update server in communication with a client device configured to execute a remote update program. The client device includes a first processor coupled to memory storing the program which, executed, performs physical device recognition on the client device to determine its machine parameters, and generates unique device identifiers based thereon, and a first transceiver configured to send the identifiers to the update server. The update server is configured to collect the identifiers from the client device, and includes a second processor for analyzing the identifiers and determining an updated program configuration based on the collected identifiers matching known identifiers, and a second transceiver configured to deliver data representing the updated program configuration to the client device for storage therein.

APPLE EXHIBIT 1102, Page 25 of 1048

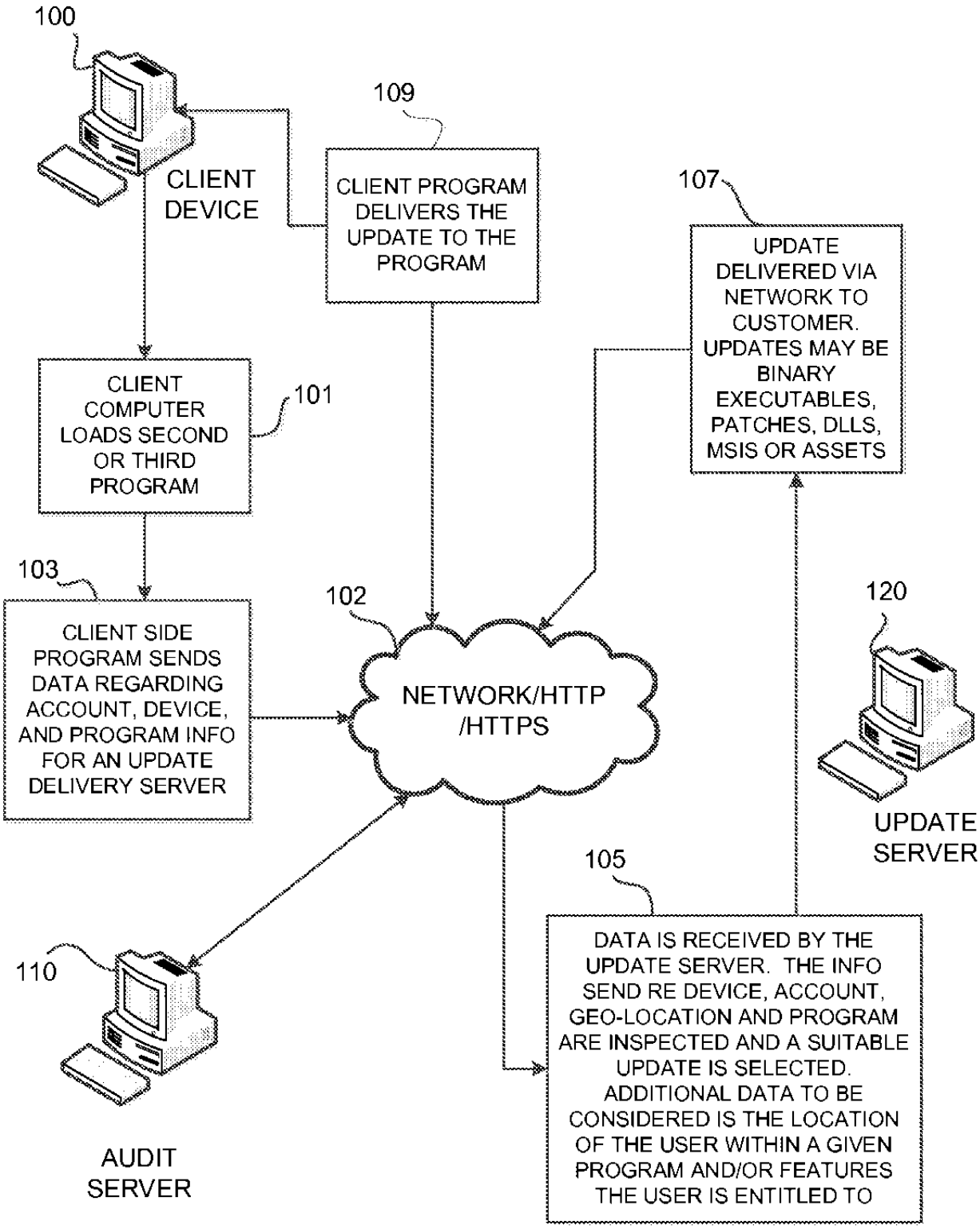

100
CLIENT DEVICE
109
CLIENT PROGRAM DELIVERS THE UPDATE TO THE PROGRAM
107
UPDATE DELIVERED VIA NETWORK TO CUSTOMER. UPDATES MAY BE BINARY EXECUTABLES, PATCHES, DLLS, MSIS OR ASSETS
CLIENT COMPUTER LOADS SECOND OR THIRD PROGRAM
101
103
CLIENT SIDE PROGRAM SENDS DATA REGARDING ACCOUNT, DEVICE, AND PROGRAM INFO FOR AN UPDATE DELIVERY SERVER
102
NETWORK/HTTP /HTTPS
120
UPDATE SERVER
105
DATA IS RECEIVED BY THE UPDATE SERVER. THE INFO SEND RE DEVICE, ACCOUNT, GEO-LOCATION AND PROGRAM ARE INSPECTED AND A SUITABLE UPDATE IS SELECTED. ADDITIONAL DATA TO BE CONSIDERED IS THE LOCATION OF THE USER WITHIN A GIVEN PROGRAM AND/OR FEATURES THE USER IS ENTITLED TO
110
AUDIT SERVER

FIG. 1

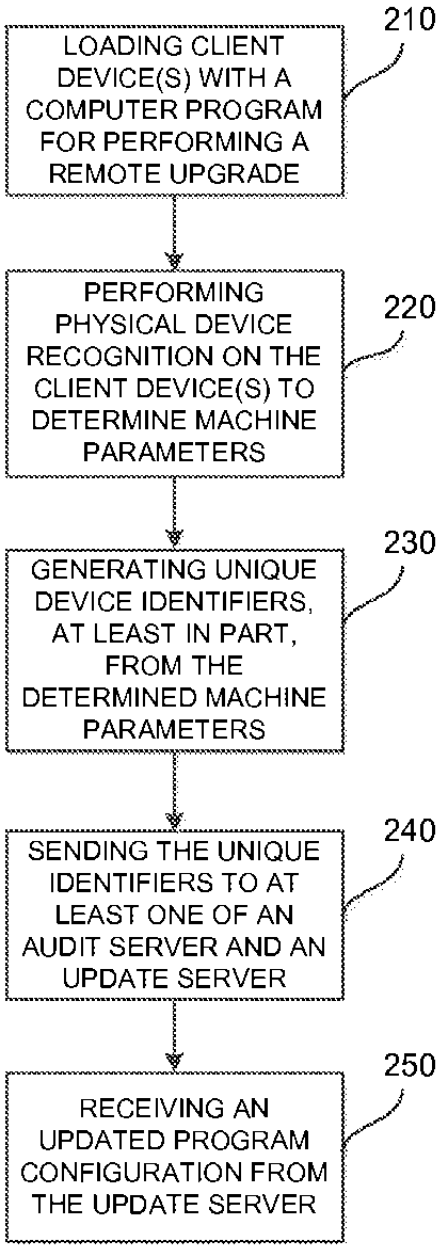

210
LOADING CLIENT DEVICE(S) WITH A COMPUTER PROGRAM FOR PERFORMING A REMOTE UPGRADE
220
PERFORMING PHYSICAL DEVICE RECOGNITION ON THE CLIENT DEVICE(S) TO DETERMINE MACHINE PARAMETERS
230
GENERATING UNIQUE DEVICE IDENTIFIERS, AT LEAST IN PART, FROM THE DETERMINED MACHINE PARAMETERS
240
SENDING THE UNIQUE IDENTIFIERS TO AT LEAST ONE OF AN AUDIT SERVER AND AN UPDATE SERVER
250
RECEIVING AN UPDATED PROGRAM CONFIGURATION FROM THE UPDATE SERVER

FIG. 2

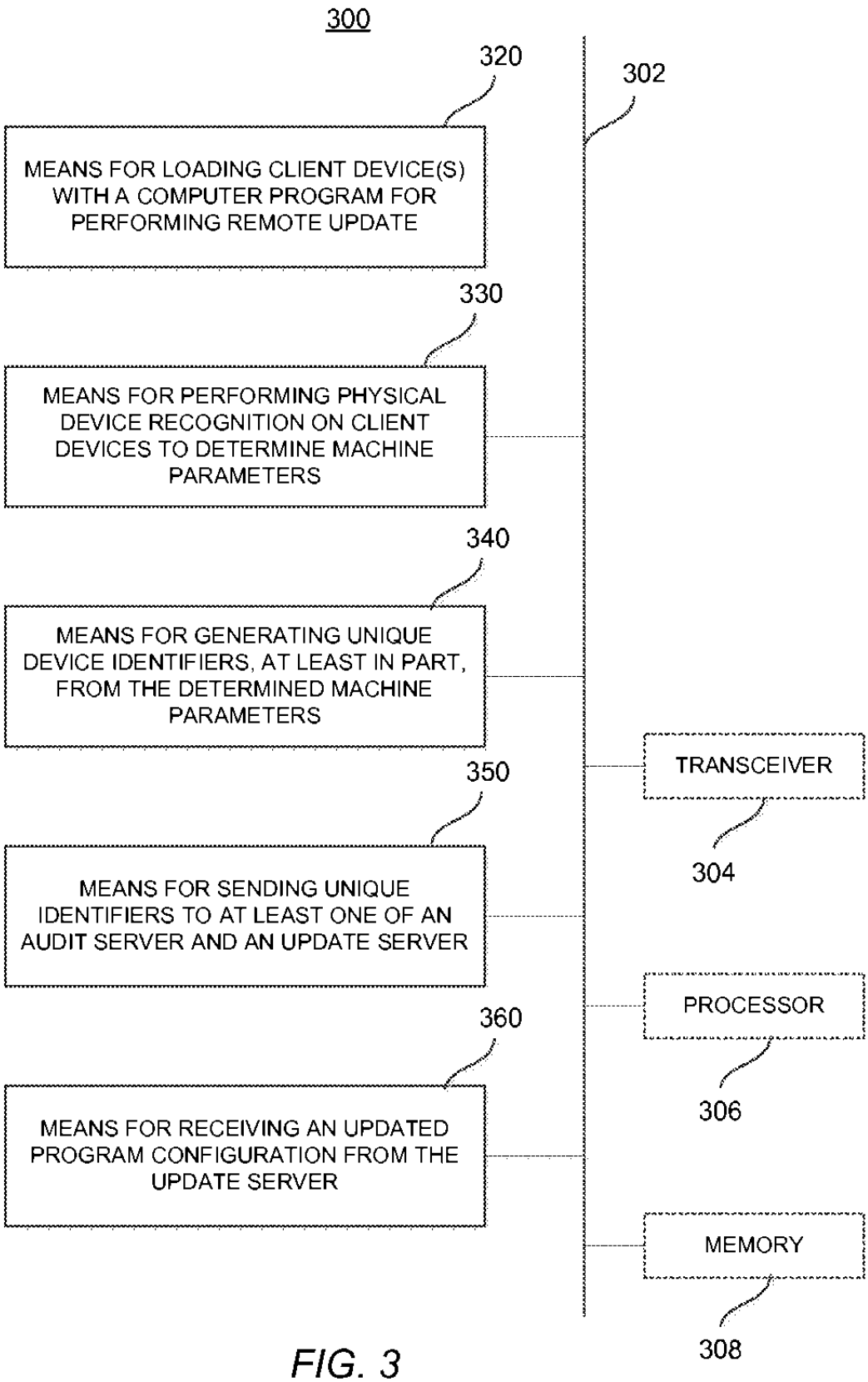

300
320
302
MEANS FOR LOADING CLIENT DEVICE(S) WITH A COMPUTER PROGRAM FOR PERFORMING REMOTE UPDATE
330
MEANS FOR PERFORMING PHYSICAL DEVICE RECOGNITION ON CLIENT DEVICES TO DETERMINE MACHINE PARAMETERS
340
MEANS FOR GENERATING UNIQUE DEVICE IDENTIFIERS, AT LEAST IN PART, FROM THE DETERMINED MACHINE PARAMETERS
350
MEANS FOR SENDING UNIQUE IDENTIFIERS TO AT LEAST ONE OF AN AUDIT SERVER AND AN UPDATE SERVER
360
MEANS FOR RECEIVING AN UPDATED PROGRAM CONFIGURATION FROM THE UPDATE SERVER
TRANSCEIVER
304
PROCESSOR
306
MEMORY
308

FIG. 3

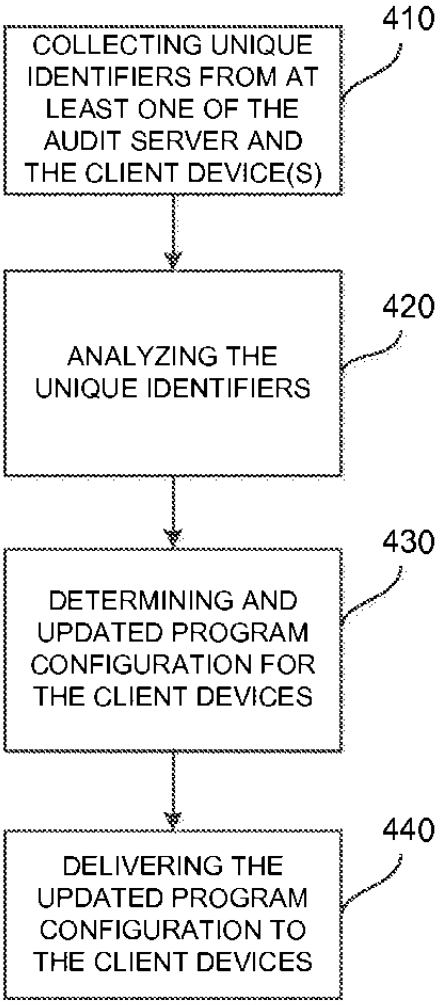

410
COLLECTING UNIQUE IDENTIFIERS FROM AT LEAST ONE OF THE AUDIT SERVER AND THE CLIENT DEVICE(S)
420
ANALYZING THE UNIQUE IDENTIFIERS
430
DETERMINING AND UPDATED PROGRAM CONFIGURATION FOR THE CLIENT DEVICES
440
DELIVERING THE UPDATED PROGRAM CONFIGURATION TO THE CLIENT DEVICES

FIG. 4

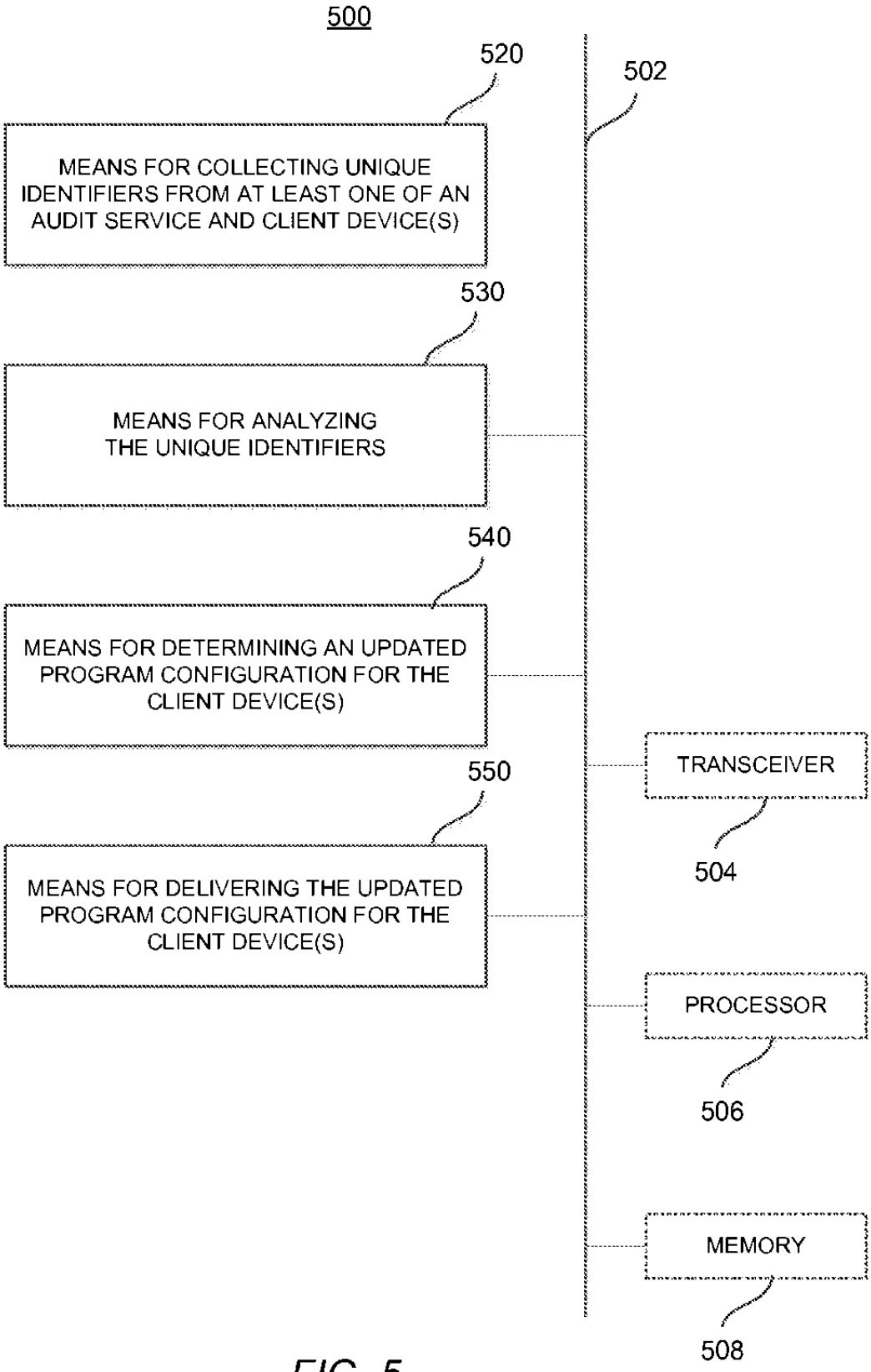

500
520
502
MEANS FOR COLLECTING UNIQUE IDENTIFIERS FROM AT LEAST ONE OF AN AUDIT SERVICE AND CLIENT DEVICE(S)
530
MEANS FOR ANALYZING THE UNIQUE IDENTIFIERS
540
MEANS FOR DETERMINING AN UPDATED PROGRAM CONFIGURATION FOR THE CLIENT DEVICE(S)
550
MEANS FOR DELIVERING THE UPDATED PROGRAM CONFIGURATION FOR THE CLIENT DEVICE(S)
TRANSCEIVER
504
PROCESSOR
506
MEMORY
508

FIG. 5

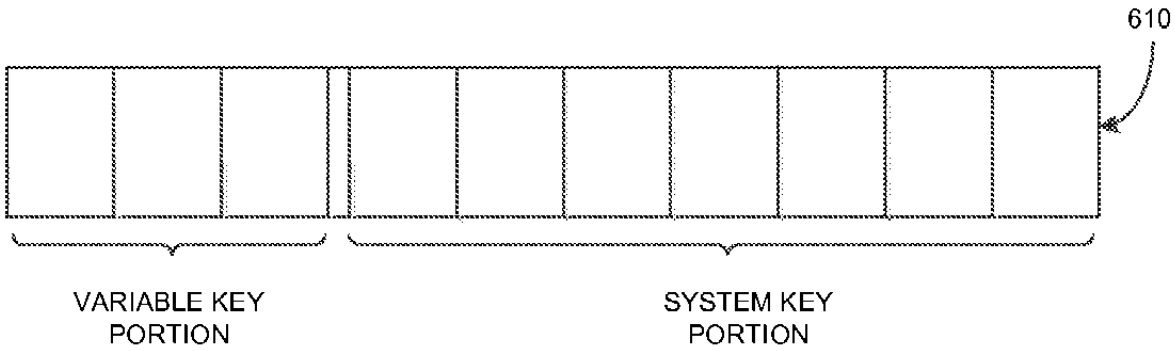


FIG. 6

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig S. Etchegoyen
Filer:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Utility filing Fee (Electronic filing)	4011	1	82	82
Utility Search Fee	2111	1	270	270
Utility Examination Fee	2311	1	110	110
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	462

Electronic Acknowledgement Receipt

EFS ID:	7848910
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig S. Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick
Filer Authorized By:	
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	18-JUN-2010
Filing Date:	
Time Stamp:	17:25:01
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$462
RAM confirmation Number	3970
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1		sb001_and_sb081.pdf	506312 f9323108Sba50c779565d73890d4d20a20c58a11	yes	4

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Power of Attorney	1	1
Oath or Declaration filed	2	4

Warnings:

Information:

2		specification.pdf	106951 a09de6370c1fbd055fa3ac2b30ae0e63b86b228a	yes	21

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Specification	1	17
Claims	18	20
Abstract	21	21

Warnings:

Information:

3	Drawings-only black and white line drawings	figures.pdf	366500 260639c03bfa5cd30b6f4ceab90c049ce90befe3	no	6

Warnings:

Information:

4	Fee Worksheet (PTO-875)	fee-info.pdf	32692 179d0457d22ddd08a0dd37291483e83d1f775302	no	2

Warnings:

Information:

Total Files Size (in bytes):	1012455

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

SCORE Placeholder Sheet for IFW Content

Application Number: 12818906 Document Date: 6/18/2010

The presence of this form in the IFW record indicates that the following document type was received in electronic format on the date identified above. This content is stored in the SCORE database.

- Drawings – Other than Black and White Line Drawings

Since this was an electronic submission, there is no physical artifact folder, no artifact folder is recorded in PALM, and no paper documents or physical media exist. The TIFF images in the IFW record were created from the original documents that are stored in SCORE.

To access the documents in the SCORE database, refer to instructions developed by SIRA.

At the time of document entry (noted above):

- Examiners may access SCORE content via the eDAN interface.
- Other USPTO employees can bookmark the current SCORE URL (http://es/ScoreAccessWeb/).
- External customers may access SCORE content via the Public and Private PAIR interfaces.

Form Revision Date: February 8, 2006

PTO/SB/06 (10-07)
Approved for use through 06/30/2010. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Date: 06/18/2010

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: **12/818,906**

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	82		N/A	
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	270		N/A	
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	110		N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	20	*	X 26=		OR	X 52=	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	2	*	X 110=			X 220=	
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $270 ($135 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR						
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))			195			390	
			TOTAL	462		TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2.

APPLICATION AS AMENDED – PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	X =		OR	X =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X =		OR	X =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	X =		OR	X =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X =		OR	X =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))					N/A		OR	N/A	
						TOTAL ADD'T FEE		OR	TOTAL ADD'T FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
12/818,906	06/18/2010	2447	462	UN-NP-AD-037	20	2

CONFIRMATION NO. 8831

96051
Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612

FILING RECEIPT

Date Mailed: 07/02/2010

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Applicant(s)

Craig Stephen Etchegoyen, Irvine, CA;

Power of Attorney: The patent practitioners associated with Customer Number 96051

Domestic Priority data as claimed by applicant

This appln claims benefit of 61/220,092 06/24/2009

Foreign Applications

Permission to Access - A proper **Authorization to Permit Access to Application by Participating Offices** (PTO/SB/39 or its equivalent) has been received by the USPTO.

If Required, Foreign Filing License Granted: 07/01/2010

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 12/818,906**

Projected Publication Date: 12/30/2010

Non-Publication Request: No

Early Publication Request: No
**** SMALL ENTITY ****

Title

Remote Update of Computers Based on Physical Device Recognition

Preliminary Class

709

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4158).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as

set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037

CONFIRMATION NO. 8831

96051
Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612

POA ACCEPTANCE LETTER

Date Mailed: 07/02/2010

NOTICE OF ACCEPTANCE OF POWER OF ATTORNEY

This is in response to the Power of Attorney filed 06/18/2010.

The Power of Attorney in this application is accepted. Correspondence in this application will be mailed to the above address as provided by 37 CFR 1.33.

/hnguyen/

Office of Data Management, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-4351982	09/28/1982	Miller et al.	
		US-4658093	04/14/1987	Hellman	
		US-4704610	11/03/1987	Smith et al.	
		US-4796220	01/03/1989	Wolfe	
		US-5210795	05/11/1993	Lipner et al.	
		US-5291598	03/01/1994	Grundy	
		US-5414269	05/09/1995	Takahashi	
		US-5418854	05/23/1995	Kaufman et al.	
		US-5440635	08/08/1995	Bellovin et al.	
		US-5490216	02/06/1996	Richardson, III	
		US-5666415	09/09/1997	Kaufman	
		US-5745879	04/28/1998	Wyman	
		US-5754763	05/19/1998	Bereiter	
		US-5790664	08/04/1998	Coley et al.	
		US-5925127	07/20/1999	Ahmad	
		US-5974150	10/26/1999	Kaish et al.	
		US-6009401	12/28/1999	Horstmann	
		US-6044471	03/28/2000	Colvin	
		US-6158005	12/05/2000	Bharathan et al.	
		US-6230199	05/08/2001	Revashetti et al.	
		US-6233567	05/15/2001	Cohen	
		US-6243468	06/05/2001	Pearce et al.	
		US-6294793	09/25/2001	Brunfeld et al.	
		US-6330670	12/11/2001	England et al.	

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-6449645	09/10/2002	Nash	
		US-6536005	03/18/2003	Augarten	
		US-6785825	08/31/2004	Colvin	
		US-6859793	02/22/2005	Lambiase	
		US-6920567	07/19/2005	Doherty et al.	
		US-6976009	12/13/2005	Tadayon et al.	
		US-7032110	04/18/2006	Su et al.	
		US-7069440	06/27/2006	Aull	
		US-7069595	06/27/2006	Cognigni et al.	
		US-7085741	08/01/2006	Lao et al.	
		US-7188241	03/06/2007	Cronce et al.	
		US-7203966	04/10/2007	Abburi et al.	
		US-7206765	04/17/2007	Gilliam et al.	
		US-7272728	09/18/2007	Pierson et al.	
		US-7319987	01/15/2008	Hoffman et al.	
		US-7337147	02/26/2008	Chen et al.	
		US-7343297	03/11/2008	Bergler et al.	
		US-7327280	02/05/2008	Bachelder et al.	
		US-7463945	12/09/2008	Kiesel et al.	
		US-7653899	01/26/2010	Lindahi et al.	
		US-20010034712	10/25/2001	Colvin	
		US-20010044782	11/22/2001	Hughes et al.	
		US-20020019814	2/14/2002	Ganesan	
		US-20020082997	6/27/2002	Kobata et al.	

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 3 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-20020161718	10/31/2002	Coley et al.	
		US-20030065918	04/03/2003	Willey	
		US-20030172035	09/11/2003	Cronce et al.	
		US-20040024860	02/05/2004	Sato et al.	
		US-20040030912	02/12/2004	Merkle et al.	
		US-20040059929	03/25/2004	Rodgers et al.	
		US-20040143746	07/22/2004	Ligeti et al.	
		US-20040187018	09/23/2004	Owen et al.	
		US-20050108173	05/19/2005	Stefik et al.	
		US-20050138155	06/23/2005	Lewis	
		US-20050172280	08/04/2005	Ziegler et al.	
		US-20060072444	040/6/2006	Engel et al.	
		US-20060095454	05/04/2006	Shankar et al.	
		US-20060265337	11/23/2006	Wesinger, Jr.	
		US-20060161914	07/20/2006	Morrison et al.	
		US-20060282511	12/14/2006	Takano et al.	
		US-20070168288	07/19/2007	Bozeman	
		US-20070198422	08/23/2007	Prahlad et al.	
		US-20070203846	08/30/2007	Kavuri et al.	
		US-20070219917	09/20/2007	Liu et al.	
		US-20070282615	12/06/2007	Hamilton et al.	
		US-20080065552	03/13/2008	Elazar et al.	
		US-20080086423	04/10/2008	Waites	
		US-20080147556	06/19/2008	Smith et al.	

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 4 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-20080228578	09/18/2008	Mashinsky	
		US-20080320607	12/25/2008	Richardson	
		US-20090083730	03/26/2009	Richardson	
		US- 20090138975	05/28/2009	Richardson	

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document Country Code – Number – Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		WO 9220022	11/12/1992	Digital Equip. Corp.		
		WO 9301550	1/21/1993	Infologic Software		
		WO 9535533	12/28/1995	Megalode Corp.		
		AU 678985	6/19/1997	Uniloc Corp. Pty Ltd		
		WO 0067095	11/9/2000	Trymedia Systems		
		WO 2005104686	11/10/2005	IPASS Inc.		
		EP 1637961	3/22/2006	Microsoft Corp.		
		EP 1637958	3/22/2006	Microsoft Corp.		
		EP 1670188	6/14/2006	Alcatel		
		WO2007060516	5/31/2007	Lo		
		WO2008013504	1/31/2008	Starhub Ltd		
		WO2008157639	12/24/2008	Uniloc Corp.		
		WO2009039504	3/26/2009	Uniloc Corp.		
		WO2009065135	5/22/2009	Uniloc Corp.		
		WO2009076232	6/9/2009	Uniloc Corp.		
		WO2009105702	8/27/2009	Etchegoyen		
		WO2009143115	11/26/2009	Etchegoyen		

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 5 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document Country Code – Number – Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		2009158525	12/30/2009	Uniloc USA, Inc.		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date page(s), volume-issue number(s), publisher, city and/or country where published.	T
		WILLIAMS, R., "A Painless Guide to CRC Error Detection Algorithms", Ver. 3, Aug. 19, 1993.	
		ANGHA, F. et al., "Securing Transportation Network Infrastructure with Patented Technology of Device Locking – Developed By Uniloc USA", *avail. at:* http://www.dksassociates.com/admin/paperfile/ITS%20World%20Paper%20Submission_Uniloc%20_2_.pdf, Oct. 24, 2006.	
		ECONOLITE, "Econolite and Uniloc Partner to Bring Unmatched Infrastructure Security to Advanced Traffic Control Networks with Launch of Strongpoint", *avail. at: http://www.econolite.com/docs/press/20080304_Econolite_StrongPoint.pdf*, Mar. 4, 2008.	

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

AU9348113

(12) PATENT ABRIDGMENT (11) Document No. AU-B-48113/93
(19) AUSTRALIAN PATENT OFFICE (10) Acceptance No. 678985

(54) Title
SYSTEM FOR SOFTWARE REGISTRATION

International Patent Classification(s)
(51)[5] **G06F 015/21** **G06F 009/44**

(21) Application No. : **48113/93** (22) Application Date : **20.09.93**

(87) PCT Publication Number : **WO94/07204**

(30) Priority Data

(31) Number	(32) Date	(33) Country
PL4842	**21.09.92**	**AU AUSTRALIA**
PL5524	**26.10.92**	**AU AUSTRALIA**

(43) Publication Date : **12.04.94**

(44) Publication Date of Accepted Application : **19.06.97**

(71) Applicant(s)
UNILOC CORPORATION PTY LIMITED

(72) Inventor(s)
RIC BAILIER RICHARDSON

(74) Attorney or Agent
PETER MAXWELL & ASSOCIATES , PO Box R1466 Royal Exchange, SYDNEY NSW 2000

(56) Prior Art Documents
US 5375240
US 5291598
WO 92/09160

(57) Claim

1. A registration system for licensing execution of digital data in the use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID first generated by said local licensee unique ID generating means has matched a licensee unique ID subsequently generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

OPI DATE 12/04/94 APPLN. ID 48113/93
AOJP DATE 07/07/94 PCT NUMBER PCT/AU93/00483
AU9348113

(51) International Patent Classification 5 : G06F 15/21, 9/44

A1

(11) International Publication Number: WO 94/07204

(43) International Publication Date: 31 March 1994 (31.03.94)

(21) International Application Number: PCT/AU93/00483

(22) International Filing Date: 20 September 1993 (20.09.93)

(30) Priority data:
PL 4842 21 September 1992 (21.09.92) AU
PL 5524 26 October 1992 (26.10.92) AU

(71) Applicant *(for AU only)*: UNILOC CORPORATION PTY LIMITED [AU/AU]; 32 Brookvale Avenue, Brookvale, NSW 2100 (AU).

(71) Applicant *(for all designated States except AU US)*: UNILOC (SINGAPORE) PRIVATE LIMITED [SG/SG]; Six Battery Road #38-01, Singapore 0104 (SG).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)* : RICHARDSON, Ric, Bailier [AU/AU]; 32 Brookvale Avenue, Brookvale, NSW 2100 (AU).

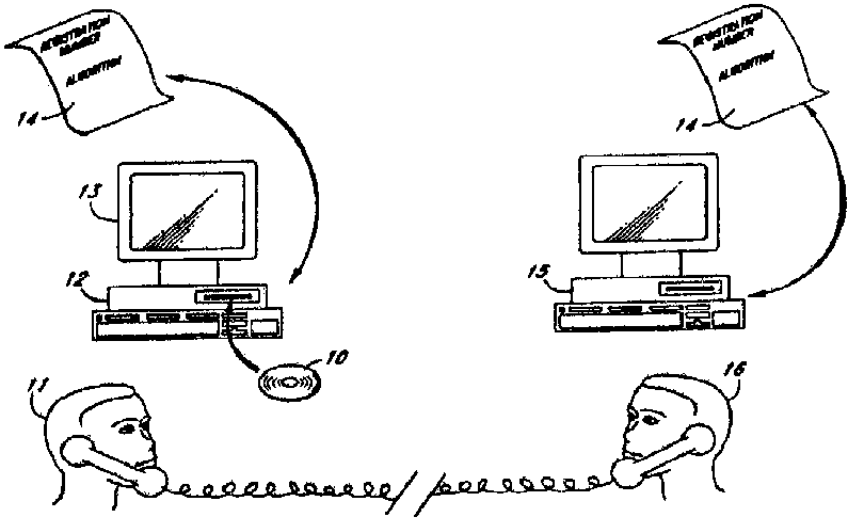
(74) Agent: PETER MAXWELL & ASSOCIATES; 5 Ross Street, North Parramatta, NSW 2151 (AU).

(81) Designated States: AT, AU, BB, BG, BR, BY, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, LV, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, US, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

678985

(54) Title: SYSTEM FOR SOFTWARE REGISTRATION



(57) Abstract

A registration system for licensing execution of digital data in a use mode, said digital data executable on a platform (12), said system including local license unique ID generating means (14), and remote licensee unique ID generating means (14), said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID generated by said local licensee unique ID generating means (14) has matched a licensee unique ID generated by said remote licensee unique ID generating means (14).

SYSTEM FOR SOFTWARE REGISTRATION

FIELD OF THE INVENTION

This invention relates to improvements in systems for software registration and, more particularly, to improvements in arrangements where software is transferable by media such as magnetic disks, CD ROMS and the like.

BACKGROUND ART

Much commercially available software is provided at time of purchase (or licence) on a magnetic media, typically a floppy disk. Frequently the only security feature attached to the software is a simple registration number stored on the media. This registration number identifies that particular copy of the software and it is often required at the time of installation of the software onto any given computer that the installer must provide the registration number independently to the installation routines.

However, such simple security arrangements for the distribution of software on media suffer from at least two disadvantages (1) each copy of the software made on any given media at the time of manufacture must include an individual, unique number, programmed into the media and, (2) this arrangement does not prevent copying of the software, once installed on any given computer, to another computer by means of file transfer (as opposed to reinstallation).

WO 92/09,160 to Tau Systems Corporation discloses a registration system which is relatively sophisticated which

relies for its security on a requirement that an intending software licensee must obtain from a remote location by file transfer significant and essential portions of the programme which the licensee desires to execute. The arrangement disclosed in WO 92/09,160 suffers from a number of deficiencies including:-

(a) the shell programme which the intending licensee initially executes requires a unique identity embodied within the shell prior to distribution of the shell programme;

(b) the shell programme is not, itself, a functional programme - that is, it does not include all of the code which the intending licensee wishes to execute. That programme must be obtained remotely with all the delays, inconveniences and possibilities of corruption during transit that that entails;

(c) the prior art system appears to require and indeed, rely on, encryption to ensure that the programme material which is communicated from a remote location is not intercepted for utilisation in an unauthorised manner;

(d) it is unclear whether the system can accommodate and react appropriately to the situation where the programme, once registered, is transferred in its entirety from one platform to another so as to avoid the requirement for payment of a further registration fee.

U.S. 4,796,220, assigned to Pride Software Development Corporation, discloses a system for unique recognition of a platform on which licensed software is to be executed. However, U.S. 4,796,220 does not contemplate or disclose utilisation of information which is unique to the user or intended licensee as part of the registration process which is to be distinguished from identification of the platform upon which the software is proposed to be run.

U.S. 4,688,169 to Joshi broadly discloses the same principles as U.S. 4,796,220 in that it discloses a computer software security system which relies for its security on a "machine identification code unique to the machine" upon which the software to be protected is to be run. Again, the disclosure is limited to identification of the platform and there is no suggestion or contemplation of linking platform identification with unique user identification.

Also this arrangement does not allow the flexibility of transfer of copies of the programme from platform to platform which can be run in a demonstration mode.

It is an object of the present invention to address or reduce the abovementioned disadvantages.

DEFINITIONS

Throughout this specification the term "software" is to be interpreted broadly so as to include all forms of digital data which are executable on a platform (as to be later defined). The digital data comprising the software can, for example, be code comprising a word processing

programme adapted to run on a PC or the like. The software can also, for example, be digital data stored on a CD ROM adapted for playback as music on a CD ROM audio drive. The digital data can be displayable information or information which is otherwise usable by a licensed user.

Throughout this specification the term "platform" denotes an environment to be associated with a computing device such as a microprocessor or other data processing device which permits execution of the digital data (to which reference has previously been made in relation to the term "software") whereby the computer can perform functions on input and output devices associated therewith.

In some circumstances the "software" or digital data may itself be the operating system environment. Typically, but by no means exclusively, examples of operating system environments include the Microsoft DOS operating system, the IBM OS/2 operating system or the Macintosh System 7 environment. In the degenerate case of microcontrollers operating from ROM the operating system environment may be the microcode of the microcontroller which enables the microcontroller to execute machine code.

In this specification "use mode" refers to use of the digital data or software by its execution on a platform so as to fulfil the seller's/licensor's obligations in relation to the sale or license of the right to execute the digital data or software in the use mode. The use mode is to be distinguished from what might generally be termed unlicensed modes of operation (which is not to say

unauthorised modes of operation) as typified by the demonstration modes later described in this specification.

DISCLOSURE OF THE INVENTION

In broad terms the system according to the invention is designed and adapted to allow digital data or software to run in a use mode on a platform if and only if an appropriate licensing procedure has been followed. In particular forms the system includes means for detecting when parts of the platform on which the digital data has been loaded has changed in part or in entirety as compared with the platform parameters when the software or digital data to be protected was for example last booted or run or validly registered.

The system relies on digital data or code which forms part of the digital data to be protected by the system. This portion of the digital data which preferably is integral to the digital data to be protected has been termed the code portion 38 elsewhere in this specification. The code portion includes an algorithm adapted to generate a registration number which is unique to an intending licensee of the digital data based on information supplied by the licensee which characterises the licensee.

The algorithm in the code portion is duplicated at a remote location on a platform under the control of the licensor or its agents and communication between the intending licensee and the licensor or its agent is required so that a matching registration number can be generated at the remote location for subsequent

communication to the intending licensee as a permissive to licensed operation of the digital data in a use mode.

Preferably the code portion is integral with the digital data and can be identical for all copies of the digital data. It is the algorithm embedded within the code portion (and which is duplicated at the remote location) which provides a registration number which can be "unique" if the information provided by the intending licensee upon which the algorithm relies when executed upon the platform is itself "unique".

In any event in particular preferred forms a serial number (see further on) is included in the registration number generation algorithm which introduces an additional level of uniqueness into the registration number calculation process.

Accordingly in one broad form of the invention there is provided a registration system for licensing execution of digital data in the use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID first generated by said local licensee unique ID generating means has matched a licensee unique ID subsequently generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

Preferably said system further includes platform unique ID generating means, wherein said mode switching means will permit said digital data to

AUSTRALIAN PATENT OFFICE

run in said use mode in subsequent execution of said digital data on said platform only if said platform unique ID has not changed.

Preferably said mode switching means permits operation of said digital data in said use mode in subsequent execution of said digital data only if said licensee unique ID generated by said local licensee unique ID generating means has not changed.

Preferably said mode switching means includes part of said digital data.

Preferably the information utilized by said local licensee unique ID generating means to produce said licensee unique ID comprises prospective licensee details, contact details and name.

Preferably said platform unique ID generating means forms part of said digital data.

Preferably said platform unique ID generating means utilises hard disk information and/or other computer hardware or firmware information to determine said platform unique ID.

Preferably said platform comprises a computer operating system environment.

Preferably said digital data comprises a software programme adapted to run under said operating system environment.

In a further broad form of the invention, there is provided a registration system attachable to software to be protected, said registration system generating a security key from information input to said software which uniquely identifies an intended registered user of said software on a computer on which said software is to be installed; and wherein said registration system is replicated at a registration authority and used for the purposes of checking by the registration authority that the information

AUSTRALIAN PATENT OFFICE

unique to the user is correctly entered at the time that the security key is generated by the registration system.

Preferably said security key is generated by a registration number algorithm.

Preferably said registration number algorithm combines information entered by a prospective registered user unique to that user with a serial number generated from information provided by the environment in which the software to be protected is to run (eg system clock, last modify date, user name).

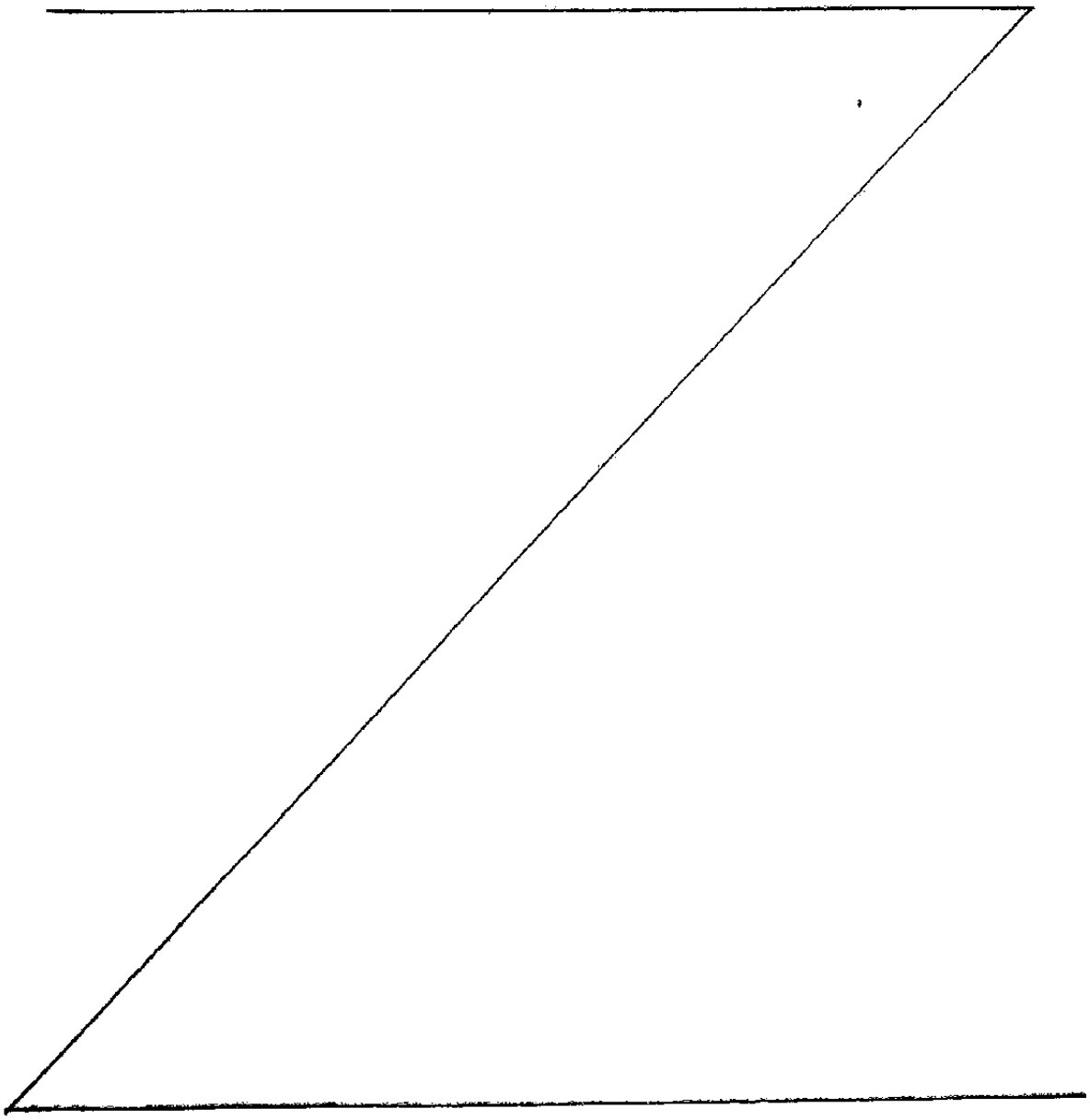
Preferably said registration means checks at the time of boot of said software as to whether it is a first boot of the software to be protected or a subsequent boot. If a subsequent boot is detected then environment and user details are compared to determine whether the programme reverts to a demonstration mode and a new user registration procedure is to commence, or a full version run.

Preferably said environment details comprise one or more of disc volume name, user name or computer, initialisation date of hard disc, hardware identifier (eg. ROM cheksum) or other elements which are generally not user-configurable on the platform.

In a further broad form of the invention there is provided a method of control of distribution of software, said method comprising providing mode-switching means associated with said software adapted to switch said software between a fully enabled mode and a partly enabled or demonstration mode, said method further comprising providing registration key generating means adapted to generate a registration key which is a function of information unique to an intending user of the software; said mode-switching means switching said software into fully enabled mode

AUSTRALIAN PATENT OFFICE

only if an enabling key provided to said mode-switching means by said intending user at the time of registration of said software has matched identically with said registration key; and wherein said enabling key is communicated to said intending user at the time of registration of said software; said enabling key generated by a third party means of operation of a duplicate copy of said registration key generating means.



AUSTRALIAN PATENT OFFICE

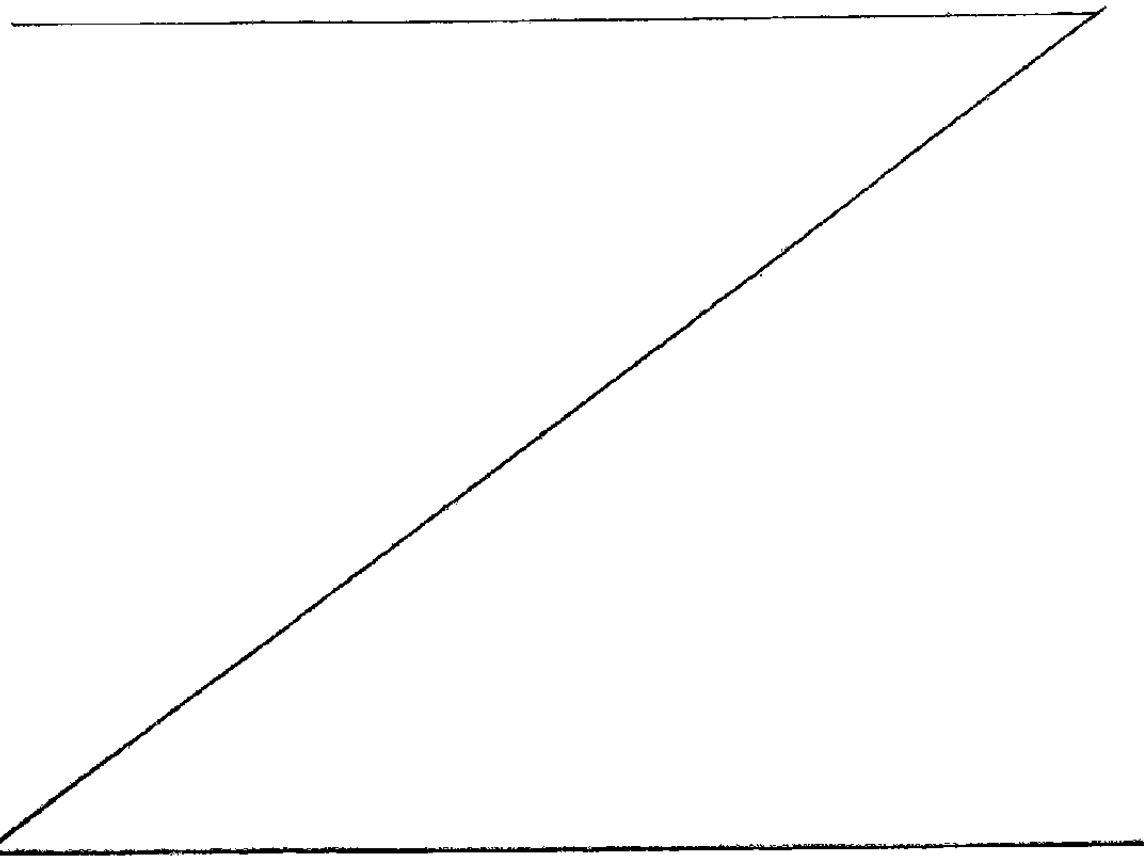
BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will now be described with reference to the accompanying drawings wherein:-

Fig. 1 is a schematic diagram of the relationship and interaction between an intending registered user and a registration authority of software on media secured according to a first embodiment of the invention,

Figs 2a, 2b, 2c are segments of a flow chart of the procedure to be followed during registration of software by a user according to a first embodiment of the invention,

Fig. 3 is a flow chart of alternative boot processes according to a second embodiment of the invention,



AUSTRALIAN PATENT OFFICE

Fig. 4 is a personal information dialogue box relating to the procedure of Figs. 2a, 2b, 2c in accordance with a third embodiment,

Fig. 5 is a schematic diagram of a system according to a fourth embodiment of the invention,

Fig. 6 is an implementation of the fourth embodiment of Fig. 5 in relation to a CD ROM drive,

Fig. 7 is a logic flow chart in relation to the decoder box of Fig. 6,

Fig. 8 is a block diagram of a generalised system according to a fifth embodiment of the invention,

Fig. 9 is a block diagram indicating one particular example of generation of a registration number for the system of Fig. 8 and

Fig. 10 is a schematic diagram of a sixth embodiment comprising a particular example of the generalised system of Fig. 8.

MODES FOR CARRYING OUT THE INVENTION

It is to be understood that, in its various embodiments, the invention is for the protection of digital code/software by control of permission to use the digital code/software. A hardware platform and a remote registration station implemented at least partially by means of electronic hardware are required by the various

embodiments.

The code/software to be protected requires at least some adaption to be useable with the invention in its various embodiments. The adaptation can be universal for all copies of the code/sofware to be protected.

1. FIRST EMBODIMENT

With reference to Figs. 1 and 8 the system according to embodiments of the invention is designed and adapted to allow digital data 39 or software to run in a use mode on a platform 31 if and only if an appropriate licensing procedure has been followed. In particular forms the system includes means for detecting when parts of the platform 31 on which the digital data 39 has been loaded has changed in part or in entirety as compared with the platform parameters when the software or digital data to be protected was for example last booted or run or validly registered.

The system relies on digital data or code 38 which forms part of the digital data to be protected by the system. This portion of the digital data which preferably is integral to the digital data to be protected has been termed the code portion 38 elsewhere in this specification. The code portion 38 includes an algorithm adapted to generate a registration number 66 or local licensee unique ID or registration key which is unique to an intending licensee of the digital data based on information supplied by the licensee which characterises the licensee. In this instance the local licensee unique ID generator which

generates the registration number comprises the execution of code 38 on platform 31.

The algorithm in the code portion is duplicated at a remote location on a platform 67 under the control of the licensor or its agents and communication between the intending licensee and the licensor or its agent is required so that a matching registration number or enabling key can be generated at the remote location for subsequent communication to the intending licensee as a permissive to licensed operation of the digital data 39 in a use mode.

Execution of the duplicated code portion on platform 67 comprises, in this instance, the remote licensee unique ID generating means.

Mode switching means can comprise execution of the code portion which additionally performs a comparison of the locally and remotely generated registration numbers.

Preferably the code portion 38 is integral with the digital data and can be identical for all copies of the digital data. It is the algorithm embedded within the code portion (and which is duplicated at the remote location) which provides a registration number which can be "unique" if the information provided by the intending licensee upon which the algorithm relies when executed upon the platform is itself "unique".

In any event in particular preferred forms a serial number (see further on) is included in the registration number generation algorithm which introduces an additional level of uniqueness into the registration number

calculation process.

With particular reference to Fig. 1 a programme comprising digital data protected according to a first embodiment of the invention is supplied recorded on a magnetic disk 10.

Included as part of the software on that disk 10 is a registration and re-registration routine which executes whenever the programme protected by the arrangement of the first embodiment "boots".

With reference to Figs. 1 and Figs. 2a, 2b and 2c the operation of the security routine will be described on the assumption that the programme on the disk 10 protected by the registration routine has not been registered on the platform or is otherwise being loaded for the first time.

The prospective new user 11 inserts disk 10 into the user PC 12 so as to be read by PC 12.

As part of the software installation procedure the registration routine is activated causing a series of dialogue boxes to appear on the display 13 of the user PC 12. Having checked to ensure that the software has not previously been registered on the PC 12 a dialogue box A (in Fig. 2a) is displayed which provides the user with a choice of either seeing a demonstration of the software (which typically has features such as save and/or print disabled) or alternatively
an invitation to register ownership/license of the software (after which all features of the software are made available to the user).

If the register option is selected or if the user cancels the demonstration in favour of registration then a contact dialogue box B (in Fig. 2a) is presented on the display 13 which provides a list (stored on disk 10 as part of the registration routine) which provides for example, names and contact numbers of the software publishing company together with other general product information.

Following the user's indication of agreement during display of license details (box B1) to proceed to register, the user can contact the registration centre after filling out the registration dialogue box C as detailed below. After selecting "continue", the registration routine begins the first step in the generation of a security key which will be unique to the current copy of the software and to certain features of the environment in which it runs.

As shown in Fig. 2b, the first step in the generation of the security key comprises the generation of a serial number generated from the current time on the system and, in this example, the last modify date of the software and other information from the computer environment. The serial number is encrypted and rearranged and then presented as a number in the registration dialogue box on the display 13.

The registration dialogue box C (in Fig. 2b) prompts the user for details unique to that user (including, for example, name, company, address, state, contact number) together with financial details for payment for the purpose of becoming a registered user of the software protected by

the registration routine (for example Mastercard or corporate account number details). This information, unique to the user, is passed through a registration number algorithm 14 (represented symbolically in Fig. 1) which generates a registration number or security key from the information unique to the user together with the serial number previously generated. The registration number or security key is not made available to the user of the PC 12 by the PC 12.

An identical registration number algorithm 14 resides on the registration authority PC 15. As an integral part of the registration procedure the prospective new user 11 communicates the information unique to the user which was entered by the user on the user PC 12, along with the serial number generated by the user's algorithm, to the registration authority 16. The registration authority feeds this information into the registration authority PC 15 wherein the registration number algorithm 14 should produce an identical registration number or security key to that produced by the user PC 12 if the details communicated to the registration authority by the prospective new user 11 match with the details that have been entered on the user PC 12. Optionally the user can communicate the information to the registration authority electronically eg. by fax or modem or tone phone.

As a final stage in registration (refer Fig. 2c) the registration authority 16 provides the registration number generated by the registration authority PC 15 to the user

11. The user 11 enters the registration number into the user PC 12 where the registration routine checks to see whether the entered registration number matches the calculated registration number. If the two match then a valid registration has taken place and access is provided by the registration routine to a full operating version of the software protected by the registration routine. If there is no match and a preference file (which stores the user details) does not exist then a dialogue box D (Fig. 2c) appears on the display 13 of user PC 12 providing the prospective new user 11 with the opportunity to check his/her details or switch to the demonstration version of the software protected by the registration routine.

Again, the registration authority PC 15 can provide to PC 12 the registration number which it generates by electronic means such as modem communication.

It will be evident that it is not obvious to the prospective new user 11 that the registration number which unlocks the full version of the software protected by the registration routine is, in fact, generated from an algorithm residing on the magnetic disk 10 and that it forms part of the software to which access is desired.

In this manner the registration procedure outlined above ensures that exactly the same details entered by the prospective new user on his/her user PC 12 are those details recorded by the registration authority 16. It will also be evident that the procedure does not require each magnetic disk 10 containing a copy of the software to be

protected to have a unique registration number recorded on the disk at the time of distribution of the disk. Each copy has exactly the same registration number algorithm located upon it. A unique registration number or "security key" is generated only at the time of registration from the details supplied by the prospective new user 11.

The registration routine behaves generally as follows where any copy of the protected software boots. In this situation, the registration routine checks at the time of boot to see what registration details are present for that particular copy of the software. If no details are present then it is assumed that the PC is booting from a newly distributed magnetic disk and registration is to occur for the first time. The registration procedure in that case is that followed in respect of Figs. 2a, 2b and 2c.

In the event that registration details are present then the registration routine checks a number of parameters which are expected to be unique to the environment in which the software to be protected operates. In this embodiment the parameters checked are hard disc volume name, user name, and computer name and user password and hard disc initialisation date (not generally user configurable on the Apple Macintosh computer). The registration routine then checks these parameters against the corresponding details that it finds from the operating environment of the computer on which the software is running. If a designated combination of these details matches then it is assumed that a properly authorised and registered copy of the

software is running and full access to the software is allowed.

In this manner, it is quite in order for users to provide other users with copies of the software protected by the security routine. The security routine attached to the software to be protected determines from the environment in which it operates whether an additional registration fee is required. If it is determined by the registration routine that this is the case then the registration routine has the capability to provide a fresh registration number as part of an authorised registration procedure pending which the protected software reverts to demonstration mode.

2. SECOND EMBODIMENT (Auto re-registration)

According to a second embodiment a more sophisticated procedure suitable for checking at first boot and at subsequent boot is shown in flow chart form in Fig. 3.

This procedure incorporates redundancy to cope with situations where the key file containing the information from which the current use has been authorised may have been deleted or does not exist on a subsequent boot.

The distinction as against the first embodiment is that a "key file" is created at the time of registration of the software and a duplicate key file is also created at the same time. The duplicate key file is arranged to be stored on the computer at a location separate from the programme to be protected. In the case of the Apple Macintosh computer the duplicate key file can be stored in

the "system" folder.

Both the key file (stored with the software) and the duplicate key file are encrypted and both contain identical information. The information contained comprises:

1. The user registration details including the serial number,
2. The environment details of the computer, and
3. Details of the application protected by the security routine for which registration is to be or has been obtained.

With reference to Fig. 3 whenever the protected application boots a check is made by the registration routine to determine whether registration details exist in the key file of the protected application. If they do a comparison is made by the registration routine between what is stored in the key file and the environment to determine whether a change has taken place to the environment as compared with what is stored in the key file. If no change is detected then the protected application is permitted to run normally.

If there are no registration details present in the key file or if the above referenced comparison between the key file contents and the application does not show a match then the re-registration routine of Fig. 3 looks for the existence of a duplicate key file within the environment. If a duplicate key file exists then the information contained within that duplicate key file is copied to the application key file and comparisons as previously

described as between the key file details and the environment and application are made. If the comparison is positive then the protected application is allowed to run normally. If the comparison proves negative then the protected application is permitted to run by the registration routine in demonstration mode only. If a duplicate key file is found not to exist at all and the internal key file if present brings a negative result then the protected application is allowed to run in demonstration mode only.

This arrangement provides improved durability for the registration routine in the sense that it is less likely that the protected application will be caused to run in demonstration mode for incorrect reasons.

3. THIRD EMBODIMENT - TRACKING SYSTEM

With reference to Fig. 4 a modified form of the dialogue box C of Fig. 2b is shown which includes provision for entry of "your user number" in box 21.

At the time a prospective new user enters his/her details into the other boxes comprising the dialogue box C, there is an option for the user to enter a user number into box 21. The user number is provided by the registration authority 16 as a number unique to that particular registered user. If the box 21 has the user number details inserted into it then the registration routine, when the next copy of the protected application is made, will transfer the user number details from box 21 to the "last user number" box 22. A similar transfer will take place

when next a copy is made of the protected application if and only if the person wishing to register the next copy enters their user number details in box 21. If they do not, then the last user number details in box 22 remain as before. In this manner a tracking system is available to the registration authority in the form of a tree where any given copy is identified by its ancestry based on current and previous user number as entered into boxes 21 and 22.

4. SELF SERIALISATION

In a particular embodiment a process termed "self serialisation" can be utilized to produce the serial number 50 which is displayable to the user/licensee as illustrated in Fig. 4.

The serial number 50 is disguised by use of a random or pseudo random number input to the algorithm which generates the serial number at the time of first boot of the software as part of the initial registration procedure. For example the serial number, when generated by the self serialisation process, can be generated by a random number routine forming part of the registration software or it can be generated by the registration software with reference to data which is available in a widely varying fashion on the platform on which the software is located - for example a time reference on the platform. The serial number 50 generated by the self serialisation process can be a required input to the registration algorithm from which the registration number is generated. Clearly the serial number 50 as determined and displayed to the user will then

be required to be communicated to the registration authority for input to the registration authority's registration number generating algorithm.

It will be observed that a serial number 50 generated in this manner is likely to be displayed as a different number on each platform on which the software to be protected is to be run and comprises a randomised input to the registration algorithm which is determined and determinable only at the time of registration.

5. FIFTH EMBODIMENT

With reference to Fig. 5 there is shown in schematic form a microprocessor 30 adapted to operate under an operating system or upon a platform 31 such as, for example, Microsoft DOS or Macintosh System 7. The platform 31 allows relatively high level commands to be used to cause the microprocessor 30 to interact with input/output devices such as keyboard 32, monitor 33, loudspeaker 34, memory 35 and magnetic or CD ROM disc 36.

By way of example a word processing programme comprising a length of code or digital data 37 has been copied onto disc 36.

The digital data 37 includes registration code portion 38 and use code portion 39.

The digital data 37 is arranged in such a way that when microprocessor 30 seeks to first execute the digital data 37 by way of operating system or platform 31 the digital data comprising the registration code portion 38 is caused to execute first in a manner previously described in

reference to the first embodiment of the invention. The execution of the digital data comprising the registration code portion 38 in conjunction with the operating system or platform 31 comprises a mode switcher which will permit the microprocessor 30 to execute the use code portion 39 of digital data 37 only in a demonstration mode unless and until registration involving reference to an external registration authority is first completed successfully. This registration procedure is as previously described with reference to the first embodiment.

The digital data 37 can comprise, for example, a word processing programme such as Wordperfect 5.1 available from Wordperfect Corporation. The registration code portion 38 is integral with the digital data 37 comprising the word processing programme. The registration code portion 38 includes the algorithm for calculation of the registration number as previously described in respect of other embodiments of the invention.

It will be appreciated that the registration code portion 38 effectively forms simply a part of the software or digital data 37 to be protected/registered and that the digital data 37 will be or can be identical for all copies of the word processing programme produced. The registration code portion 38 allows a unique link to be made between the digital data 37 and an individual authorised or licensed to use the digital data 37 by way of initial execution of a copy of the digital data comprising registration code portion 38.

With reference to Figs. 6 and 7 a specific realisation of the fifth embodiment will be described.

With particular reference to Fig. 6 a decoder 51 is interposed in the data path from the CD in CD player 52 and a digital to analogue converter 53. The digital to analogue converter 53 is the device by which digitally encoded musical or video information residing on CD ROM 54 is converted to analogue form suitable for playback on current mass produced television sets (video) or hi-fi sets (audio).

The decoder 51 comprises part of the platform upon which the digital data 37 is executed and includes means to interpret the code portion 38 of the digital data 37 whereby the registration system is implemented such that the digital data 37 and, more particularly, the use code portion 39 of that digital data 37 can be executed on the platform in a use mode only if the registration procedure to which reference has been made in respect of previous embodiments has been performed.

The registration code portion 38 can include a preview or demonstration related to a subset of the balance of the digital data on the CD 54 which can be executed by the platform without license.

The decoder 51 includes LCD display 55 and keypad 56 whereby the licensee can enter information via keypad 56 and receive information via the LCD display 55 for the purpose of the registration procedure.

In addition a smart card (SRAM) 57 is receivable by

the decoder 51 for the purpose of customising or amending operation of the decoder 51.

With reference to Fig. 7 the registration procedure following insertion of CD 54 into CD player 52 is as follows. The user operates the play control and decoder 51 reads from CD 54 code portion 38 of digital data 37 located thereon and executes this code so as to determine whether the digital data is already licensed for the platform. If not, a demonstration is communicated via digital to analogue converter 53 whilst the user determines whether to register as a licensee of the digital data 37 in the manner indicated in the flow chart of Fig. 7.

6. SIXTH EMBODIMENT

With reference to Fig. 8 there is shown a block diagram of a system according to a further embodiment of the invention which is to be read in the context of the earlier generalised description in respect of Fig. 1.

The system illustrated in Fig. 8 operates in the manner generally described in respect of previous embodiments and as generally outlined in the diagram. In the context of the block C illustrated in Fig. 4 and with reference to Fig. 9 the algorithm which generates the unique user identification and which is resident both as the registration code portion 38 in digital data 37 integrally bound to use code portion 39 for execution on local platform 31 and also as remote algorithm 61 attached to registration database programme 62 for execution on the remote platform 63.

The algorithm, in this embodiment, combines by addition the serial number 50 with the software product name 64 and customer information 65 and previous user identification 22 to provide registration number 66.

As discussed earlier all of the items to be summed, namely items 50, 64, 65 and 22 must be communicated to the remote licensee unique ID generator 67 by the intending licensee whereby algorithm 61 causes the production of a registration number 66 which matches identically with the locally produced registration number. When mode switcher 68 verifies the match then the mode switcher 68 allows execution on platform 31 of the full user programme 39.

Prior to allowing execution of the full programme mode switcher 68 will also check whether platform ID 69 has changed as provided to it by platform unique ID generator 70.

In this embodiment serial number 50 is comprised of two components, namely system information 71 and a variable key portion 72. The variable key portion 72 provides the characteristic of self serialisation described earlier in the specification and, in this embodiment, is generated at the time of registration on platform 31 by reference to a variable platform parameter, in this case reference to system time information although other parameters which are variable can be utilised in other embodiments.

System information 71 can include information which identifies the hardware comprising the platform 31 on which the user programme 39 is to be executed such as, for

example, CPU number (where available), or unique parameters associated with the firmware in use. The system information, optionally, can further include system configuration information such as amount of memory, type of processor etc.

It will be noted, therefore, that serial number 50 will appear to an intending licensee when it appears on screen as per box C in Fig. 4 as an apparently random variable having no obvious link to the platform 31 or the user programme 39.

However, when the serial number 50 is communicated to the remote licensee unique ID generator 67 a secondary algorithm complementary to the algorithm which generated the serial number including variable key portion 72 and system information 71 is able to "decode" or otherwise strip away the variable key portion 72 so as to make use of the system information 71 if allowable and desirable in the circumstances.

Whether the system information 71 is utilised or not the serial number 50 generated in this manner provides an input to the algorithm which generates registration number 66 which presents as an apparently variable parameter thereby rendering "cracking" of the software registration system more difficult and unlikely.

7. SEVENTH EMBODIMENT

The schematic diagram of Fig. 10 illustrates a substantially hardware implementation of the invention applicable, for example, for implementation of the CD

arrangement of Fig. 6 or the more generalised arrangement of Figs. 8 and 9.

In this embodiment a prospective user 80 of digital code 81 on media 82 by its execution on platform 83 firstly inserts the media 82 into an appropriate digital code reading device within platform 83 (eg a floppy disk drive or a CD ROM drive).

Customer information C is provided by user 80 both direct to local encoder/decoder 84 and also to local adder or summer 85.

Additionally product information P derived from media 82 (typically via platform 83) or else via the intermediary of the user (signified by the small man symbol) is provided to encoder/decoder 84 and to summer 85.

Finally, a serial number S derived from platform 83 is supplied either directly or via the intermediary of user 80 to encoder/decoder 84 and to summer 85.

Summer 85 acts as a local licensee unique ID generating means by combining, by addition, customer information C, product information P and serial number S in order to provide a local licensee unique ID here designated Y.

Encoder/decoder 84 transmits the serial number S, the customer information C and the product information P via modems 86, 87 over the public switched telephone network to a remote encoder/decoder 88 which, in turn, supplies signals S, C and P to the inputs of remote summer 89. Remote summer 89 combines these signals by addition

(thereby acting as a remote licensee unique ID generating means) so as to provide a summed output here termed X which represents a licensee unique ID or enabling key which should match identically with the local licensee unique ID or registration key or registration number Y if inputs S, C and P to summers 85 and 89 are identical.

The licensee unique ID termed X is transmitted back via encoder/decoders and modems 84, 86, 87, 88 to comparator 90 which outputs a high signal if X equals Y. This condition corresponds to the local licensee unique ID matching with the licensee unique ID generated at the remote location by the remote licensee unique ID generating means generally comprising summer 89.

Digital code 81 on media 82 comprises code identified as a demonstration portion D together with code identified as a use portion U. There may be other kinds of code designated O as well.

Code 81 is executed on platform 83 (for example a microprocessor or a substantially hardware based, dedicated playback device such as a CD drive with the code being passed through a mode switcher comprising first gate 91 and second gate 92 together with relay 93.

First gate 91 energises relay 93 so as to permit execution of code of type D but not code of any other type such as of type U.

Second gate 92 permits execution of any kind of code by closure of relay 93 provided only that the output of comparator 90 is high (which is to say that X equals Y or

that the local licensee unique ID matches with the licensee unique ID generated by the remote licensee unique ID generating means comprising summer 89).

Comparator 90 together with gates 91, 92 and relay 93 comprise one particular form of mode switcher or switching means suitable for recognizing and allowing execution on platform 83 of various kinds of code such as the code of types D and U.

INDUSTRIAL APPLICABILITY

The aforementioned may be applied either in dedicated electronic hardware or by means of more generalised digital computation devices such as microprocessors and the like in order to regulate use of digital code.

The above describes only some embodiments of the present invention and modifications, obvious to those skilled in the art, can be made thereto without departing from the scope and spirit of the present invention.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:-

1. A registration system for licensing execution of digital data in the use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID first generated by said local licensee unique ID generating means has matched a licensee unique ID subsequently generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

2. The system of claim 1, wherein said local licensee unique ID generating means generates said local licensee unique ID by execution of a registration algorithm which combines information in accordance with said algorithm, said information uniquely descriptive of an intending licensee of said digital data to be executed in said use mode.

3. The system of claim 2, wherein said mode switching means permits operation of said digital data in said use mode in subsequent execution of said digital data only if said licensee unique ID generated by said local licensee unique ID generating means has not changed.

AUSTRALIAN PATENT OFFICE

4. The system of claim 3, wherein said local licensee unique ID generating means comprises part of said digital data when executed on said platform.

5. The system of claim 4, wherein said mode switching means comprises part of said digital data when executed on said platform.

6. The system of claim 5, wherein the information utilized by said local licensee unique ID generating means to produce said licensee unique ID comprises prospective licensee details including at least one of payment details, contact details and name.

7. The system of claim 1, said system further including platform unique ID generating means, wherein said mode switching means will permit said digital data to run in said use mode in subsequent execution of said digital data on said platform only if said platform unique ID has not changed.

8. The system of claim 7, wherein said platform unique ID generating means comprises part of said digital data when executed on said platform.

9. The system of claim 8, wherein said platform unique ID generating means utilizes hard disc or other platform information to determine said platform unique ID.

10. The system of claim 1, wherein said platform comprises a computer operating system environment.

AUSTRALIAN PATENT OFFICE

11. The system of claim 10, wherein said digital data comprises a software program adapted to run under said operating system environment.

12. A registration system attachable to software to be protected, said registration system generating a security key from information input to said software which uniquely identifies an intended registered user of said software on a computer on which said software is to be installed; and wherein said registration system is replicated at a registration authority and used for the purposes of checking by the registration authority that the information unique to the user is correctly entered at the time that the security key is generated by the registration system.

13. The registration system of claim 12, wherein said security key is generated by a registration number algorithm.

14. The registration system of claim 13, wherein said registration number algorithm combines information entered by a prospective registered user unique to that user with a serial number generated from information provided by the environment in which the software to be protected is to run.

15. The registration system of claim 12, wherein said registration system checks at the time of boot of said software as to whether it is a first boot of the software to be protected or a subsequent boot, and, if a subsequent boot is detected, then environment and user details are compared to determine whether the program reverts to a demonstration mode and a new user registration procedure is to commence or a full version run.

AUSTRALIAN PATENT OFFICE

16. The registration system of claim 15, wherein said environment details comprise at least one element which is not user-configurable on the platform.

17. A method of control of distribution of software, said method comprising providing mode-switching means associated with said software adapted to switch said software between a fully enabled mode and a partly enabled or demonstration mode, said method further comprising providing registration key generating means adapted to generate a registration key which is a function of information unique to an intending user of the software; said mode-switching means switching said software into fully enabled mode only if an enabling key provided to said mode-switching means by said intending user at the time of registration of said software has matched identically with said registration key; and wherein said enabling key is communicated to said intending user at the time of registration of said software; said enabling key generated by a third party means of operation of a duplicate copy of said registration key generating means.

18. The method of claim 17, wherein said registration key is also a function of the environment in which said software is installed.

19. A remote registration station incorporating remote licensee unique ID generating means, said station forming part of a registration system for licensing execution of digital data in a use mode, said digital data executable on a platform, said system including local licensee unique ID

AUSTRALIAN PATENT OFFICE

generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only of a licensee unique ID generated by said local licensee unique ID generating means has matched a licensee unique ID generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

20. A method of registration of digital data so as to enable execution of said digital data in a use mode, said method comprising an intending licensee operating a registration system for licensing execution of digital data in a use mode, said digital data executable on a platform, said system including local licensee unique ID generating means and remote licensee unique ID generating means, said system further including mode switching means operable on said platform which permits use of said digital data in said use mode on said platform only if a licensee unique ID generated by said local licensee unique ID generating means has matched a licensee unique ID generated by said remote licensee unique ID generating means; and wherein said remote licensee unique ID generating means comprises software executed on a platform which includes the algorithm utilized by said local licensee unique ID generating means to produce said licensee unique ID.

AUSTRALIAN PATENT OFFICE

21. A registration system for licensing execution of digital data in the use mode substantially as hereinbefore described with reference to the accompanying drawings.

DATED this 3rd day of January, 1997.
UNILOC CORPORATION PTY LIMITED
by their Patent Attorney
PETER MAXWELL & ASSOCIATES.

AUSTRALIAN PATENT OFFICE

48113/93

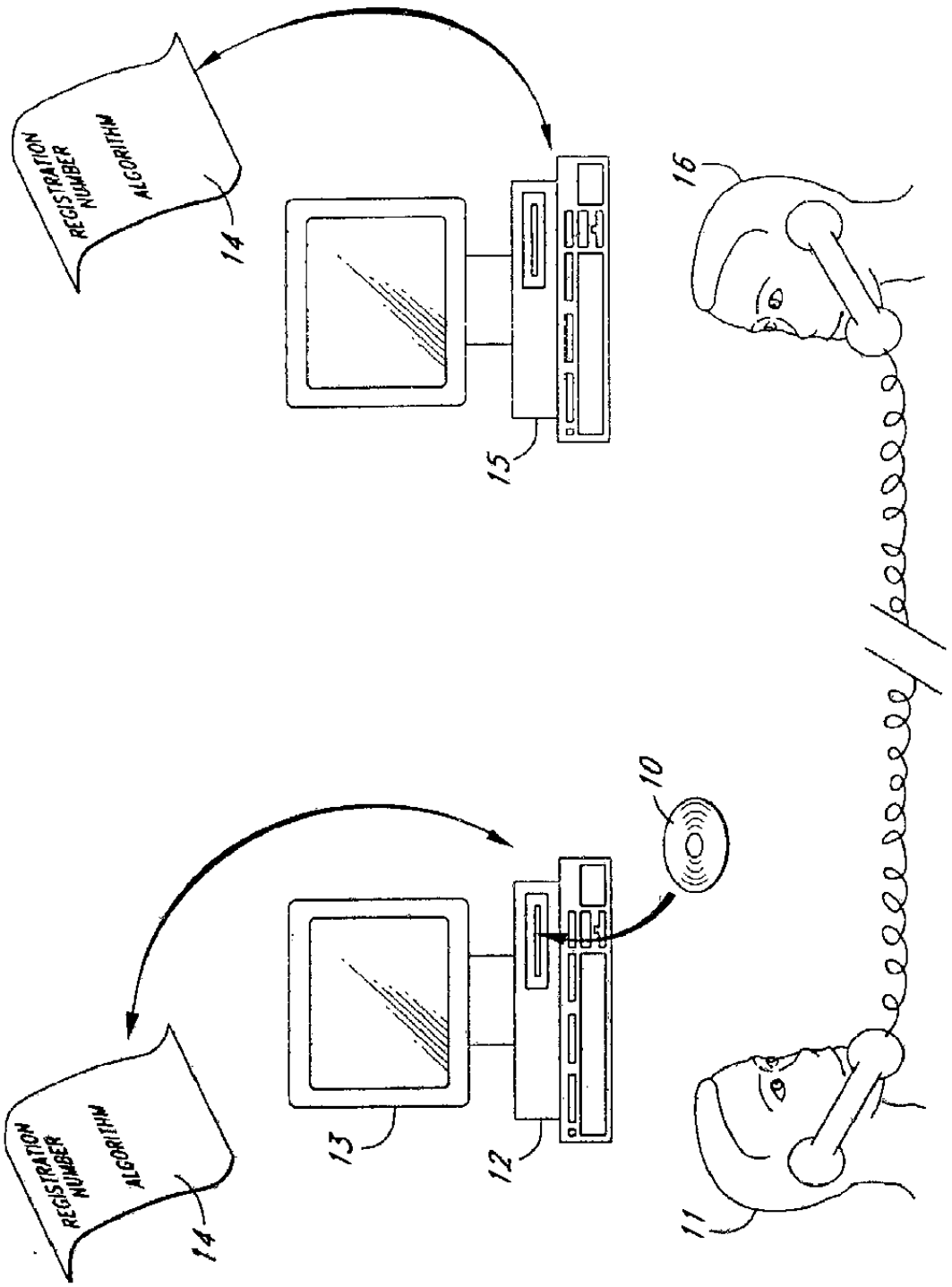


FIG. 1

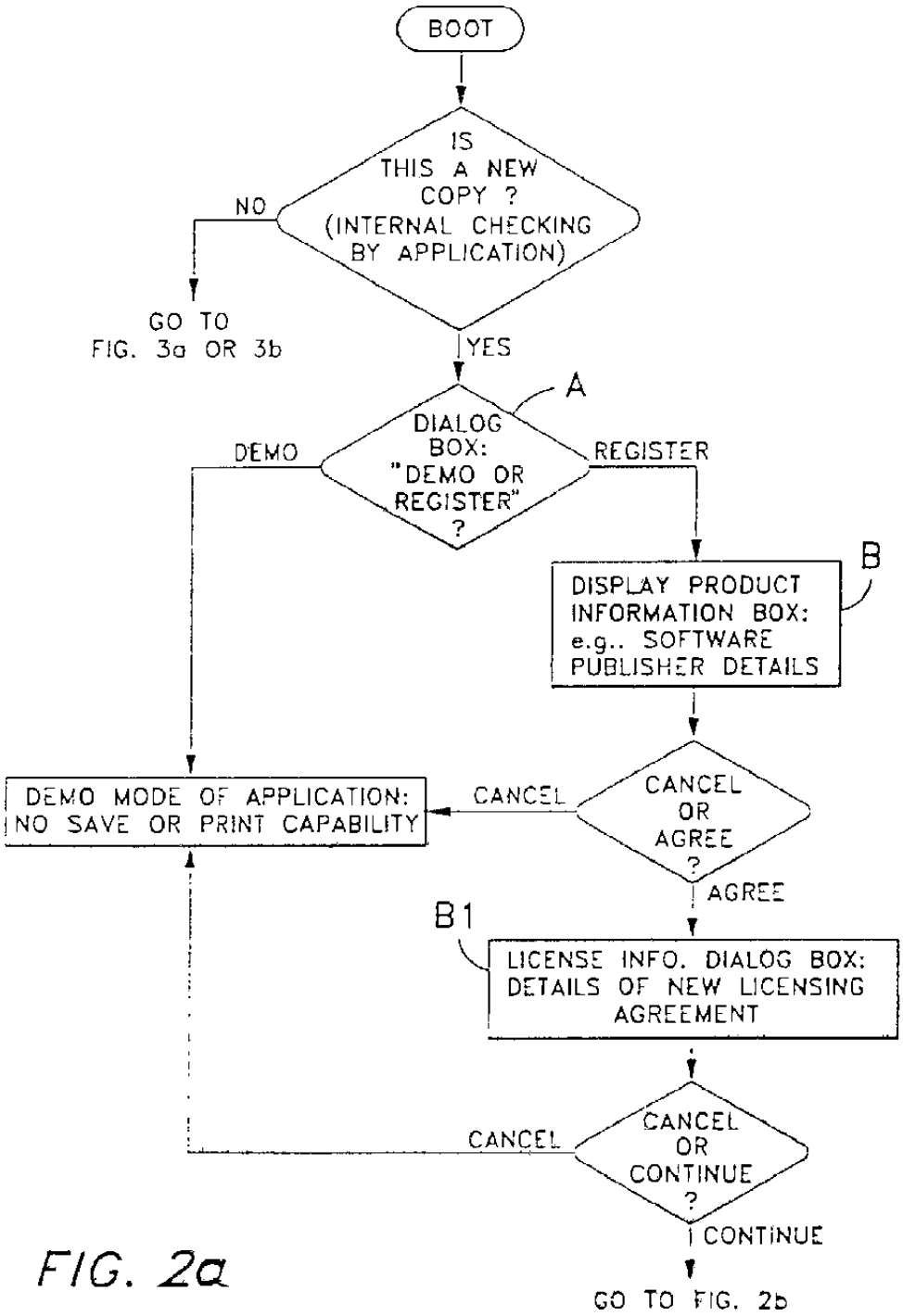
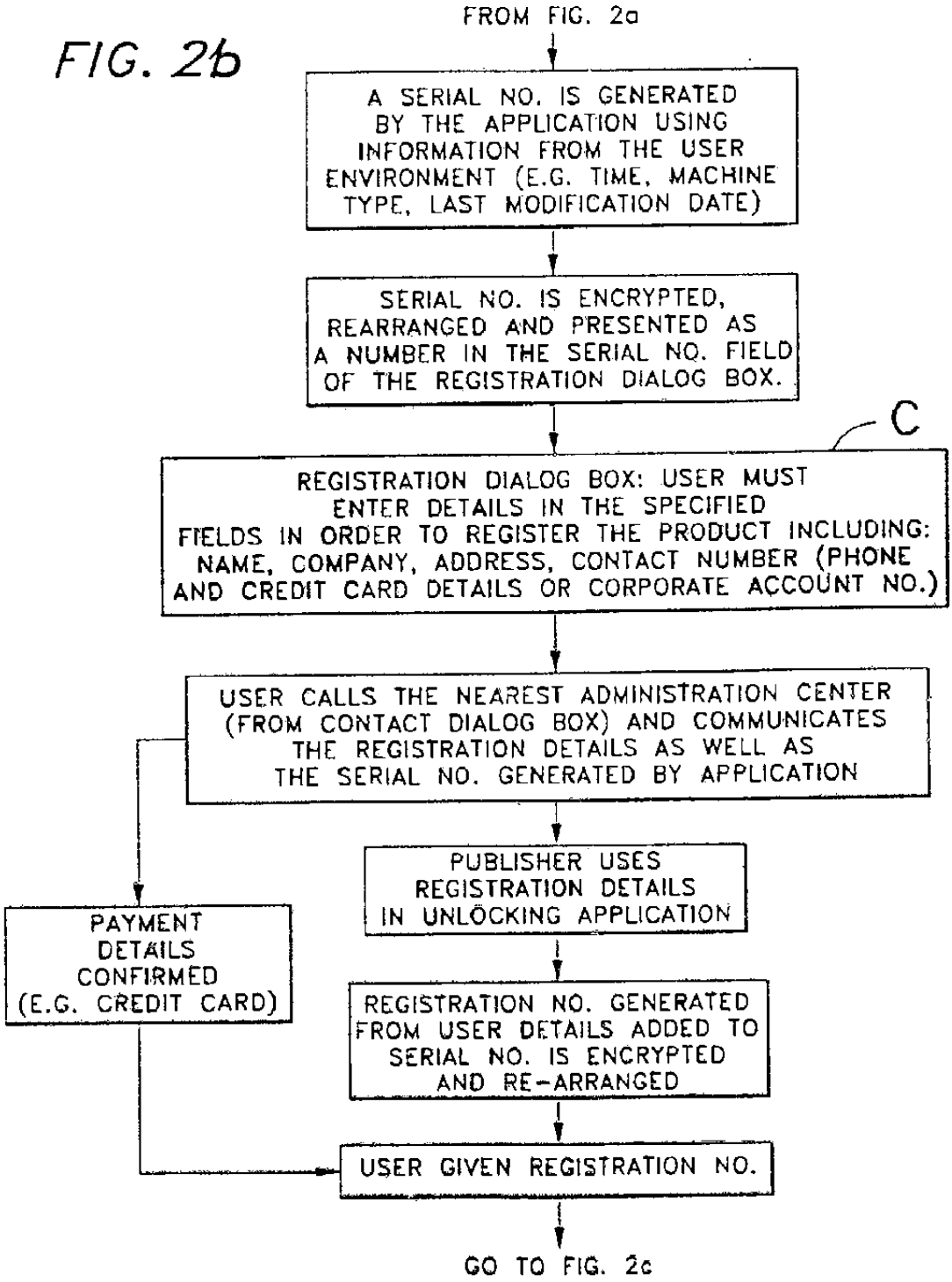

BOOT
IS THIS A NEW COPY ? (INTERNAL CHECKING BY APPLICATION)
NO
GO TO FIG. 3a OR 3b
YES
A
DIALOG BOX: "DEMO OR REGISTER" ?
DEMO
REGISTER
B
DISPLAY PRODUCT INFORMATION BOX: e.g.. SOFTWARE PUBLISHER DETAILS
CANCEL OR AGREE ?
CANCEL
DEMO MODE OF APPLICATION: NO SAVE OR PRINT CAPABILITY
AGREE
B1
LICENSE INFO. DIALOG BOX: DETAILS OF NEW LICENSING AGREEMENT
CANCEL OR CONTINUE ?
CANCEL
CONTINUE
GO TO FIG. 2b

FIG. 2a

FIG. 2b



SUBSTITUTE SHEET

FIG. 2c

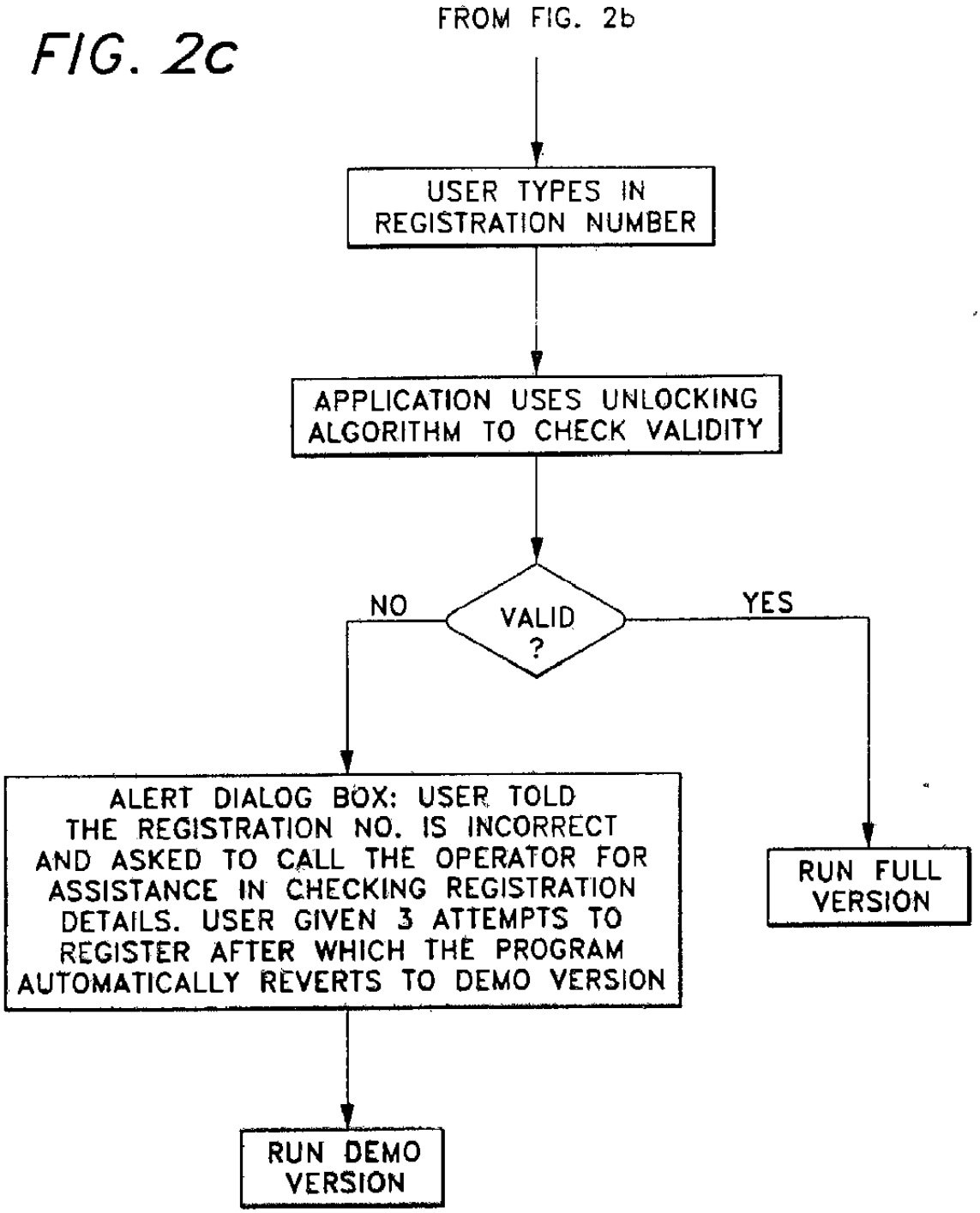
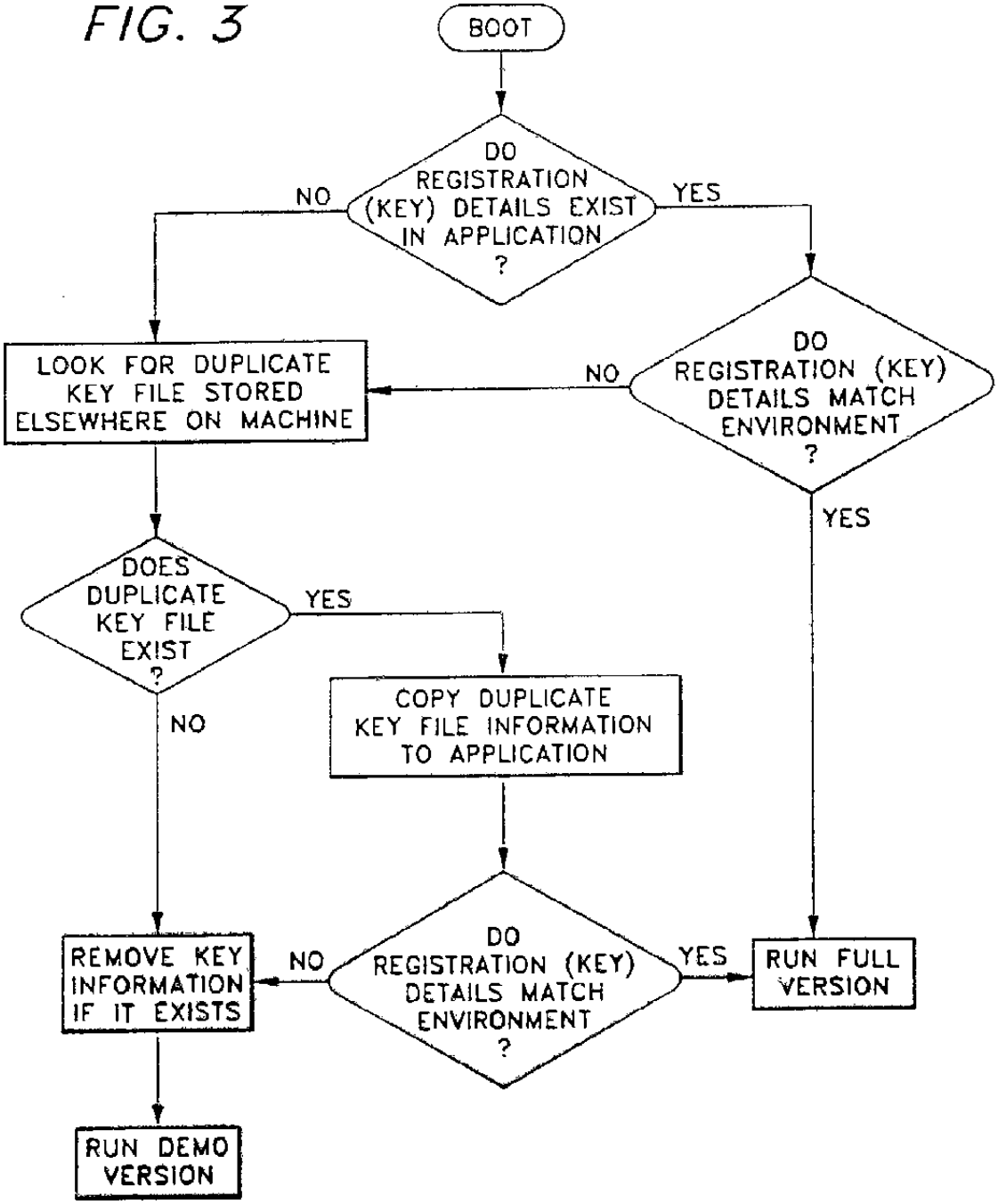

FROM FIG. 2b
USER TYPES IN REGISTRATION NUMBER
APPLICATION USES UNLOCKING ALGORITHM TO CHECK VALIDITY
VALID ?
NO
YES
ALERT DIALOG BOX: USER TOLD THE REGISTRATION NO. IS INCORRECT AND ASKED TO CALL THE OPERATOR FOR ASSISTANCE IN CHECKING REGISTRATION DETAILS. USER GIVEN 3 ATTEMPTS TO REGISTER AFTER WHICH THE PROGRAM AUTOMATICALLY REVERTS TO DEMO VERSION
RUN FULL VERSION
RUN DEMO VERSION

FIG. 3



SUBSTITUTE SHEET

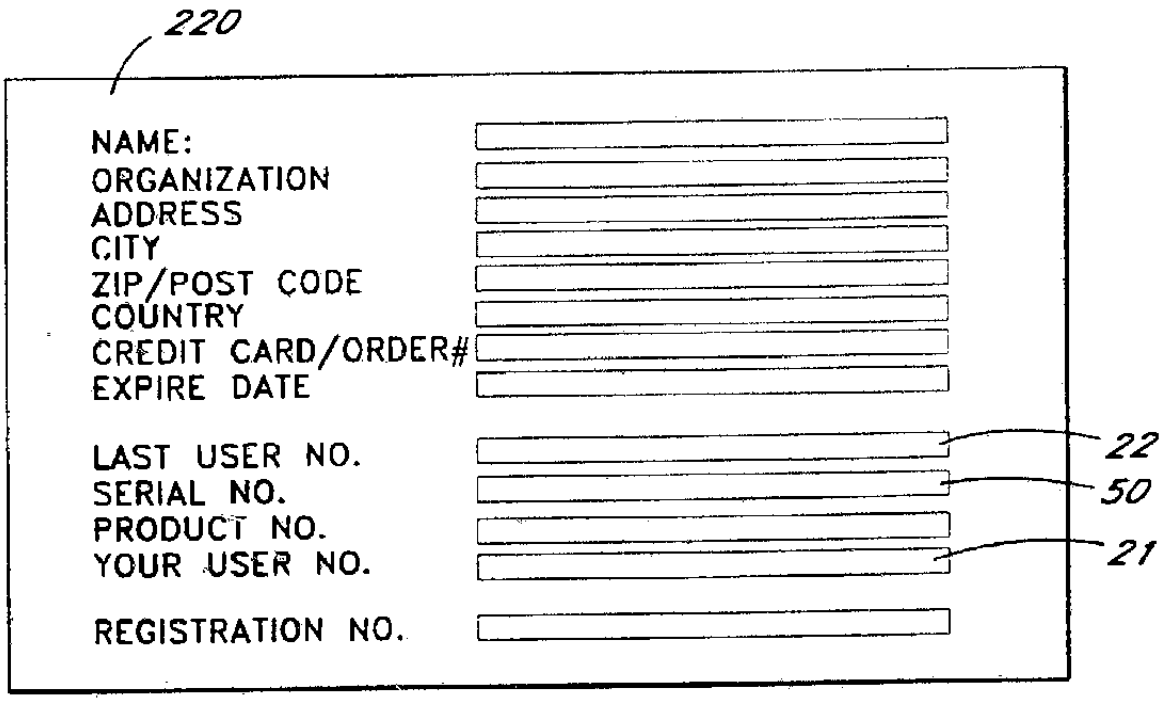

220
NAME:
ORGANIZATION
ADDRESS
CITY
ZIP/POST CODE
COUNTRY
CREDIT CARD/ORDER#
EXPIRE DATE
LAST USER NO.
SERIAL NO.
PRODUCT NO.
YOUR USER NO.
REGISTRATION NO.
22
50
21

FIG. 4

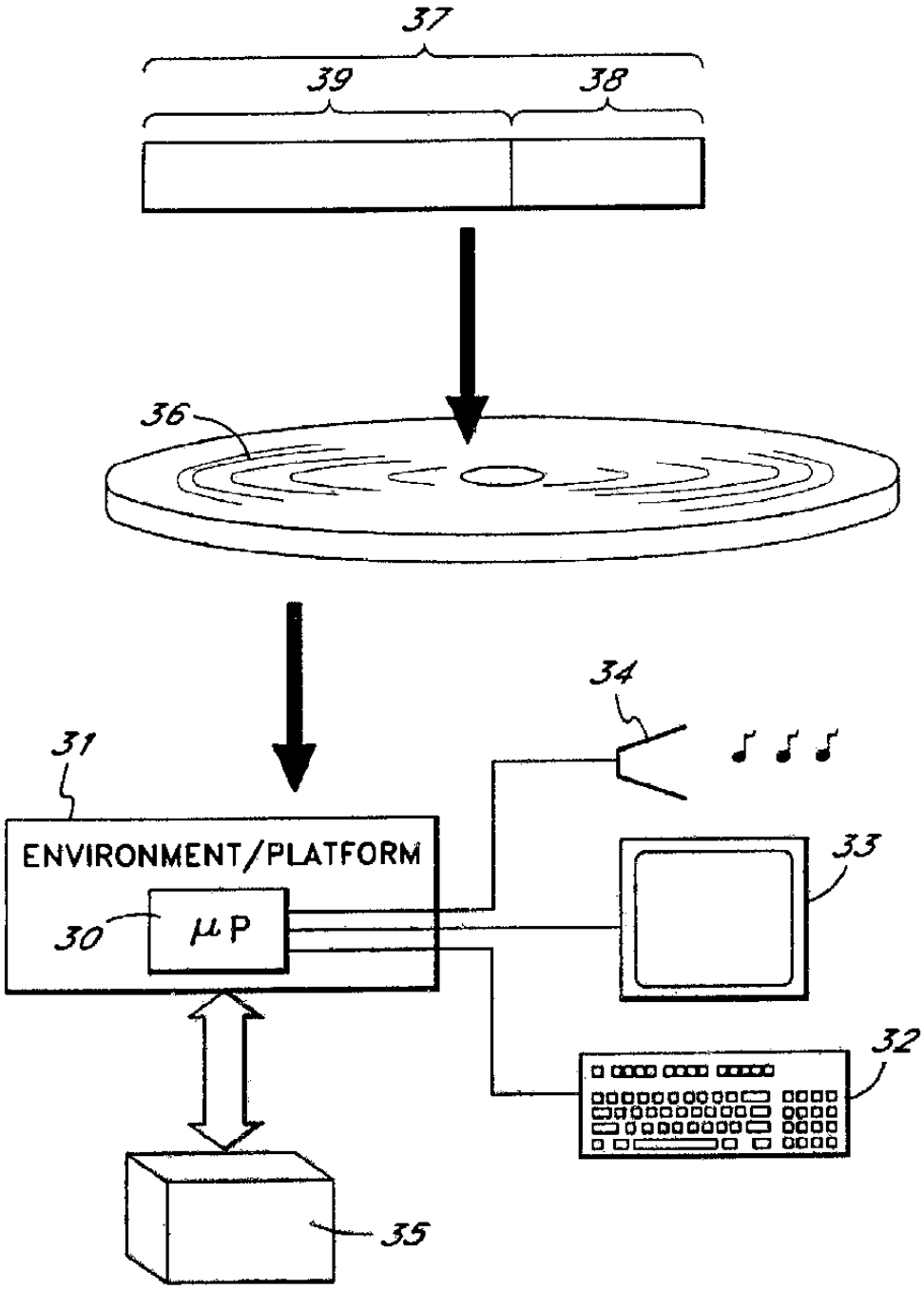


FIG. 5

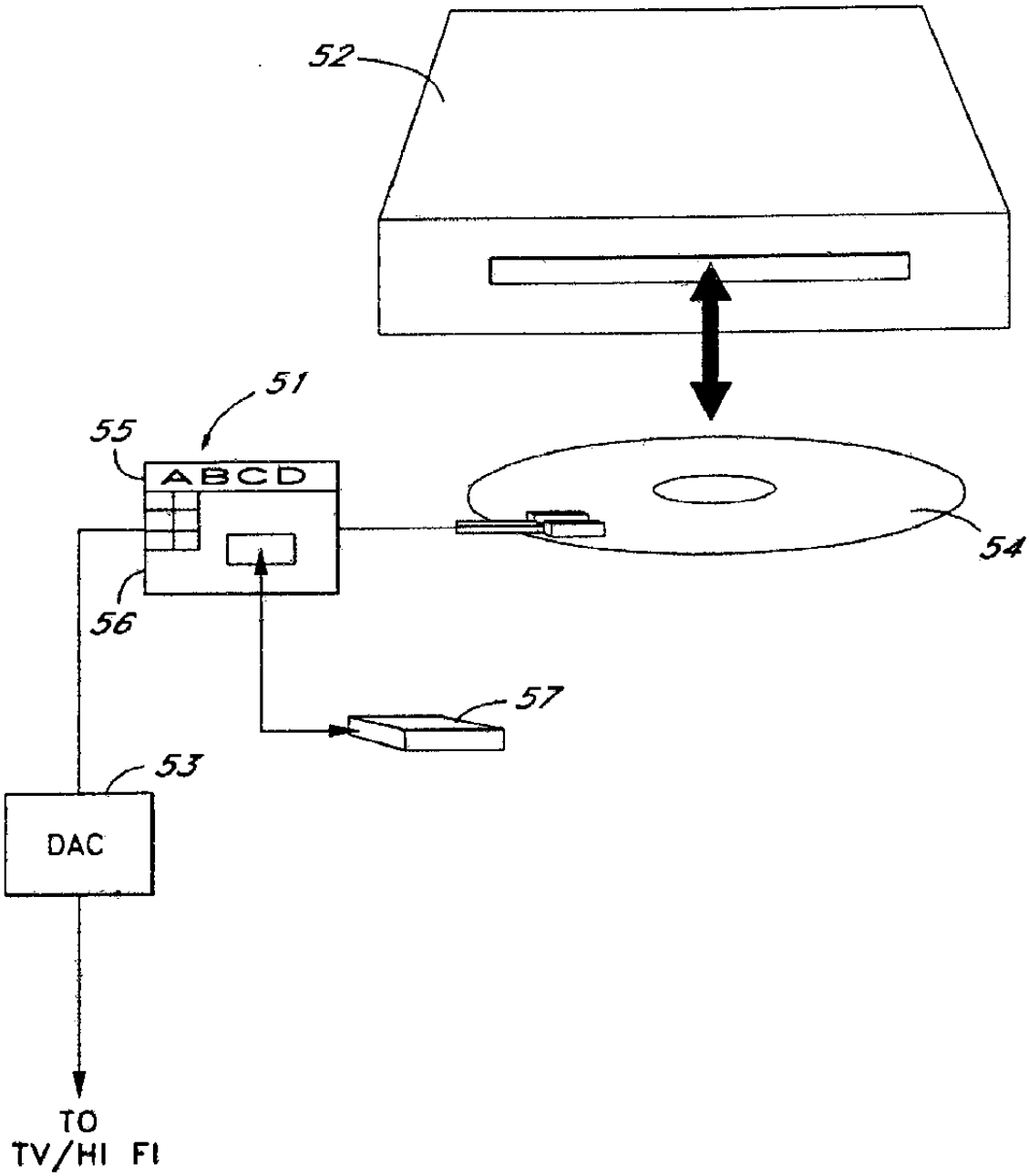


FIG. 6

SUBSTITUTE SHEET

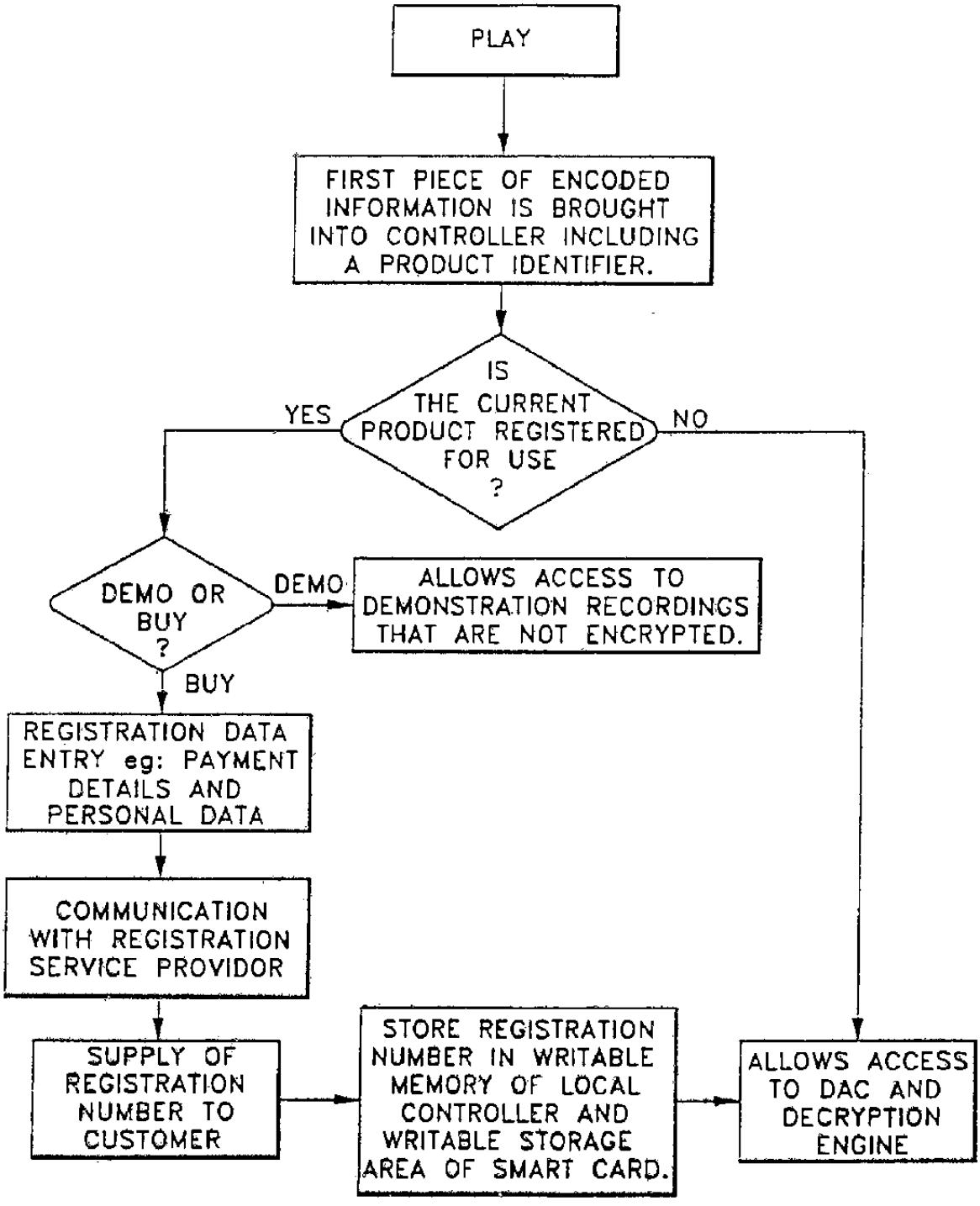


FIG. 7

SUBSTITUTE SHEET

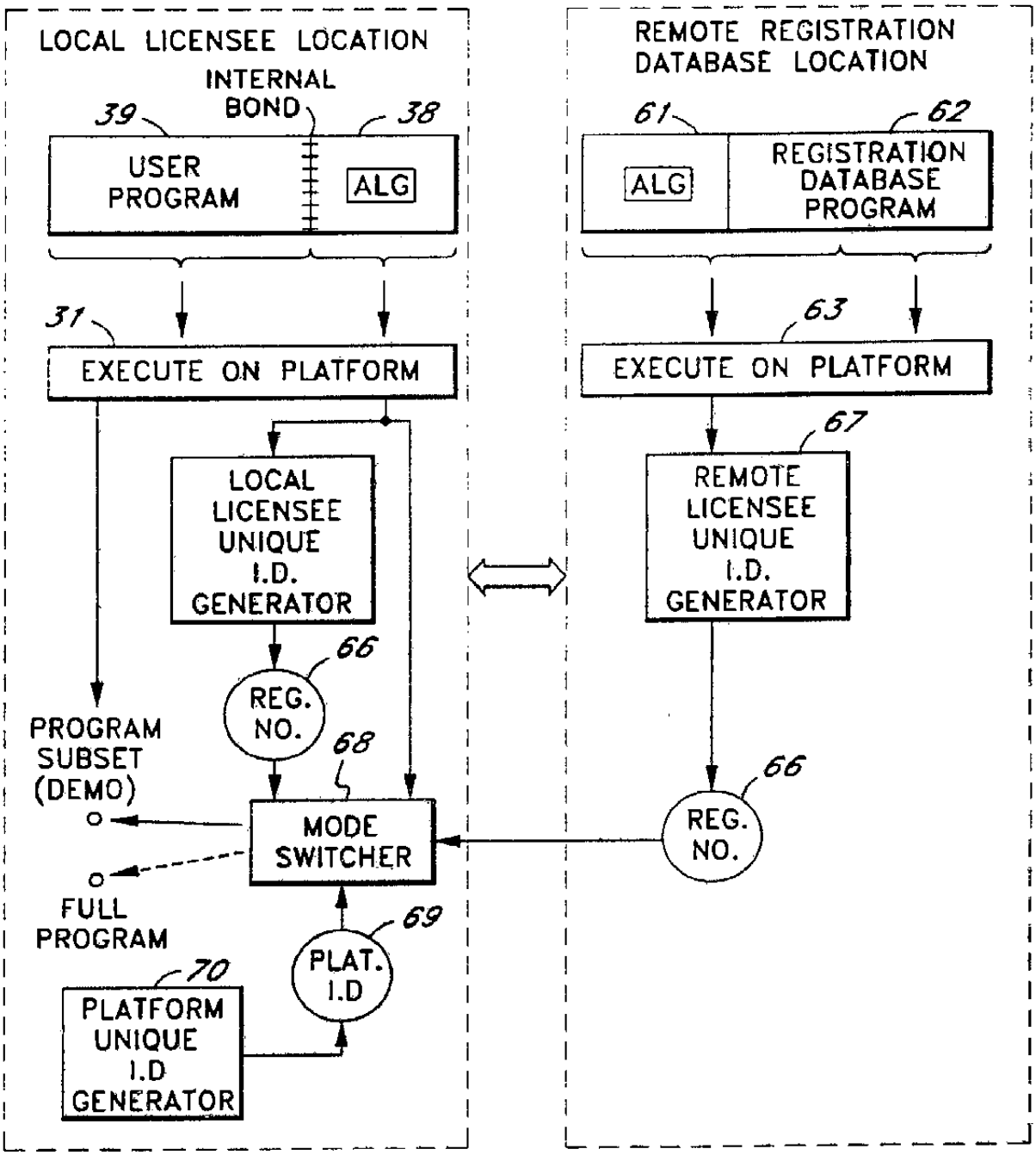


FIG. 8

SUBSTITUTE SHEET

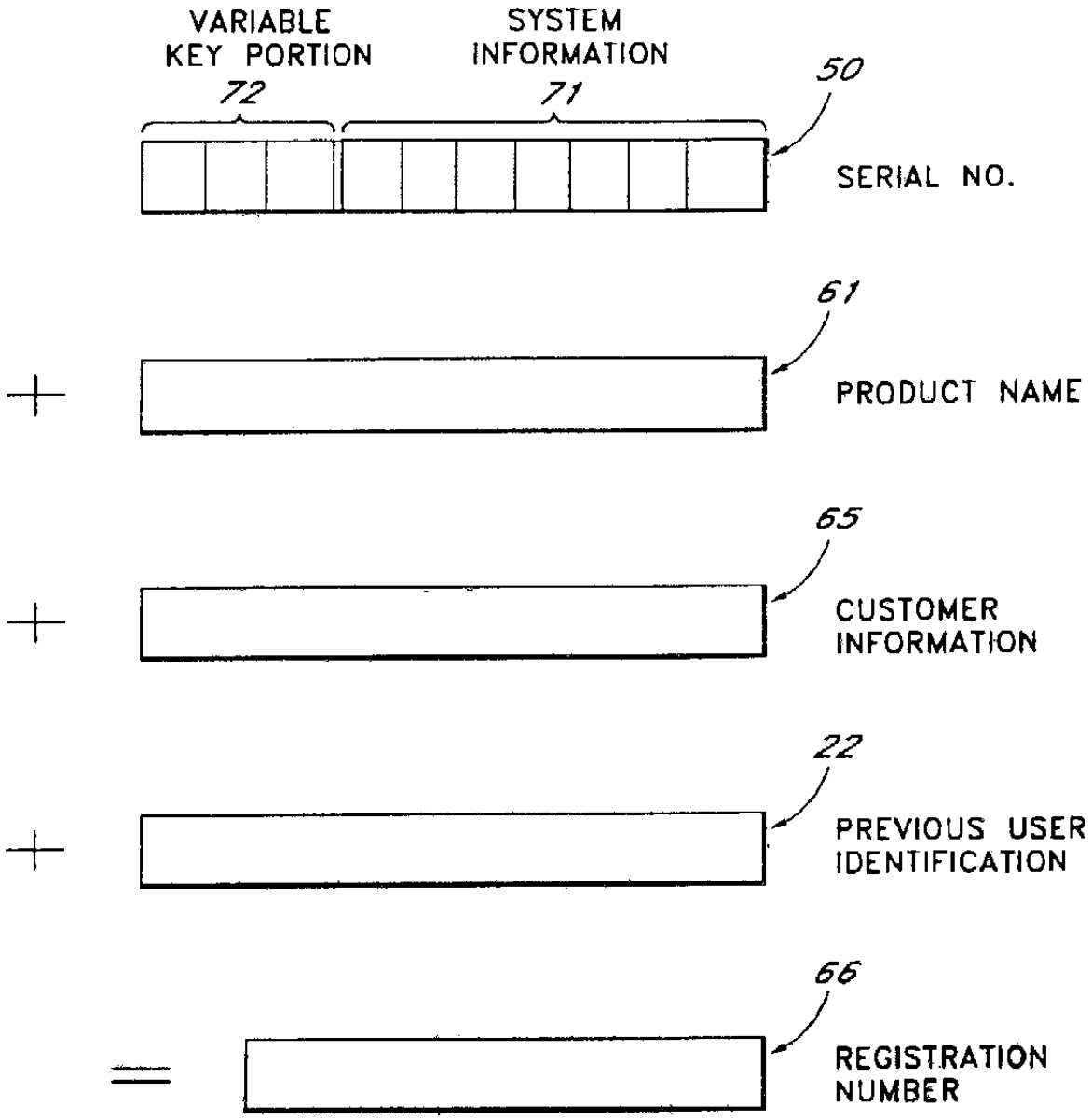

VARIABLE KEY PORTION
72
SYSTEM INFORMATION
71
50
SERIAL NO.
61
PRODUCT NAME
65
CUSTOMER INFORMATION
22
PREVIOUS USER IDENTIFICATION
66
REGISTRATION NUMBER

FIG. 9

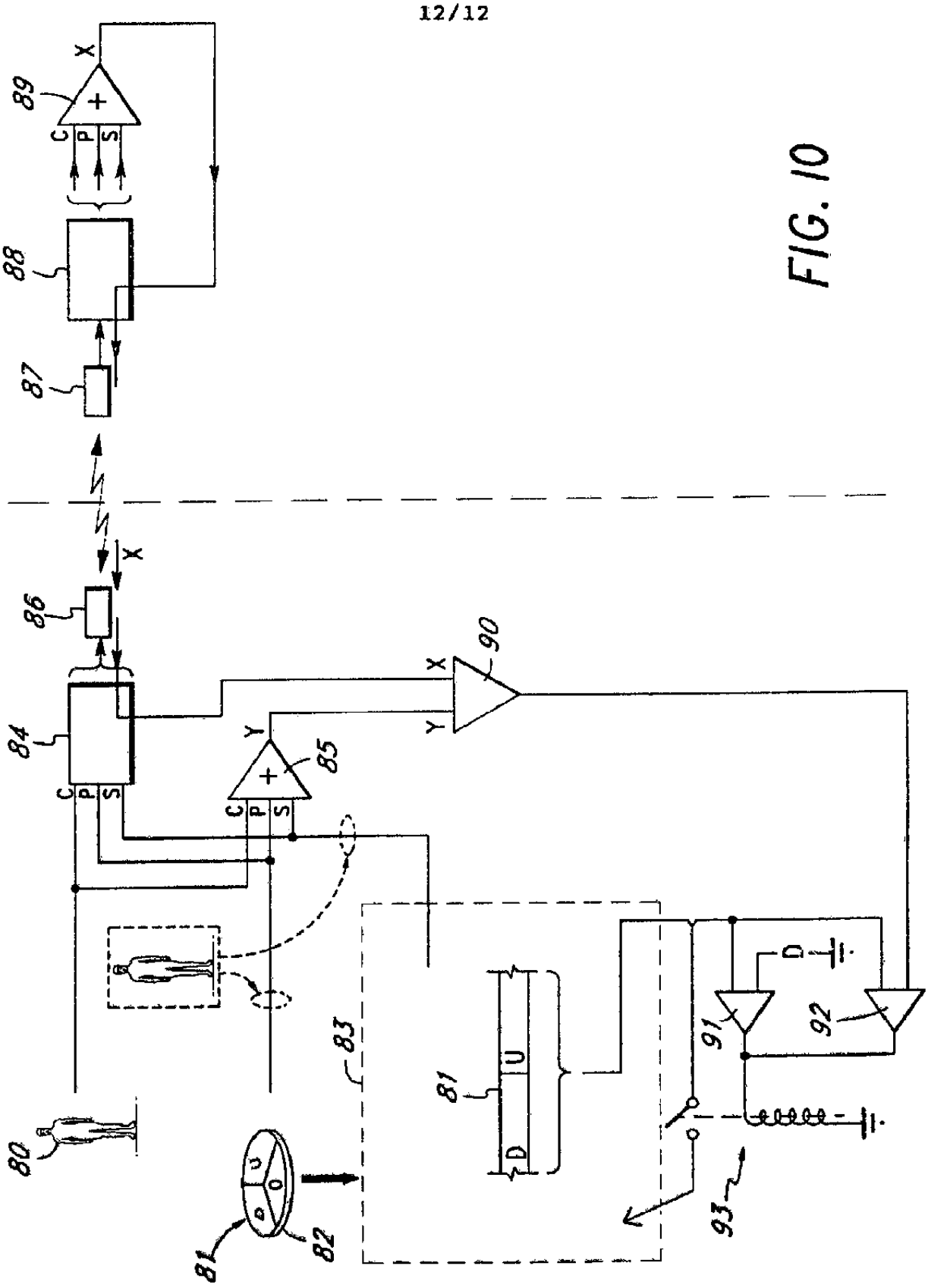


FIG. 10

SUBSTITUTE SHEET

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU 93/00483

A. CLASSIFICATION OF SUBJECT MATTER
Int. Cl.[5] G06F 15/21, 9/44

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC : G06F 15/21, 9/44

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
AU : IPC as above

Electronic data base consulted during the international search (name of data base, and where practicable, search terms used)
Derwent : (REGIST: or LICEN: or PROTECT: or SECUR: or VALID: or AUTHORIS: or VERIF:)
Japio : as above and (SOFTWARE: or PROGRA:)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
A	WO,A, 92/09160 (TAU SYSTEMS CORPORATION) 29 May 1992 (29.05.92) See whole document	
A	US,A, 4982430 (FREZIA et al) 1 January 1991 (01.01.91) See whole document	
A	US,A, 4796220 (WOLFE) 3 January 1989 (03.01.89) See whole document	
A	US,A, 4688169 (JOSHI) 18 August 1987 (18.08.87) See whole document	

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 December 1993 (16.12.93)	30 DEC 1993 (30.12.93)
Name and mailing address of the ISA/AU AUSTRALIAN INDUSTRIAL PROPERTY ORGANISATION PO BOX 200 WODEN ACT 2606 AUSTRALIA Facsimile No. 06 2853929	Authorized officer M. EASTHOPE Telephone No. (06) 2832212

Form PCT/ISA/210 (continuation of first sheet (2)) (July 1992) copjne

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate of the relevant passages	Relevant to Claim No.
A	US,A, 4654799 (OGAKI et al) 31 March 1987 (31.03.87) See whole document	

INTERNATIONAL SEARCH REPORT

International application No.
PCT/AU 93/00483

Box I Observations where certain claims were found unsearchable (Continuation of Item 1 of first sheet)

This international search report has not established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. [X] Claims Nos.: 22-24
because they relate to subject matter not required to be searched by this Authority, namely:

Mere presentation of information

2. [ ] Claim Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. [ ] Claims Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box II Observations where unity of invention is lacking (Continuation of item 2 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

1. [ ] As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims

2. [ ] As all searchable claims could be searched without effort justifying an additional fee, this Authority did not invite payment of any additional fee.

3. [ ] As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically

claims Nos.:

4. [ ] No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos.:

Remark on Protest

[ ] The additional search fees were accompanied by the applicant's protest.

[ ] No protest accompanied the payment of additional search fees.

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report	Patent Family Member
WO 9209160	CA 2095723, US 5222134, EP 556305, US 5103476
US 4982430	CA 1295412, JP 61248636, DE 3679711, EP 200704
US 4796220	
US 4688169	
US 4654799	
	END OF ANNEX

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 637 958 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
22.03.2006 Bulletin 2006/12

(51) Int Cl.:
G06F 1/00 *(2006.01)*

(21) Application number: **05021320.6**

(22) Date of filing: **02.03.2004**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IT LI LU MC NL PL PT RO SE SI SK TR

(30) Priority: **03.03.2003 US 378294**

(62) Document number(s) of the earlier application(s) in accordance with Art. 76 EPC:
04004848.0 / 1 455 258

(71) Applicant: **Microsoft Corporation**
Redmond WA 98052 (US)

(72) Inventors:
- **Gunyakti, Caglar**
Sammamish,
Washington 98074 (US)
- **Alabraba, Fernidand, Jay**
Seattle,
Washington 98122 (US)
- **Hughes, Aidan, T.**
Bellevue,
Washington 98007 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser
Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

Remarks:
This application was filed on 29 - 09 - 2005 as a divisional application to the application mentioned under INID code 62.

(54) Compact hardware identification for binding a software package to a computer system having tolerance for hardware changes

(57) Systems and methods for generating a compact hardware identification (CHWID) for a given computer system are disclosed. The compact hardware identification (CHWID) may be used to control the use of software on the given computer system depending on the degree of hardware changes to the computer system. The compact hardware identification (CHWID) may be electronically transmitted over limited bandwidth media, such as a telephone.

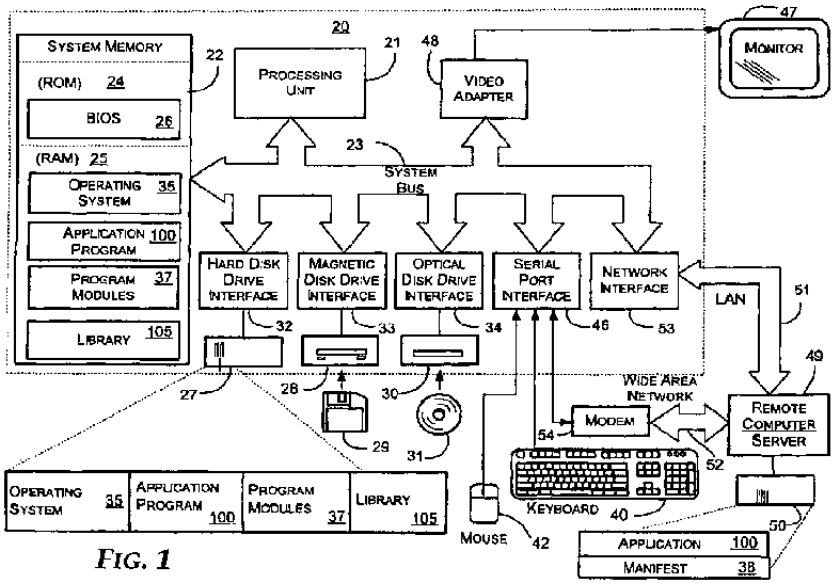


FIG. 1

EP 1 637 958 A2

Description

FIELD OF THE INVENTION

[0001] The present invention relates to systems and methods for generating a compact hardware identification (CHWID) for a given computer system. The compact hardware identification (CHWID) may be used to control the use of software on the given computer system depending on the degree of hardware changes to the computer system. The compact hardware identification (CHWID) may be electronically transmitted over limited bandwidth media, such as a telephone.

BACKGROUND OF THE INVENTION

[0002] There has been considerable effort in recent years to prevent or minimize the unlawful use of computer software. Due to its reproducibility and ease of distribution, piracy of computer software and illegal use of computer software beyond the scope of a license agreement are common occurrences, which significantly hurt software manufacturers.

[0003] Methods have been developed in an effort to reduce the occurrences of computer software piracy and illegal use of computer software beyond the scope of a license agreement. However, such methods often cause problems for legitimate software purchasers and users in the form of consumer inconvenience. For instance, a user who has upgraded his/her computer should be able to legitimately reinstall the software product on the upgraded machine. However, presently available methods may either (i) not allow the software to be installed, or (ii) force the user (who is now disgruntled) to call the software manufacturer for assistance.

[0004] Accordingly, there remains a need for improved technology solutions to piracy and illicit use, but which also recognize and accommodate the needs and practices of a legitimate software purchaser and user.

SUMMARY OF THE INVENTION

[0005] The present invention addresses some of the difficulties and problems discussed above by the discovery of an improved hardware identification for a computer system. The hardware identification of the present invention provides a method of minimizing or preventing software piracy and the illegal use of computer software beyond the scope of a license agreement, while allowing for machine upgrades by legitimate software users.

[0006] The hardware identification of the present invention, referred to herein as a "compact hardware identification" (CHWID), identifies (1) a number of component classes typically used to build a hardware configuration for a computer system, and (2) a single component device or instance within a given component class for a particular computer system. By taking into account a single component device or instance within a select number of component class, a secure and reliable compact hardware identification (CHWID) for a particular computer system is generated, while enabling a degree of tolerance for component changes to the hardware configuration of the particular computer system.

[0007] The compact hardware identification (CHWID) may be used when a limited amount of space is available to identify a particular hardware configuration when initially loading a software product onto a computer. The compact hardware identification (CHWID) may be stored for future use, such as (i) when the same software product is launched on the same computer or a variation of the same computer, or (ii) when the same software product is reloaded onto a variation of the same computer or a completely different computer. For example, when the same software product is launched on the same computer or a variation of the same computer, a second compact hardware identification (sCHWID) is generated and compared to (1) a previously stored compact hardware identification (iCHWID), or (2) a previously stored verbose hardware identification (VHWID) described below. If a desired number of matches exist between component classes of the second compact hardware identification (sCHWID) and corresponding component classes of either (1) the previously stored compact hardware identification (iCHWID), or (2) the previously stored verbose hardware identification (VHWID), the method of the present invention allows the software product to be launched. However, if a desired number of matches do not exist between component classes of the second compact hardware identification (sCHWID) and corresponding component classes of either (1) the previously stored compact hardware identification (iCHWID), or (2) the previously stored verbose hardware identification (VHWID), the method of the present invention will not allow the software product to be launched due to changes to the original hardware system beyond a desired threshold.

[0008] Accordingly, the present invention is directed to a compact hardware identification (CHWID), and a method of generating a compact hardware identification (CHWID). The present invention is further directed to a method for preventing the use of software on a computer system if an attempt to launch the software product generates a new compact hardware identification (CHWID), which is out of tolerance when compared to either (1) a previously stored compact hardware identification (iCHWID), or (2) a previously stored verbose hardware identification (VHWID) due to one or more hardware changes to the original computer system.

APPLE EXHIBIT 1102, Page 104 of 1048

[0009] These and other features and advantages of the present invention will become apparent after a review of the following detailed description of the disclosed embodiments and the appended claims.

BRIEF DESCRIPTION OF THE FIGURES

[0010]

FIG. **1** is a flow diagram of some of the primary components of an exemplary operating environment for implementation of the present invention;

FIG. **2** depicts an exemplary hardware configuration containing eight component classes and a total of 19 component devices or instances distributed within the eight component classes;

FIGS. **3** depicts one possible verbose hardware identification (VHWID) and a corresponding compact hardware identification (CHWID) for the exemplary hardware configuration shown in FIG. **2**;

FIGS. **4-8** are a flow diagram showing exemplary steps in determining a compact hardware identification (CHWID) for a hardware configuration; and

FIGS. **9-10** are a flow diagram showing exemplary steps in determining whether a software product can be used on a computer hardware system by comparing a newly generated compact hardware identification (CHWID) to either (1) a previously stored compact hardware identification (iCHWID), or (2) a previously stored verbose hardware identification (VHWID).

DETAILED DESCRIPTION OF THE INVENTION

[0011] To promote an understanding of the principles of the present invention, descriptions of specific embodiments of the invention follow and specific language is used to describe the specific embodiments. It will nevertheless be understood that no limitation of the scope of the invention is intended by the use of specific language. Alterations, further modifications, and such further applications of the principles of the present invention discussed are contemplated as would normally occur to one ordinarily skilled in the art to which the invention pertains.

[0012] The present invention is directed to a method for identifying a hardware configuration of a given computer system by a compact hardware identification (CHWID). The present invention is also directed to a method of generating a compact hardware identification (CHWID) by identifying a single component instance within each of a selected number of component classes. The present invention is further directed to a method of using a compact hardware identification (CHWID) to determine whether a software product can be used on a computer hardware configuration.

[0013] The compact hardware identification (CHWID) may be generated for a computer system comprising a variety of hardware components. An exemplary computer system may comprise a number of hardware components, which are grouped into classes including, but not limited to, hard disk drives, optical disk drives network cards, display adapters, read only memory (ROM), random access memory (RAM), and a basic input/output system (BIOS). An exemplary computer system and exemplary operating environment for practicing the present invention is described below.

Exemplary Operating Environment

[0014] Exemplary embodiments of the present invention will hereinafter be described with reference to the drawings, in which like numerals represent like elements throughout the several figures. FIG. **1** illustrates an exemplary operating environment for implementation of the present invention. The exemplary operating environment includes a general-purpose computing device in the form of a conventional personal computer 20. Generally, a personal computer **20** includes a processing unit **21**, a system memory **22**, and a system bus **23** that couples various system components including the system memory **22** to processing unit **21.** System bus **23** may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes a read only memory (ROM) **24** and random access memory (RAM) **25.** A basic input/output system (BIOS) **26**, containing the basic routines that help to transfer information between elements within personal computer **20**, such as during start-up, is stored in ROM **24.**

[0015] Personal computer **20** further includes a hard disk drive **27** for reading from and writing to a hard disk, not shown, a magnetic disk drive **28** for reading from or writing to a removable magnetic disk **29**, and an optical disk drive **30** for reading from or writing to a removable optical disk **31** such as a CD-ROM or other optical media. Hard disk drive **27**, magnetic disk drive **28**, and optical disk drive 30 are connected to system bus **23** by a hard disk drive interface **32**, a magnetic disk drive interface **33**, and an optical disk drive interface **34**, respectively. Although the exemplary environment described herein employs hard disk **27**, removable magnetic disk **29**, and removable optical disk **31**, it should be appreciated by those skilled in the art that other types of computer readable media, which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, RAMs, ROMs,

APPLE EXHIBIT 1102, Page 105 of 1048

and the like, may also be used in the exemplary operating environment. The drives and their associated computer readable media provide nonvolatile storage of computer-executable instructions, data structures, program modules, and other data for personal computer **20.** For example, one or more data files 60 (not shown) may be stored in the RAM **25** and/or hard drive **27** of the personal computer **20.**

[0016] A number of program modules may be stored on hard disk **27,** magnetic disk **29,** optical disk **31,** ROM **24,** or RAM **25,** including an operating system 35, an application program module **36,** other program modules **37,** and program data **38.** Program modules include, but are not limited to, routines, sub-routines, programs, objects, components, data structures, etc., which perform particular tasks or implement particular abstract data types. Aspects of the present invention may be implemented as an integral part of an application program module 36 or as a part of another program module **37.**

[0017] A user may enter commands and information into personal computer **20** through input devices, such as a keyboard **40** and a pointing device **42.** Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to processing unit **22** through a serial port interface **46** that is coupled to the system bus **23,** but may be connected by other interfaces, such as a parallel port, game port, a universal serial bus (USB), or the like. A monitor **47** or other type of display device may also be connected to system bus **23** via an interface, such as a video adapter **48.** In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

[0018] Personal computer 20 may operate in a networked environment using logical connections to one or more remote computers **49.** Remote computer **49** may be another personal computer, a server, a client, a router, a network PC, a peer device, or other common network node. While a remote computer **49** typically includes many or all of the elements described above relative to personal computer **20,** only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. **1** include a local area network (LAN) **51** and a wide area network (WAN) **52.** Such networking environments are commonplace in offices, enterprise-wide computer networks, Intranets, and the Internet.

[0019] When used in a LAN networking environment, personal computer **20** is connected to local area network **51** through a network interface or adapter **53.** When used in a WAN networking environment, personal computer **20** typically includes a modem **54** or other means for establishing communications over WAN **52,** such as the Internet. Modem **54,** which may be internal or external, is connected to system bus **23** via serial port interface **46.** In a networked environment, program modules depicted relative to personal computer **20,** or portions thereof, may be stored in the remote memory storage device **50.** It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0020] Moreover, those skilled in the art will appreciate that the present invention may be implemented in other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor based or programmable consumer electronics, network person computers, minicomputers, mainframe computers, and the like. The present invention may also be practiced in distributed computing environments, where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Implementation of Exemplary Embodiments of the Present Invention

[0021] As described above, a computer system typically comprises multiple classes of hardware components. Further, the computer system may comprise multiple components (e.g., two disk hard drives) within each class of hardware components.

[0022] The compact hardware identification (CHWID) of the present invention takes into account a single component device (also referred to herein as an "instance") within each desired class of hardware components used to determine the CHWID. The compact hardware identification (CHWID) of the present invention may also take into account the absence of a component device within a given component class of a computer hardware system. For example, a determination of the component classes to be used to identify a given computer hardware configuration may be made prior to examining the given computer hardware configuration. The computer hardware configuration may or may not contain a component instance for each of the selected component classes used to produce the compact hardware identification (CHWID) of the computer hardware configuration.

[0023] An exemplary method of the present invention for generating a compact hardware identification (CHWID) is given below. Further, an exemplary method of the present invention for using the compact hardware identification (CHWID) as an anti-pirating tool is also described below.

I. Generating A Compact Hardware Identification (CHWID) For A Computer System

[0024] A description of the components of an exemplary compact hardware identification (CHWID) is given below.

APPLE EXHIBIT 1102, Page 106 of 1048

A. The Compact Hardware Identification (CHWID)

[0025] The compact hardware identification (CHWID) of a given computer system desirably comprises two distinct parts: (1) an optional version component and (2) a hash component. Each of the possible compact hardware identification (CHWID) parts may be derived from the hardware configuration of a given computer system. An example of a computer hardware configuration and the instances within each component class is shown in FIG. **2.**

[0026] As shown in **FIG 2**, the exemplary computer hardware configuration **20** comprises **8** distinct component classes **21-28** having a total of 19 component instances **200** distributed among component classes **21-28.** CDRom component class **21** contains 4 component instances; IDE component class **22** contains 2 component instances; drive component class **23** contains 1 component instance; display component class **24** contains 1 component instance; SCSI component class **25** contains 2 component instances; disk component class **26** contains 6 component instances; network card component class **27** contains 1 component instance; and processor (i.e., cpu) component class **28** contains 2 component instances. Row **29** in FIG. 2 depicts a string of numbers, which represent the number of component instances within each of the 8 component classes **21-28.**

[0027] It should be noted that the number of component instances within a given component class may vary from 0 to as many as required to represent a given hardware configuration, although an implementation may arbitrarily limit the number of component instances per component class. Typically, a given hardware configuration contains from 0 to about 6 component instances per component class. As described below, even when a component class does not contain a component instance, the absence of a component instance within the component class contributes a piece of information, which is incorporated into the compact hardware identification (CHWID).

[0028] An exemplary compact hardware identification (CHWID) is shown in FIG. **3.** The exemplary compact hardware identification (CHWID) **35** of FIG. **3** is one possible way to identify the computer hardware configuration shown in FIG. **2.** As discussed above, the compact hardware identification (CHWID) **35** desirably comprises two separate components: version component **31'** and hash portion **33'.** Each of the two separate portions of compact hardware identification (CHWID) **35** is depicted in FIG **3.**

[0029] Version component **310** of header portion **31** identifies a particular version of the verbose hardware identification (VHWID) **34** and its corresponding compact hardware identification (CHWID) **35** used to identify a given computer system. For example, a particular version (e.g., version "I") of a verbose hardware identification (VHWID) or compact hardware identification (CHWID) may vary from another version (e.g., version "2") of a verbose hardware identification (VHWID) or compact hardware identification (CHWID) by using one set of parameters to create version 1, while using a separate, different set of parameters to create version 2. A variety of parameters may be used to create a given version of a verbose hardware identification (VHWID) and its corresponding compact hardware identification (CHWID). Suitable parameters include, but are not limited to, (1) the number of component classes represented in the compact hardware identification (CHWID) **35**; (2) the maximum and/or minimum number of component classes used to create verbose hardware identification (VHWID) **34**; (3) the maximum and/or minimum number of component instances to be considered within a given VHWID component class; (4) the first hash function used to produce hash values for each component instance in the verbose hardware identification (VHWID) **34;** (5) the length of each hash result used to form the verbose hardware identification (VHWID) **34**; (6) the maximum length of the verbose hardware identification (VHWID) **34**; (7) the maximum and/or minimum number of component class hash results used to create the compact hardware identification (CHWID) **35;** (8) the second hash function used to produce second hash values for each component instance; (9) the length of each second hash result used to form the compact hardware identification (CHWID) **35;** and (10) the maximum length of the compact hardware identification (CHWID) **35.**

[0030] Some component classes cannot have multiple component instances and are known as single-instance classes. Total system RAM is one example of a single-instance class. The data for a single-instance class is hashed and truncated if necessary, then stored in the truncated numerical portion **312** of the header part **31.** Each single-instance class represented in the header will have an associated truncated numerical portion **312.**

[0031] Desirably, version component **31'** of compact hardware identification (CHWID) **35** has a fixed length, which is consistent for all compact hardware identifications having a particular version type (e.g., version 1 CHWIDs).

[0032] Count portion **32** comprises a string of n numbers, which represent the number of component instances within each of the n component classes used to prepare the verbose hardware identification (VHWID) **34.** As shown in FIG. **3,** count portion **32** comprises the string of numbers: "4 2 1 1 2 6 1 2", which corresponds to the component instances within component classes **21-28** shown in FIG. **2.** It should be noted that if a component class does not contain a component instance, count portion **32** contains a "0" for the particular component class.

[0033] Hash portion **33** used to create verbose hardware identification (VHWID) **34** comprises a concatenated string of first hash values representing each of the 19 component instances within component classes **21-28** shown in FIG. **2.** Each separate hash result may have a length of up to about 160 bits. Desirably, each separate hash result has a length of from about 10 to about 20 bits, more desirably, about 16 bits.

[0034] Hash portion **33** of verbose hardware identification (VHWID) **34** typically has a length, which varies depending

APPLE EXHIBIT 1102, Page 107 of 1048

upon the number of component instances within n component classes of a given hardware configuration. The total length of hash portion **33** is equal to the number of component instances times the desired hash result length for each component instance hash result. In this example, the resulting hash portion **33** of verbose hardware identification (VHWID) **34** has a desired total length of 304 bits (i.e., 19 x16 = 304).

[0035] Hash portion **33'** of compact hardware identification (CHWID) **35** differs from hash portion **33** used to create verbose hardware identification (VHWID) **34.** In one exemplary embodiment of the present invention, hash portion **33'** of compact hardware identification (CHWID) 35 comprises one component instance second hash value per component class, resulting from a second hash function performed on one component instance first hash value per component class. The component instance first hash value is selected from one or more of the n component classes used to create hash portion **33** of verbose hardware identification (VHWID) **34.** The method of choosing component instance first hash values within hash portion 33 to be further processed through a second hash function may be (i) by a random selection procedure or (ii) by a predetermined method. One exemplary predetermined method comprises selecting the first instance within each component class as shown in FIG. **3.**

[0036] The resulting hash portion **33'** of compact hardware identification (CHWID) **35** comprises a concatenated string of component instance second hash values (v'_x) resulting from performing a second hash function on select component instance first hash values of hash portion 33. Each separate component instance second hash value may have a length of up to about 16 bits. Desirably, each separate component instance second hash value has a length of up to about 8 bits, more desirably, from about 3 to about 6 bits. Component instance second hash values (v'_x) are shown in FIG. 3 and are derived by performing a second hash function on the following string of first hash values: v_1, v_5, v_7, v_8, v_9, v_{11}, v_{17} and v_{18} to produce component instance second hash values: v'_1, v'_5, v'_7, v'_8, v'_9, v'_{11}, v'_{17} and v'_{18}.

[0037] Hash portion **33'** of compact hardware identification (CHWID) **35** typically has a length of less than about 64 bits. The length of hash portion **33'** may vary depending upon (i) the number of component instance first hash values used to create hash portion **33',** and (ii) the second hash value length for each individual component instance second hash value.

[0038] As shown in FIG. **3**, verbose hardware identification (VHWID) **34** may be represented by a concatenated string of header portion **31**, count portion **32** and hash portion **33.** Similarly, compact hardware identification (CHWID) **35** may be represented by a concatenated string of version component **31'** and hash portion **33'.** An exemplary method of determining a verbose hardware identification (VHWID) **34** and a compact hardware identification (CHWID) **35** for a computer hardware configuration is described below.

[0039] It should be noted that the compact hardware identification (CHWID) of the present invention may only comprise one of the above-described portions. In one exemplary embodiment of the present invention, the compact hardware identification (CHWID) **35** of a given computer hardware configuration comprises hash portion **33'** alone. In this embodiment, the compact hardware identification (CHWID) 35 does not contain version component **31'.**

[0040] Regardless of the components used to create compact hardware identification (CHWID) **35**, compact hardware identification (CHWID) **35** desirably has a total length of less than about 256 bits. In one exemplary embodiment of the present invention, compact hardware identification (CHWID) **35** has a total length of from about **32** bits to about 64 bits.

B. Determining A Verbose Hardware Identification (VHWID) For A Computer System

[0041] The VHWID of the present invention may be determined by an exemplary method as shown in FIGS. **4-6.** The steps of the exemplary method may be performed by software code within a software product on a customer's computer, similar to computer 20 described above with reference to FIG. **1.** As shown in FIGS. **4-6**, an exemplary determination of a VHWID for a given computer hardware configuration (referred to herein as "HW1") begins with step **401,** wherein a number of component classes, n, is chosen to identify a given computer hardware configuration HW1. As discussed above, a given computer system may include a variety of hardware components and classes of hardware components. Exemplary hardware component classes include, but are not limited to, hard disk drives, logical disk partitions, optical disks, network cards, display adapters, read only memory (ROM), random access memory (RAM), IDE devices, sound cards, video cards, processors, SCSI devices and the system BIOS. Desirably, n, the number of hardware component classes, is a whole number ranging from about 2 to about 16. In general, it is desirable for n to be as large as possible in order (i) to more precisely identify a given computer system, and (ii) to more accurately measure the degree of tolerance of a given computer system.

[0042] After choosing the number of component classes, n, in step 401, each component class is identified in step **402.** The component classes may include any of the above-described component classes such as the class of disk hard drives. An exemplary list of component classes used to identify sample hardware configuration HW1 is given below in Table 1.

APPLE EXHIBIT 1102, Page 108 of 1048

Table 1. Exemplary List of Hardware Component Classes Used To Identify Sample Hardware Configuration HW1

Component Class No.	Class Description	Class Identifier
1	CdRom	CdRom device identifier
2	IDE devices	IDE device identifier
3	Hard Disk Drive	Drive partition serial number
4	Display adapter device	Identifier
5	SCSI devices	SCSI device identifier
6	Disk Devices	Disk device identifier
7	Network Card	MAC address
8	Processors	Processor device identifier

[0043] As shown in Table 1, in this example, n equals 8, and the identified hardware component classes include: (1) a CdRom class; (2) an IDE devices class; (3) a drive class; (4) a display adapter device class; (5) a SCSI device class; (6) a disk class; (7) a network card class; and (8) a CPU processor class.

[0044] After each component class is identified in step **402,** all devices or instances within each hardware component class are identified in step 403. The "count" (i.e., the number of component devices or instances within each component class) is also determined in step **403.** Desirably, each instance within a particular component class is identified by the most unique identification string associated with the instance. For example, the hardware configuration may contain a CdRom manufactured by NEC Corporation and having an identification string of "NEC CDRW24 S15." Any available method for determining the most unique identification string of a given instance may be used in the present invention. The step of assigning an identification string for each component instance is shown in step **404.**

[0045] Once an identification string for each component instance is assigned, the header portion of the verbose hardware identification (VHWID) is prepared in step **405.** In step **406**, a particular version of the verbose hardware identification (VHWID) is inputted into the header to form header portion **310** (as shown in FIG 3). As described above, the version number may represent one or more parameters used to determine the verbose hardware identification (VHWID) and its corresponding compact hardware identification (CHWID).

[0046] In step **407,** a component class to be represented in the header is identified. Typically, component classes capable of having only a single component instance, or single instance classes, are represented in the header portion of the VHWID. Suitable component classes, which may be represented in the header portion of the VHWID, included, but are not limited to, a memory component class, a computer dockability component class (i.e., whether the computer is dockable or not), the system BIOS, or a combination thereof. In one exemplary embodiment of the present invention, the header portion of the VHWID comprises information from a single component class of the hardware configuration.

[0047] From step **407,** the method proceeds to decision block **409.** At decision block **409,** a decision is made as to whether the identification string of the component instance used to form a portion of the header is subjected to a hashing function. The identification string may be subjected to a hash function or truncated to a desired number of bits. Although not shown in FIG. 5 as an option, it should be noted that the identification string could be used verbatim as long as the identification string has less than a desired maximum of characters, typically less than about 16 bits.

[0048] If the identification string is to be subjected to a hash function, the method proceeds to step **411,** wherein a hash function is performed on the identification string of the component instance and truncated to a desired bit length. Desirably, the hash result is truncated to a length of about 16 bits. In step **412,** the truncated hash result is inputted into the truncated numerical portion **312** of header portion **31** (as shown in FIG. 3). If the identification string is not subjected to a hash function, the method proceeds to step **410,** where the identification string is truncated to a desired length and inputted into the truncated numerical portion **312** of header portion **31.** Desirably, the identification string is truncated to a length of less than about 16 bits.

[0049] Once a truncated hash result from step **412** or a truncated identification string from step **410** is inputted into truncated numerical portion **312** of header portion **31**, the method proceeds to decision block **413.** At decision block **413,** a decision is made whether to add details of another component class to header portion **31** of the VHWID. If additional details of another component class are to be added to the header portion **31** of the VHWID, the method returns to step **407** and proceeds as described above. If no further information is to be added to the header portion **31** of the VHWID, the method proceeds to step **414,** where the count portion **32** of the VHWID is prepared. As discussed above, the count portion **32** of the VHWID comprises a numerical string of n numbers, which represent the number of component instances within each of the n component classes used to form the VHWID. (See count portion **32** of FIG **3.**)

[0050] In step **415,** a first hash function is performed on the identification strings for each component instance repre-

APPLE EXHIBIT 1102, Page 109 of 1048

sented in the count portion **32** of the VHWID. If a given component class does not contain a component instance, a special first hash result may be generated for use in the VHWID, wherein the special first hash result indicates that a given component class did not contain a component instance. Alternatively, no first hash value may be stored and the part of count potion **32** corresponding to the missing component class will be set to zero, indicating that the component class is absent. The first hash results for each component instance may be truncated to a desired length. In one exemplary embodiment of the present invention, each of the first hash function results are truncated to a length of from about 10 to about 20 bits, more desirably, about 16 bits.

[0051] Any known hash functions may be used in the present invention as long as the hash function is capable of accepting an identification string of arbitrary length and producing a hash output or result having a fixed length of less than about 160 bits. Examples of suitable hash functions include, but are not limited to, hash function algorithms HAVAL, MD2, MD4, MD5, and SHA-1, all of which are known to those of ordinary skill in the art. Suitable hash functions and a description thereof may be found in Applied Cryptography by Bruce Schneier, published by John Wiley & Sons (ISBN #0471117099), the disclosure of which is incorporated herein in its entirety.

[0052] In one embodiment of the present invention, a "salt value" may be added to the component instance identifier prior to performing the first hash function for a given component instance. In this embodiment, adding a salt value enables the production of different VHWIDs based on the same computer hardware configuration. Different VHWIDs for the same hardware configuration may be beneficial when running different applications or different passes. One example of a situation where different VHWIDs for the same hardware configuration may be beneficial is discussed below.

[0053] For example, if a user activates multiple software packages from the same vendor, it may be possible to use the VHWID to relate the separate activation records to build a picture of the software purchasing habits of the user. To guard against this, different VHWIDs from the same machine may be made to appear unrelated by constructing each separate hash using a hash function such as $hash_x = MD5[(salt\ value)_x + ID\ string]$ where the salt value is different for each software package.

[0054] In step **416**, the first hash results for each component instance are concatenated to form hash portion 33 of verbose hardware identification (VHWID) **34** (as shown in FIG. **3**). In step **417**, the final verbose hardware identification (VHWID) **34** is assembled by concatenating header part **31**, count part **32**, and hash part **33.**

[0055] In step **418**, the resulting verbose hardware identification (VHWID) for hardware configuration HW1 is stored for future use. The verbose hardware identification (VHWID) for hardware configuration HW1 may be stored locally (e.g., in the registry, file system, or secure store), at an accessible remote location (e.g., database), or transmitted to a clearinghouse server for license acquisition.

[0056] Although the exemplary method described above produces a verbose hardware identification (VHWID) containing header part **31**, count part **32**, and hash part **33**, in some embodiments of the present invention, the verbose hardware identification (VHWID) for hardware configuration HW1 may only contain (i) hash portion **33** alone or (ii) count part **32** in combination with hash portion **33**, such as a VHWID comprising count part **32** concatenated with hash portion **33.**

C. Determining A Compact Hardware Identification (CHWID) For A Computer System

[0057] The compact hardware identification (CHWID) of the present invention may be determined by as shown in FIGS. **7-8.** The steps of the exemplary method for forming the compact hardware identification (CHWID) may be performed by software code within a software product on a customer's computer, similar to computer 20 described above with reference to FIG. **1.** As shown in FIGS. **7-8**, the exemplary method for forming one possible compact hardware identification (CHWID) for hardware configuration HW1 begins with step **420.**

[0058] In step **420**, a second hash function is performed on one component instance first hash value from each of the q component classes selected from one or more component classes, *n*, used to create hash portion **33** of verbose hardware identification (VHWID) **34.** As described above, the second hash function may be performed on one or more component instance first hash values, wherein the method of selecting component instance first hash values from one or more of the n component classes is accomplished via (i) a random selection procedure or (ii) a predetermined method. Desirably, one component instance first hash value is selected from at least (n - 5) component classes, more desirably, from at least (n - 3) component classes, even more desirably, from at least (n - 2) component classes. In one exemplary embodiment of the present invention, one component instance first hash value is selected from all of the n component classes to form corresponding component instance second hash values.

[0059] As with the first hash function used to form the hash portion 33 of verbose hardware identification (VHWID) **34**, any known hash function may be used in the present invention as long as the hash function is capable of accepting a component instance first hash value of up to about 160 bits and producing a component instance second hash value having a fixed length of less than about 32 bits. Examples of suitable second hash functions include, but are not limited to, hash function algorithms HAVAL, MD2, MD4, MD5, and SHA-1, all of which are known to those of ordinary skill in the art as discussed above.

[0060] In one embodiment of the present invention, a "salt value" may be added to the component instance first hash

APPLE EXHIBIT 1102, Page 110 of 1048

value prior to performing the second hash function for a given component instance first hash value. In this embodiment, adding a salt value enables the production of different compact hardware identifications (CHWIDs) based on the same computer hardware configuration. Different compact hardware identifications (CHWIDs) for the same hardware configuration may be beneficial when running different applications or different passes. One example of a situation where different CHWIDs for the same hardware configuration may be beneficial is discussed below.

[0061] For example, if a user activates multiple software packages from the same vendor, it may be possible to use the CHWID to relate the separate activation records to build a picture of the software purchasing habits of the user. To guard against this, different CHWIDs from the same machine may be made to appear unrelated by constructing each separate hash using a hash function such as $hash_x = MD5[(salt\ value)_x + ID\ string]$ where the salt value is different for each software package.

[0062] In step **421**, the component instance second hash values are concatenated to form hash portion **33'** of compact hardware identification (CHWID) **35** (as shown in FIG. **3**).

[0063] In step **427**, the version component **31'** of the compact hardware identification (CHWID) is concatenated with the hash portion **33'** of the compact hardware identification (CHWID) to form the final compact hardware identification (CHWID) for hardware configuration HW1. The method then proceeds to step **428.** Returning to decision block **425** described above, if the compact hardware identification (CHWID) does not comprise a version component **31'**, the method proceeds directly to step **428.**

[0064] In step **428**, the resulting compact hardware identification (CHWID) for hardware configuration HW1 is stored for future use. The compact hardware identification (CHWID) for hardware configuration HW1 may be stored locally (e.g., in the registry, file system, or secure store), or at an accessible remote location (e.g., database) as described below.

[0065] As discussed above, in some embodiments of the present invention, the compact hardware identification (CHWID) for hardware configuration HW1 may only contain hash portion **33'.**

II. Using A Compact Hardware Identification (CHWID) To Enable The Use Of A Software Product On A Computer System

[0066] The present invention is further directed to a method of using a compact hardware identification (CHWID) to enable the use of a software product on a computer system having a given computer hardware configuration. In one embodiment of the present invention, the method of using a compact hardware identification (CHWID) to enable the use of a software product on a computer system having a given computer hardware configuration is initiated (i) during any installation of the software product on a computer other than an initial installation, (ii) during launching of a software product or application already existing on a component of a computer hardware configuration, or (iii) both. An exemplary method for using the compact hardware identification (CHWID) is described in FIGS. **9-10.** The steps of the exemplary method may be performed by software code within a software product on a customer's computer, similar to computer 20 described above with reference to FIG. **1.**

[0067] As shown in step **501** of FIG. **9**, a software product is either loaded or launched on a computer having hardware configuration HW2. Computer hardware configuration HW2 (i) may be identical to hardware configuration HW1 used to produce an initial verbose hardware identification (referred to herein as iVHWID) or an initial compact hardware identification (referred to herein as iCHWID) or (ii) may be a completely different computer.

[0068] In step **502**, a new compact hardware identification (referred to herein as nCHWID) is generated for computer hardware configuration HW2. The new compact hardware identification (nCHWID) for computer hardware configuration HW2 may be generated as described above and shown in FIGS. **4-8.** Once a new compact hardware identification (nCHWID) is generated for computer hardware configuration HW2, a stored verbose hardware identification (VHWID) or a stored compact hardware identification (CHWID) is retrieved in step 503. Typically, the stored verbose hardware identification (VHWID) is the initial verbose hardware identification (iVHWID), which was generated on a first computer hardware configuration HW1 during an initial software product installation onto HW1. Similarly, the stored compact hardware identification (CHWID) is typically the initial compact hardware identification (iCHWID), which was generated on a first computer hardware configuration HW1 during an initial software product installation onto HW1.

[0069] In decision block **504,** a determination is made whether the previously stored hardware identification is a stored compact hardware identification (CHWID). If the previously stored hardware identification is a stored compact hardware identification (sCHWID), the method proceeds to step **505,** wherein the new compact hardware identification (nCHWID) of hardware configuration HW2 is compared with the previously stored compact hardware identification (sCHWID) of hardware configuration HW1. If the previously stored hardware identification is a stored verbose hardware identification (sVHWID), the method proceeds to step **509.** In step **509,** the new compact hardware identification (CHWID) is compared to the stored verbose hardware identification (VHWID). A second hash function is performed for each of the first hash values in each component class of the stored verbose hardware identification (VHWID) and the results compared to the second hash value associated with each component class in the new compact hardware identification (CHWID). The method then proceeds to decision block **506.**

[0070] At decision block **506,** a determination is made as to whether the number of component class matches equals

APPLE EXHIBIT 1102, Page 111 of 1048

or exceeds a required number of component class matches, *m*, needed to enable the use of the software product on hardware configuration HW2. If the number of component class matches equals or exceeds a required number of component class matches, m, the method proceeds to step **507**, wherein the method enables the use of the software product on hardware configuration HW2. If the number of component class matches is less than the required number of component class matches, m, the method proceeds to step **508**, wherein the method disables the use of the software product on hardware configuration HW2.

[0071] In step 505, the comparison of new compact hardware identification (nCHWID) of hardware configuration HW2 with (1) the previously stored compact hardware identification (CHWID) of hardware configuration HW1 or (2) the previously stored verbose hardware identification (VHWID) of hardware configuration HW1, collectively referred to herein as "the hardware identification (HWID) of hardware configuration HW1," may involve one or more rules for determining whether or not there is a match for a given component class. Desirably, the method of using a compact hardware identification (CHWID) to enable the use of a software product comprises one or more of the following rules for determining the number of component class matches between a newly generated compact hardware identification (nCHWID) for a hardware configuration HW2 and the compact hardware identification (CHWID) or verbose hardware identification (VHWID) of hardware configuration HW1:

(i) each component instance second hash result within new compact hardware identification (nCHWID) representing select component instances within one or more component classes of hardware configuration HW2 is compared with each component instance second hash result within the corresponding one or more component classes in the compact hardware identification (CHWID) or derived from the verbose hardware identification (VHWID) of hardware configuration HW1;

(ii) a match exists between a component class of hardware configuration HW2 and a corresponding component class of hardware configuration HW1 when one second component instance hash result within a component class of new compact hardware identification (nCHWID) for hardware configuration HW2 matches any one of the second component instance hash results within the corresponding component class of the compact hardware identification (CHWID) or derived from the verbose hardware identification (VHWID) of hardware configuration HW1;

(iii) a single match exists between a component class of hardware configuration HW2 and a corresponding component class of hardware configuration HW1 when one second component instance hash result within a component class used to form new compact hardware identification (nCHWID) for hardware configuration HW2 matches two or more derived second component instance hash results within a corresponding component class used to form the verbose hardware identification (VHWID) of hardware configuration HW1;

(iv) no match exists between a component class of hardware configurations HW2 and a corresponding component class of hardware configuration HW1 when the component class in hardware configuration HW2 does not contain a second component instance hash result, and the corresponding component class in hardware configuration HW1 does contain a second component instance hash result;

(v) no match exists between a component class of hardware configuration HW2 and a corresponding component class of hardware configuration HW1 when the component class in hardware configuration hardware configuration HW2 contains a single second component instance hash result, and the corresponding component class in hardware configuration HW1 does not contain a second component instance hash result; and

(vi) a match exists between a component class of hardware configuration HW2 and a corresponding component class of hardware configuration HW1 when the component class in hardware configuration hardware configuration HW2 does not contain a second component instance hash result, and the corresponding component class in hardware configuration HW1 does not contain a second component instance hash result; and

(vii) the number of required component classes matches, m, between hardware configuration HW2 and hardware configuration HW1 may be predetermined and embedded in code on a given software product.

[0072] The number of required component class matches, m, is chosen depending on the degree of tolerance desired for hardware configuration component changes. The number of required component class matches, *m*, may be (i) as great as *n*, the total number of component classes considered during the determination of a verbose hardware identification (VHWID), or (ii) as great as q, the total number of selected component classes considered during the determination of the compact hardware identification (CHWID), or (iii) may be as small as 1. As m increases, the degree of tolerance to computer hardware configuration changes decreases. For example, if the total number of component classes n is equal to 10 and m is equal to 7, 7 out of 10 component classes must match at least one component instance to enable the loading or running of a software product. If the number of component class matches is less than 7, the software product will not run or be loaded onto the computer hardware configuration.

[0073] The number of required component class matches, m, may be predetermined by a software manufacturer and encoded into the software product code used to generate a compact hardware identification (CHWID). In one exemplary embodiment of the present invention, m is desirably equal to (*n* - 3). More desirably, *m* is equal to *(n* - 2). In another

APPLE EXHIBIT 1102, Page 112 of 1048

exemplary embodiment of the present invention, m is desirably equal to (q - 3). More desirably, m is equal to (q - 2). However, as indicated above, m may range from 1 to n.

[0074] The method steps described above and illustrated in **FIGS. 4-10** may be performed locally or at a remote location. Typically, a customer purchases a software product that can run on a given computer, such as computer **20** shown in FIG. **1.** The software product may be a shrink-wrap product having a software program stored on a transportable computer-readable medium, such as a CD-ROM or floppy diskette. Alternatively, the software product may be delivered electronically over a network, such as a local area network (LAN) **51** or a wide area network (WAN) **52.** The customer loads the software product onto the computer 20 as a program stored in system memory 22.

[0075] During a software product installation, the customer is typically prompted to enter a portion of the software product identification (PID) for the software product into computer **20.** The PID may be derived, for example, from a CD key printed on a label of the shrink-wrap package. The customer enters the PID, which is associated with a software program of the software product. The PID is stored locally on computer 20 and/or remotely at an accessible location, either on a local area network (LAN) 51 or a wide area network (WAN) **52** with a third party, such as an activation authority.

[0076] As described above, during installation or activation of the software product, a verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) is also generated using code within the software product or triggered by the installation of the software product. The verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) generated by the method of the present invention is associated with the software product identification (PID) and stored along with the software product identification (PID) locally on computer 20 and/or remotely at an accessible location, either on a local area network (LAN) **51** or a wide area network (WAN) **52,** such as with a third party activation authority.

[0077] As part of the installation process, the customer may be required to activate the software product with an activation authority. This authority might be, for example, the product manufacturer or an authorized third party. The activation process is intended to force the customer to activate the software product (i) for installation and use on a specific computer or (ii) for installation and use according to terms of a product licensing agreement. Such an activation process is described in detail in U.S. Patent No. 6,243,468, assigned to Microsoft Corporation (Redmond, WA), the contents of which are hereby incorporated in its entirety by reference.

[0078] The verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) generated by the method of the present invention and the software product identification (PID) may be stored locally on computer 20 and/or remotely at an accessible location, either on a local area network (LAN) **51** or a wide area network (WAN) **52** with an activation authority. Desirably, the software product (i) stores (a) the verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) and (b) the associated software product identification (PID) on computer **20,** and (ii) sends (a) the verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) and (b) the associated software product identification (PID) electronically over wide area network (WAN) **52** to an activation server. Desirably, the software product automatically displays a graphical user interface (UI) dialog window when it is first launched, which prompt the user to initiate a connection with the activation server to activate. The activation server maintains a database to store (a) received verbose hardware identifications (VHWIDs) and/or compact hardware identifications (CHWIDs) and (b) the associated software product identifications (PIDs).

[0079] The verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) and the associated software product identification (PID) for a given software product may be stored for an indefinite period of time until the software product is re-installed onto another computer or launched on the first computer (i.e., the computer used during the initial installation). When the same software product is re-installed onto another computer or launched on the first computer, code on the software product initiates a method of generating a new compact hardware identification (CHWID) according to the present invention. The software product also retrieves the previously stored (a) verbose hardware identification (VHWID) and/or compact hardware identification (CHWID) and (b) the associated software product identification (PID) of the software product either from local computer 20 or from a remote location via a local area network (LAN) **51** or a wide area network (WAN) **52.** A comparison between the new compact hardware identification (CHWID) and the previously stored compact hardware identification (CHWID) is made as described above.

[0080] In an alternative manual case, a customer provides a service representative with a compact hardware identification (CHWID) over the phone and the service representative provides the customer with a confirmation identification (CID) based on the compact hardware identification (CHWID). The customer enters the confirmation identification (CID) via a UI window.

[0081] When the use of a software product is denied due to significant changes in the hardware configuration of a first computer (i.e., the computer used during the initial installation), a dialog box may be provided to the customer indicating that the use of the software product is being denied, and that further information regarding future use of the software product may be obtained from a given source.

APPLE EXHIBIT 1102, Page 113 of 1048

III. Other Uses of A Compact Hardware Identification (CHWID)

[0082] The compact hardware identification (CHWID) of the present invention may also be used for other purposes than those described above. In one embodiment of the present invention, the compact hardware identification (CHWID) is used to create semi-unique installation ID to track the machine. In another embodiment of the present invention, the compact hardware identification (CHWID) is used on a clearinghouse server when granting licenses to use software on a customer's computer.

[0083] While the specification has been described in detail with respect to specific embodiments thereof, it will be appreciated that those skilled in the art, upon attaining an understanding of the foregoing, may readily conceive of alterations to, variations of, and equivalents to these embodiments. Accordingly, the scope of the present invention should be assessed as that of the appended claims and any equivalents thereto.

The following is a list of further preferred embodiments of the invention:

[0084]

Embodiment 1. A method of generating a compact hardware identification (CHWID) for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;

identifying all component instances within each of the n component classes;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification (CHWID) for the first computer system.

Embodiment 2. The method of embodiment 1, further comprising:

concatenating (i) a version component and (ii) the hash portion of the compact hardware identification (CHWID) to form the compact hardware identification (CHWID) for the first computer system.

Embodiment 3. The method of embodiment 1, wherein n is a whole number up to about 16.

Embodiment 4. The method of embodiment 2, wherein the version component comprises a version number.

Embodiment 5. The method of embodiment 1, wherein at least one of the n component classes contains two or more component instances.

Embodiment 6. The method of embodiment 1, wherein each of the n component classes contains from 0 to 14 component instances.

Embodiment 7. The method of embodiment 1, wherein each component instance first hash result is truncated to a 16 bit number.

Embodiment 8. The method of embodiment 1, wherein each component instance second hash result is truncated to a number having less than 8 bits.

Embodiment 9. The method of embodiment 1, wherein the compact hardware identification (CHWID) for the first computer system has a length of up to about 256 bits.

APPLE EXHIBIT 1102, Page 114 of 1048

Embodiment 10. The method of embodiment 9, wherein the compact hardware identification (CHWID) for the first computer system has a length of from about 32 to about 64 bits.

Embodiment 11. The method of embodiment 1, wherein the method is initiated during a step of loading a software product onto the first computer system.

Embodiment 12. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing the method of embodiment 1.

Embodiment 13. A computer readable medium having stored thereon computer-executable instructions for performing the method of embodiment 1.

Embodiment 14. A method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

generating a second compact hardware identification (sCHWID) for the second hardware configuration;

comparing the second compact hardware identification (sCHWID) for the second hardware configuration to (i) a first compact hardware identification (fCHWID) for the first hardware configuration or (ii) a first verbose hardware identification (fVHWID) for the first hardware configuration;

if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds m, a number of required component class matches, loading the software product onto the second computer system; and

if the number of matches is less than m, preventing the software product from being loaded onto the second computer system.

Embodiment 15. The method of embodiment 14, wherein the second compact hardware identification (sCHWID) is generated by a method comprising:

selecting n component classes of the second hardware configuration;

identifying all component instances within each of the n component classes of the second hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification (sCHWID) for the second hardware configuration.

Embodiment 16. The method of embodiment 14, wherein the first compact hardware identification (fCHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first

APPLE EXHIBIT 1102, Page 115 of 1048

hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification (fCHWID) for the first hardware configuration.

Embodiment 17. The method of embodiment 14, wherein the first verbose hardware identification (fVHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and

concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 18. The method of embodiment 14, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification (fCHWID) for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 19. The method of embodiment 14, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 20. The method of embodiment 14, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 21. The method of embodiment 14, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 22. The method of embodiment 14, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification (sCHWID).

Embodiment 23. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing the method of embodiment 14.

APPLE EXHIBIT 1102, Page 116 of 1048

Embodiment 24. A computer readable medium having stored thereon computer-executable instructions for performing the method of embodiment 14.

Embodiment 25. A computer readable medium having stored thereon computer-executable instructions for performing a method of generating a compact hardware identification (CHWID) for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;

identifying all component instances within each of the n component classes;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification (CHWID) for the first computer system.

Embodiment 26. The computer readable medium of embodiment 25, further comprising:

concatenating (i) a version component and (ii) the hash portion of the compact hardware identification (CHWID) to form the compact hardware identification (CHWID) for the first computer system.

Embodiment 27. The computer readable medium of embodiment 25, wherein n is a whole number up to about 16.

Embodiment 28. The computer readable medium of embodiment 27, wherein the version component comprises a version number.

Embodiment 29. The computer readable medium of embodiment 25, wherein at least one of the n component classes contains two or more component instances.

Embodiment 30. The computer readable medium of embodiment 25, wherein each of the n component classes contains from 0 to 14 component instances.

Embodiment 31. The computer readable medium of embodiment 25, wherein the compact hardware identification (CHWID) for the first computer system has a length of up to about 256 bits.

Embodiment 32. The computer readable medium of embodiment 25, wherein the method is initiated during a step of loading a software product onto the first computer system.

Embodiment 33. A computer readable medium having stored thereon computer-executable instructions for performing a method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

generating a second compact hardware identification (sCHWID) for the second hardware configuration;

comparing the second compact hardware identification (sCHWID) for the second hardware configuration to (i) a first compact hardware identification (fCHWID) for the first hardware configuration or (ii) a first verbose hardware identification (fVHWID) for the first hardware configuration;

if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds

APPLE EXHIBIT 1102, Page 117 of 1048

m, a number of required component class matches, loading the software product onto the second computer system; and

if the number of matches is less than m, preventing the software product from being loaded onto the second computer system.

Embodiment 34. The computer readable medium of embodiment 33, wherein the second compact hardware identification (sCHWID) is generated by a method comprising:

selecting n component classes of the second hardware configuration;

identifying all component instances within each of the n component classes of the second hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification (sCHWID) for the second hardware configuration.

Embodiment 35. The computer readable medium of embodiment 33, wherein the first compact hardware identification (fCHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification (fCHWID) for the first hardware configuration.

Embodiment 36. The computer readable medium of embodiment 33, wherein the first verbose hardware identification (fVHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and

concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification (fVHWID) for the first hardware configuration.

APPLE EXHIBIT 1102, Page 118 of 1048

Embodiment 37. The computer readable medium of embodiment 33, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification (fCHWID) for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 38. The computer readable medium of embodiment 33, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 39. The computer readable medium of embodiment 33, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 40. The method of embodiment 33, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 41. The computer readable medium of embodiment 33, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification (sCHWID).

Embodiment 42. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing a method of generating a compact hardware identification (CHWID) for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;

identifying all component instances within each of the n component classes;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification (CHWID) for the first computer system.

Embodiment 43. The computing system of embodiment 42, further comprising:

concatenating (i) a version component and (ii) the hash portion of the compact hardware identification (CHWID) to form the compact hardware identification (CHWID) for the first computer system.

Embodiment 44. The computing system of embodiment 42, wherein n is a whole number up to about 16.

APPLE EXHIBIT 1102, Page 119 of 1048

Embodiment 45. The computing system of embodiment 43, wherein the. version component comprises a version number.

Embodiment 46. The computing system of embodiment 42, wherein at least one of the n component classes contains two or more component instances.

Embodiment 47. The computing system of embodiment 42, wherein each of the n component classes contains from 0 to 14 component instances.

Embodiment 48. The computing system of embodiment 42, wherein the compact hardware identification (CHWID) for the first computer system has a length of up to about 256 bits.

Embodiment 49. The computing system of embodiment 42, wherein the method is initiated during a step of loading a software product onto the first computer system.

Embodiment 50. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing a method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

generating a second compact hardware identification (sCHWID) for the second hardware configuration;

comparing the second compact hardware identification (sCHWID) for the second hardware configuration to (i) a first compact hardware identification (fCHWID) for the first hardware configuration or (ii) a first verbose hardware identification (fVHWID) for the first hardware configuration;

if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds m, a number of required component class matches, loading the software product onto the second computer system; and

if the number of matches is less than m, preventing the software product from being loaded onto the second computer system.

Embodiment 51. The computing system of embodiment 50, wherein the second compact hardware identification (sCHWID) is generated by a method comprising:

selecting n component classes of the second hardware configuration;

identifying all component instances within each of the n component classes of the second hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification (sCHWID) for the second hardware configuration.

Embodiment 52. The computing system of embodiment 50, wherein the first compact hardware identification (fCHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

APPLE EXHIBIT 1102, Page 120 of 1048

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;

generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and

concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification (fCHWID) for the first hardware configuration.

Embodiment 53. The computing system of embodiment 50, wherein the first verbose hardware identification (fVHWID) is generated by a method comprising:

selecting n component classes of the first hardware configuration;

identifying all component instances within each of the n component classes of the first hardware configuration;

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and

concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 54. The computing system of embodiment 50, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification (fCHWID) for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 55. The computing system of embodiment 50, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification (sCHWID) for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification (fVHWID) for the first hardware configuration.

Embodiment 56. The computing system of embodiment 50, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 57. The method of embodiment 50, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

Embodiment 58. The computing system of embodiment 50, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification (sCHWID).

APPLE EXHIBIT 1102, Page 121 of 1048

Important Note:

[0085] While the attached claims relate to a preferred aspect of the present invention, the applicant wishes to reserve the right to file one or several further divisional applications at a later point in time for other aspects disclosed in the application. Those further applications will be divided out from the present divisional application. By this statement, the public is herewith informed that more divisional applications relating to different subject matter may follow.

Claims

1. A method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

   generating (502) a second compact hardware identification for the second hardware configuration;
   comparing (505, 509) the second compact hardware identification for the second hardware configuration to (i) a first compact hardware identification for the first hardware configuration or (ii) a first verbose hardware identification for the first hardware configuration;
   if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds m, a number of required component class matches, loading (507) the software product onto the second computer system; and
   if the number of matches is less than m, preventing (508) the software product from being loaded onto the second computer system.

2. The method of Claim 1, wherein the second compact hardware identification is generated by a method comprising:

   selecting n component classes of the second hardware configuration;
   identifying all component instances within each of the n component classes of the second hardware configuration;
   generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;
   generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and
   concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification for the second hardware configuration.

3. The method of Claim 1, wherein the first compact hardware identification is generated by a method comprising:

   selecting n component classes of the first hardware configuration;
   identifying all component instances within each of the n component classes of the first hardware configuration;
   generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;
   generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and
   concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification for the first hardware configuration.

4. The method of Claim 1, wherein the first verbose hardware identification is generated by a method comprising:

   selecting n component classes of the first hardware configuration;
   identifying all component instances within each of the n component classes of the first hardware configuration;

APPLE EXHIBIT 1102, Page 122 of 1048

generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and
concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification for the first hardware configuration.

5. The method of Claim 1, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification for the first hardware configuration.

6. The method of Claim 1, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification for the first hardware configuration.

7. The method of Claim 1, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

8. The method of Claim 1, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

9. The method of Claim 1, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification.

10. A computer readable medium having stored thereon computer-executable instructions for performing a method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

generating a second compact hardware identification for the second hardware configuration;
comparing the second compact hardware identification for the second hardware configuration to (i) a first compact hardware identification for the first hardware configuration or (ii) a first verbose hardware identification for the first hardware configuration;
if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds m, a number of required component class matches, loading the software product onto the second computer system; and
if the number of matches is less than m, preventing the software product from being loaded onto the second computer system.

11. The computer readable medium of Claim 10, wherein the second compact hardware identification is generated by a method comprising:

selecting n component classes of the second hardware configuration;
identifying all component instances within each of the n component classes of the second hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first

APPLE EXHIBIT 1102, Page 123 of 1048

hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification for the second hardware configuration.

12. The computer readable medium of Claim 10, wherein the first compact hardware identification is generated by a method comprising:

selecting n component classes of the first hardware configuration;
identifying all component instances within each of the n component classes of the first hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification for the first hardware configuration.

13. The computer readable medium of Claim 10, wherein the first verbose hardware identification is generated by a method comprising:

selecting n component classes of the first hardware configuration;
identifying all component instances within each of the n component classes of the first hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and
concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification for the first hardware configuration.

14. The computer readable medium of Claim 10, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification for the first hardware configuration.

15. The computer readable medium of Claim 10, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification for the first hardware configuration.

16. The computer readable medium of Claim 10, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

17. The computer-readable medium of Claim 10, wherein a match exists between a component class of the second

APPLE EXHIBIT 1102, Page 124 of 1048

hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

18. The computer readable medium of Claim 10, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification.

19. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing a method of determining whether a software product can be used on a second computer system having a second hardware configuration, wherein the second computer system is identical to or different from a first computer system having a first hardware configuration, the software product being initially installed on the first computer, wherein the method comprises:

generating a second compact hardware identification for the second hardware configuration;
comparing the second compact hardware identification for the second hardware configuration to (i) a first compact hardware identification for the first hardware configuration or (ii) a first verbose hardware identification for the first hardware configuration;
if a number of matches exists between component classes of the second hardware configuration and corresponding component classes of the first hardware configuration, and the number of matches equals or exceeds m, a number of required component class matches, loading the software product onto the second computer system; and
if the number of matches is less than m, preventing the software product from being loaded onto the second computer system.

20. The computing system of Claim 19, wherein the second compact hardware identification is generated by a method comprising:

selecting n component classes of the second hardware configuration;
identifying all component instances within each of the n component classes of the second hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the second hardware configuration;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the second hardware configuration; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the second compact hardware identification for the second hardware configuration.

21. The computing system of Claim 19, wherein the first compact hardware identification is generated by a method comprising:

selecting n component classes of the first hardware configuration;
identifying all component instances within each of the n component classes of the first hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes of the first hardware configuration; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the first compact hardware identification for the first hardware configuration.

22. The computing system of Claim 19, wherein the first verbose hardware identification is generated by a method comprising:

selecting n component classes of the first hardware configuration;

APPLE EXHIBIT 1102, Page 125 of 1048

identifying all component instances within each of the n component classes of the first hardware configuration;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes of the first hardware configuration; and
concatenating the plurality of first hash results to form a hash portion, wherein the hash portion forms the first verbose hardware identification for the first hardware configuration.

23. The computing system of Claim 19, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches any one of the second component instance hash results within a corresponding component class of (i) the first compact hardware identification for the first hardware configuration or (ii) derived from the first component instance hashes of the verbose hardware identification for the first hardware configuration.

24. The computing system of Claim 19, wherein a single match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when one second component instance hash result within a component class of the second compact hardware identification for the second hardware configuration matches two or more second component instance hash results derived from the first component instance hashes within a corresponding component class of the first verbose hardware identification for the first hardware configuration.

25. The computing system of Claim 19, wherein no match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration (a) when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does contain a component instance, and (b) when the component class in the first hardware configuration contains a single component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

26. The computing system of Claim 19, wherein a match exists between a component class of the second hardware configuration and a corresponding component class of the first hardware configuration when the component class in the first hardware configuration does not contain a component instance, and the corresponding component class in the second hardware configuration does not contain a component instance.

27. The computing system of Claim 19, wherein m equals (n - 3), wherein n represents the number of component classes within the second hardware configuration used to form the second compact hardware identification.

28. A method of generating a compact hardware identification for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;
identifying all component instances within each of the n component classes;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification for the first computer system.

29. A computer readable medium having stored thereon computer-executable instructions for performing a method of generating a compact hardware identification for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;
identifying all component instances within each of the n component classes;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the

APPLE EXHIBIT 1102, Page 126 of 1048

n component classes;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification for the first computer system.

30. A computing system containing at least one application module usable on the computing system, wherein the at least one application module comprises application code for performing a method of generating a compact hardware identification for a first computer system having a first hardware configuration, wherein the method comprises:

selecting n component classes;
identifying all component instances within each of the n component classes;
generating a plurality of component instance first hash results, wherein the plurality of component instance first hash results comprises a first hash result for each component instance and at least one first hash result for the n component classes;
generating a plurality of component instance second hash results, wherein the plurality of component instance second hash results comprises second hash results resulting from performing a second hash function on select component instance first hash results within one or more select component classes; and
concatenating the plurality of second hash results to form a hash portion, wherein the hash portion forms the compact hardware identification for the first computer system.

APPLE EXHIBIT 1102, Page 127 of 1048

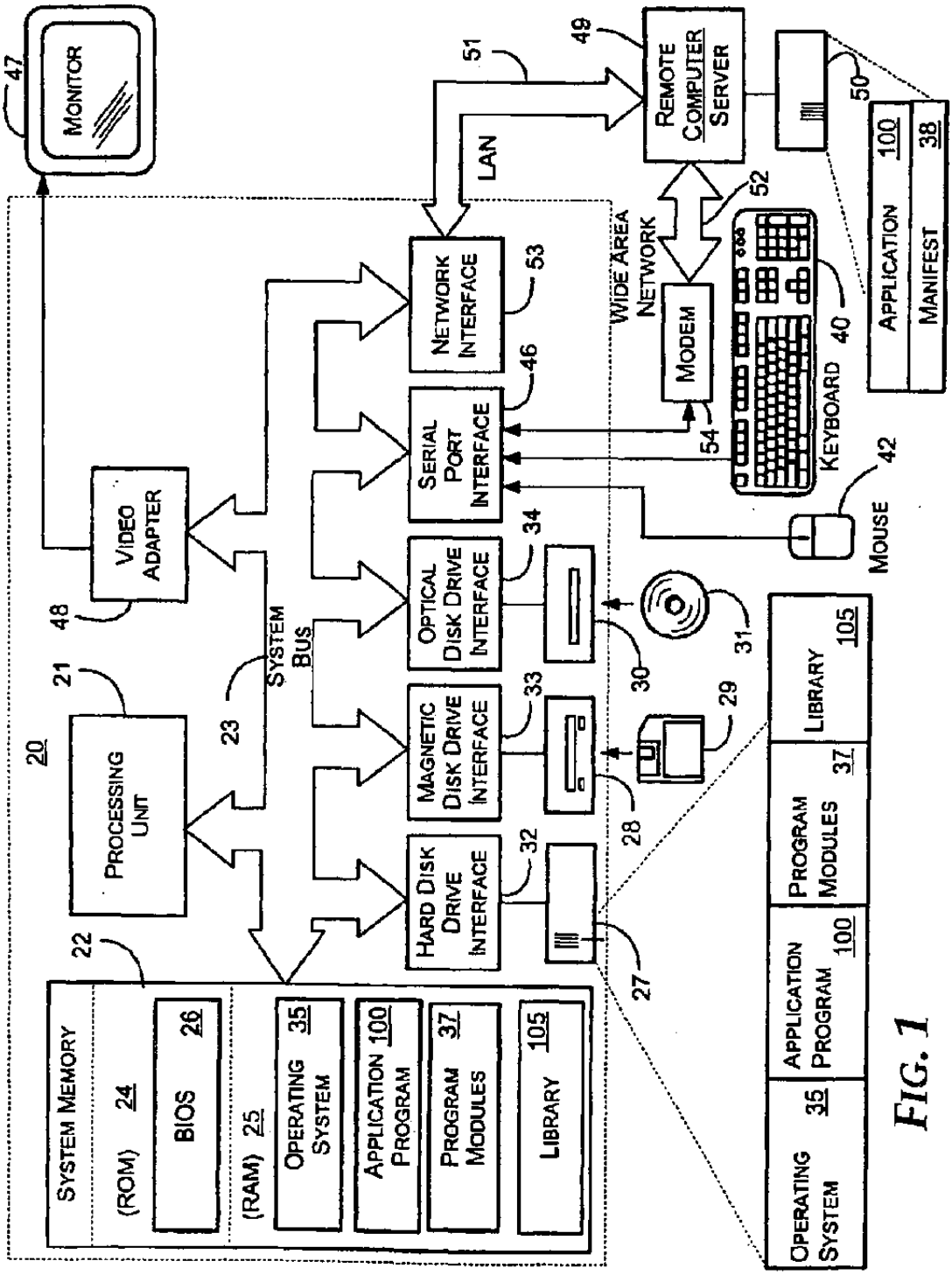


FIG. 1

APPLE EXHIBIT 1102, Page 128 of 1048

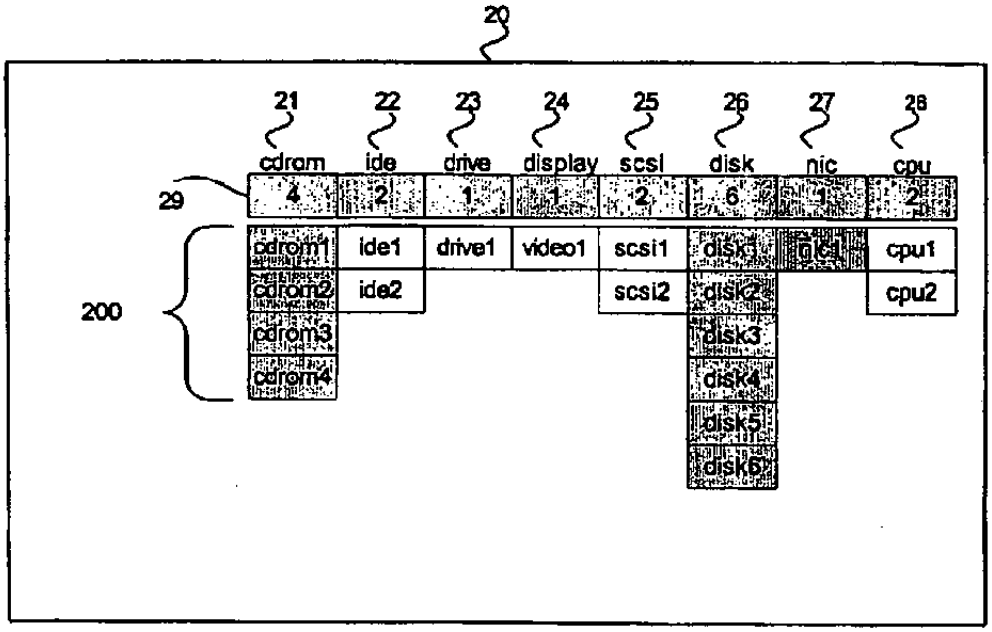


Fig. 2

APPLE EXHIBIT 1102, Page 129 of 1048

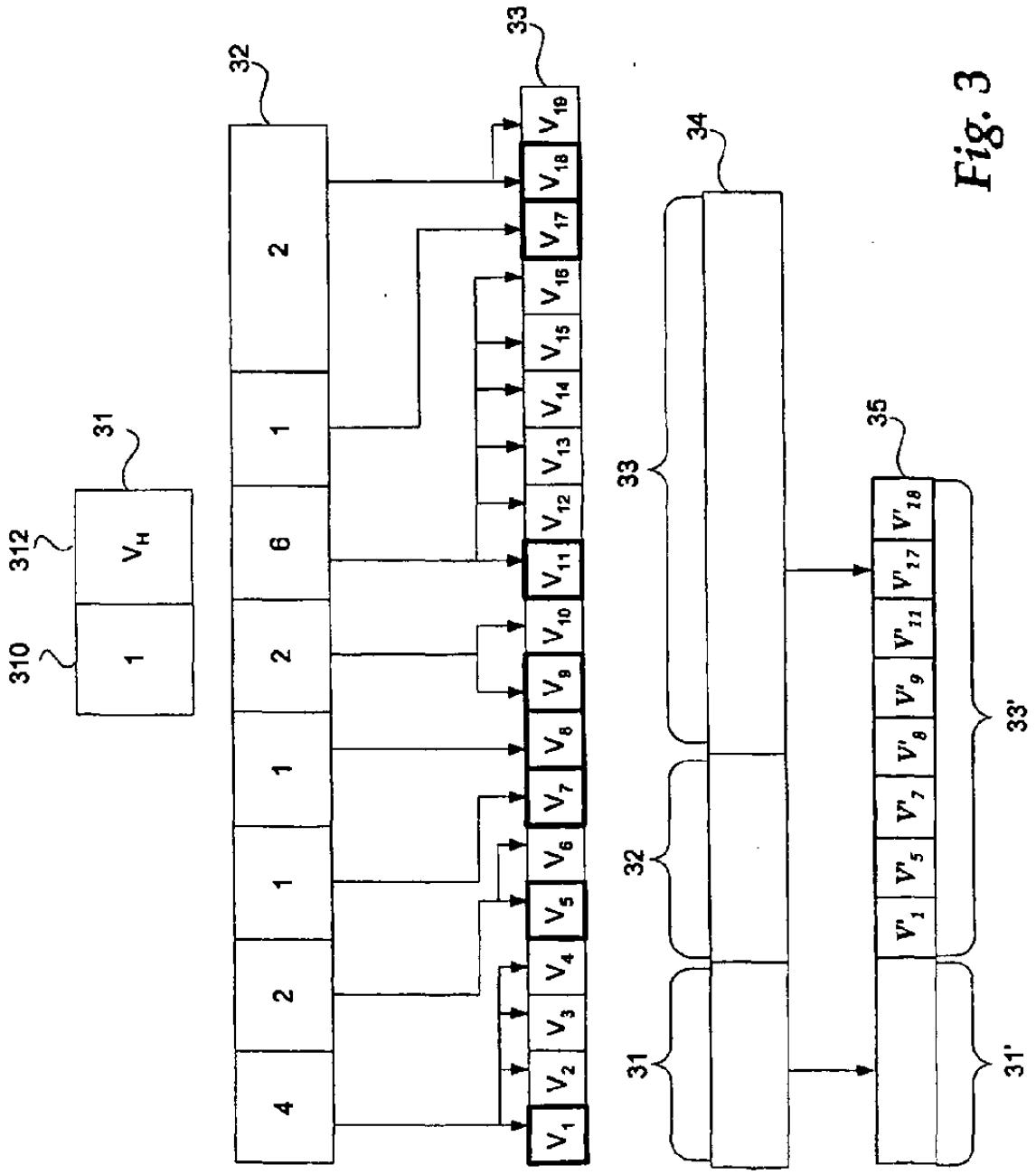


Fig. 3

APPLE EXHIBIT 1102, Page 130 of 1048

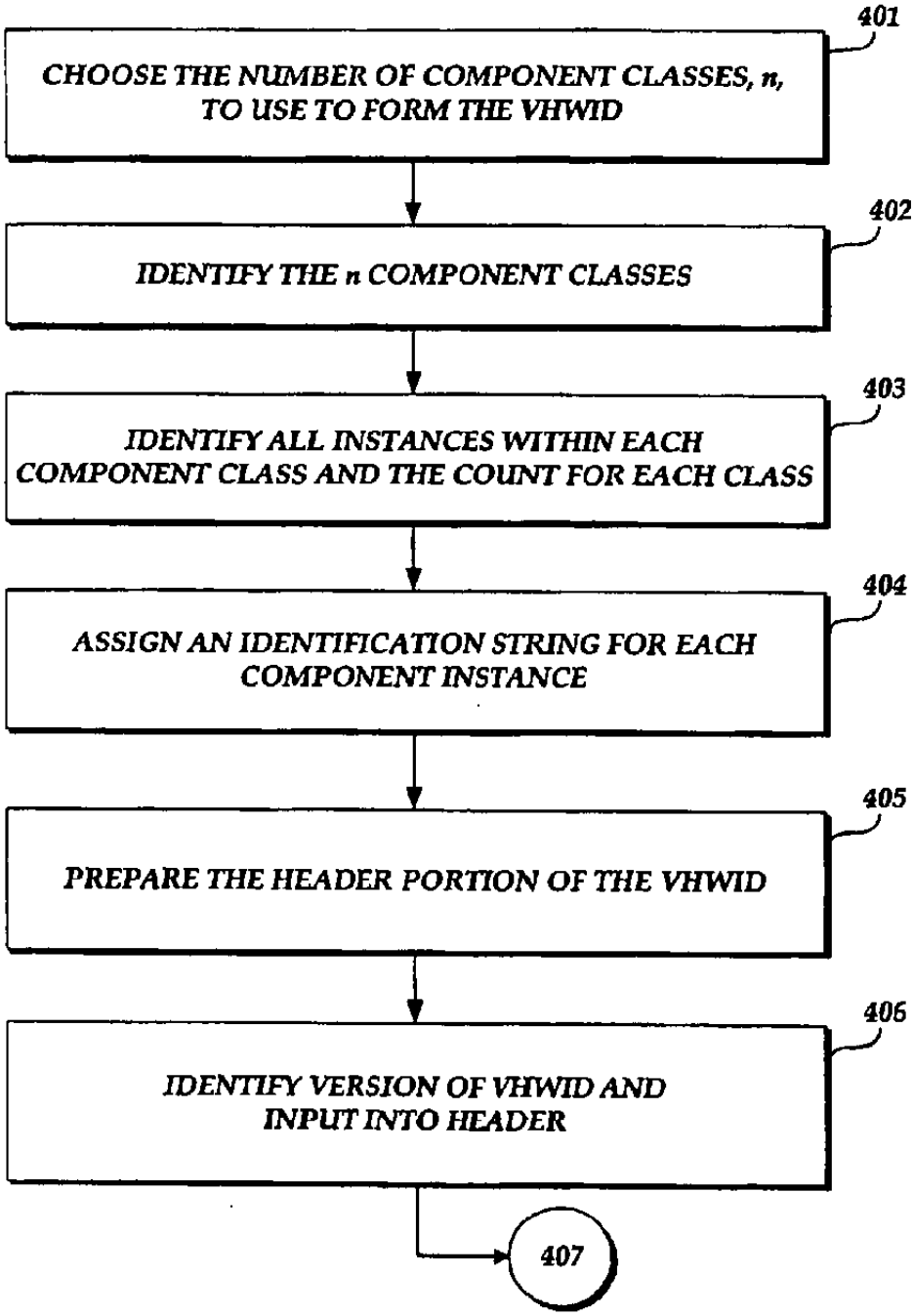


Fig. 4

APPLE EXHIBIT 1102, Page 131 of 1048

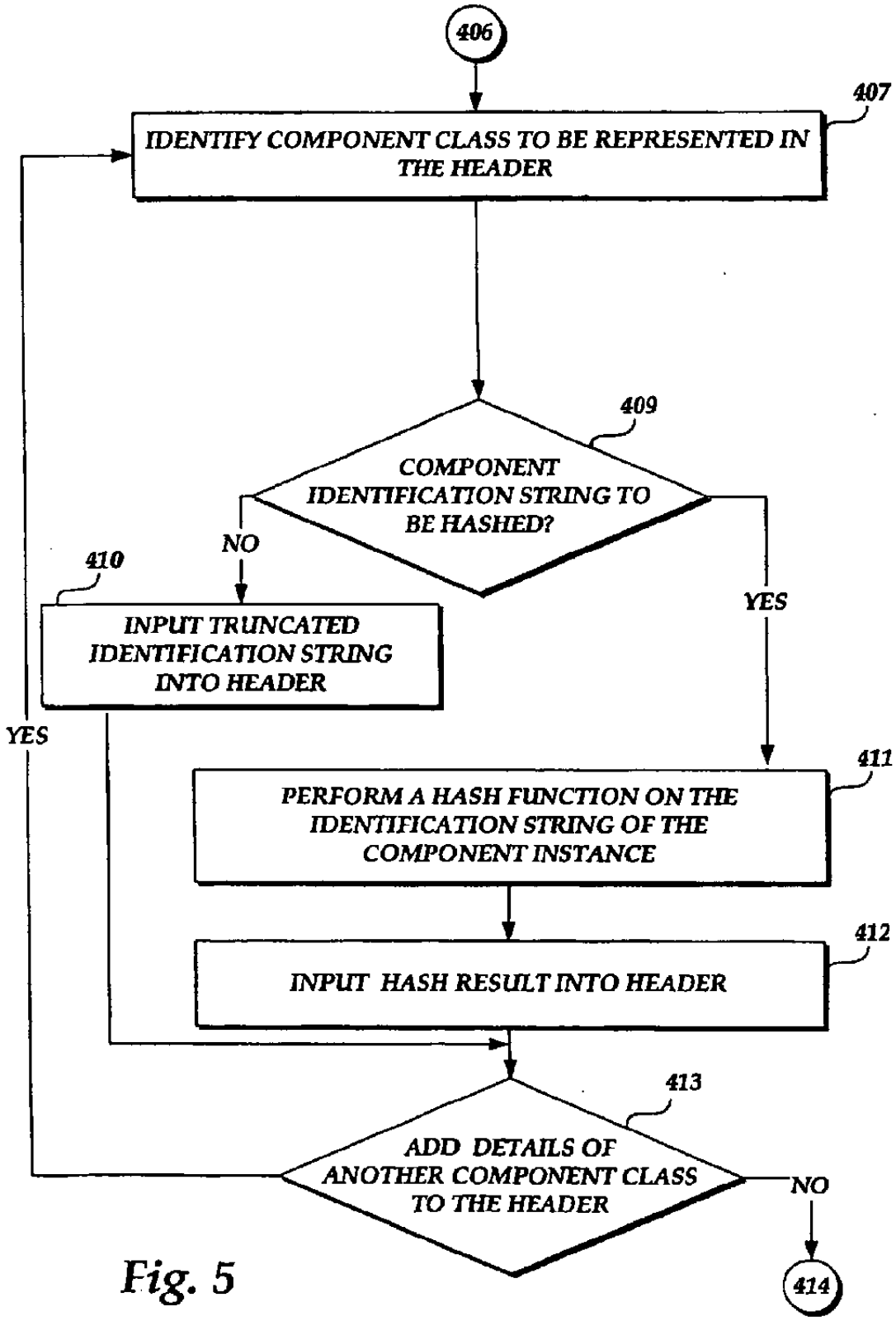


Fig. 5

APPLE EXHIBIT 1102, Page 132 of 1048

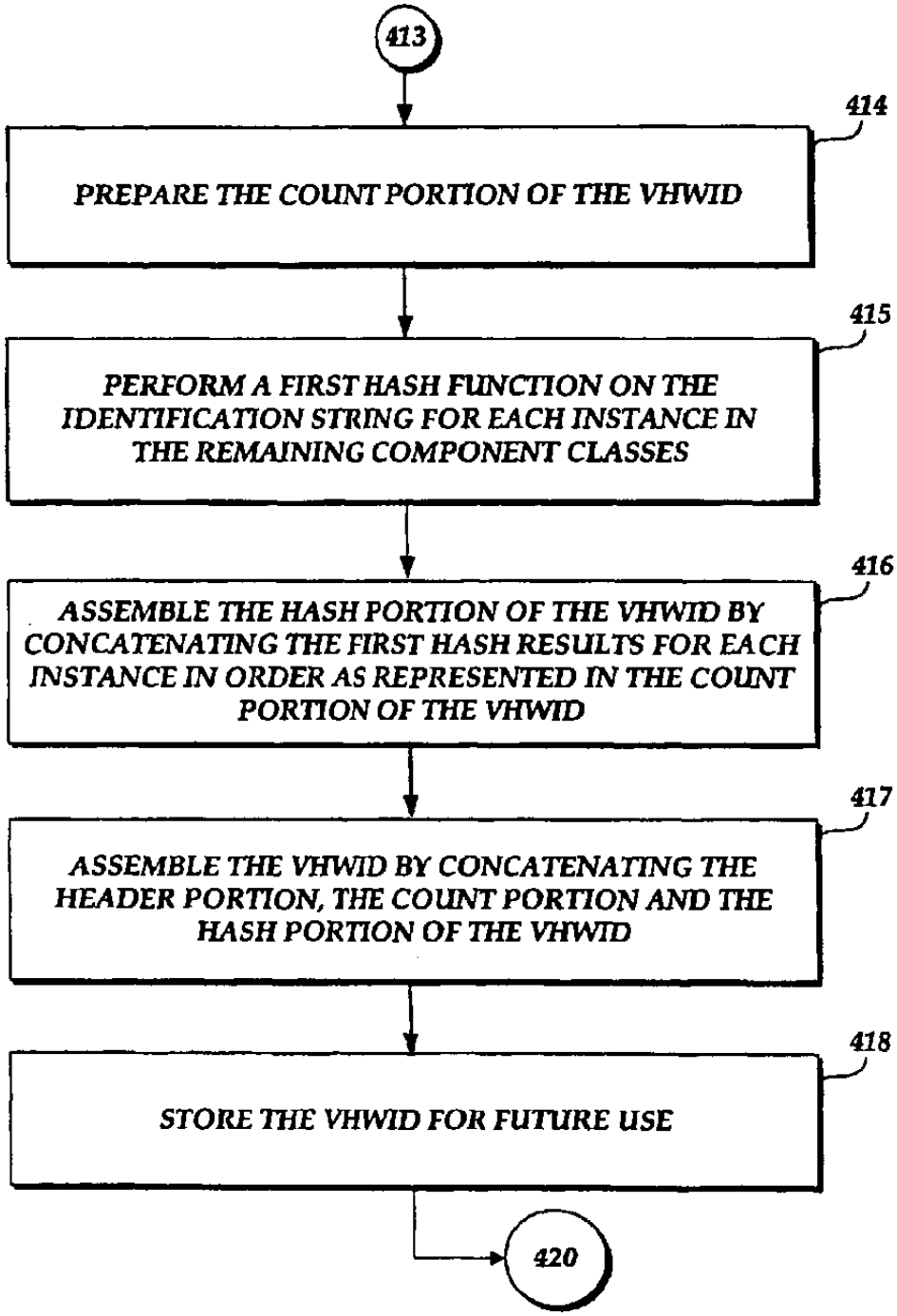


Fig. 6

APPLE EXHIBIT 1102, Page 133 of 1048

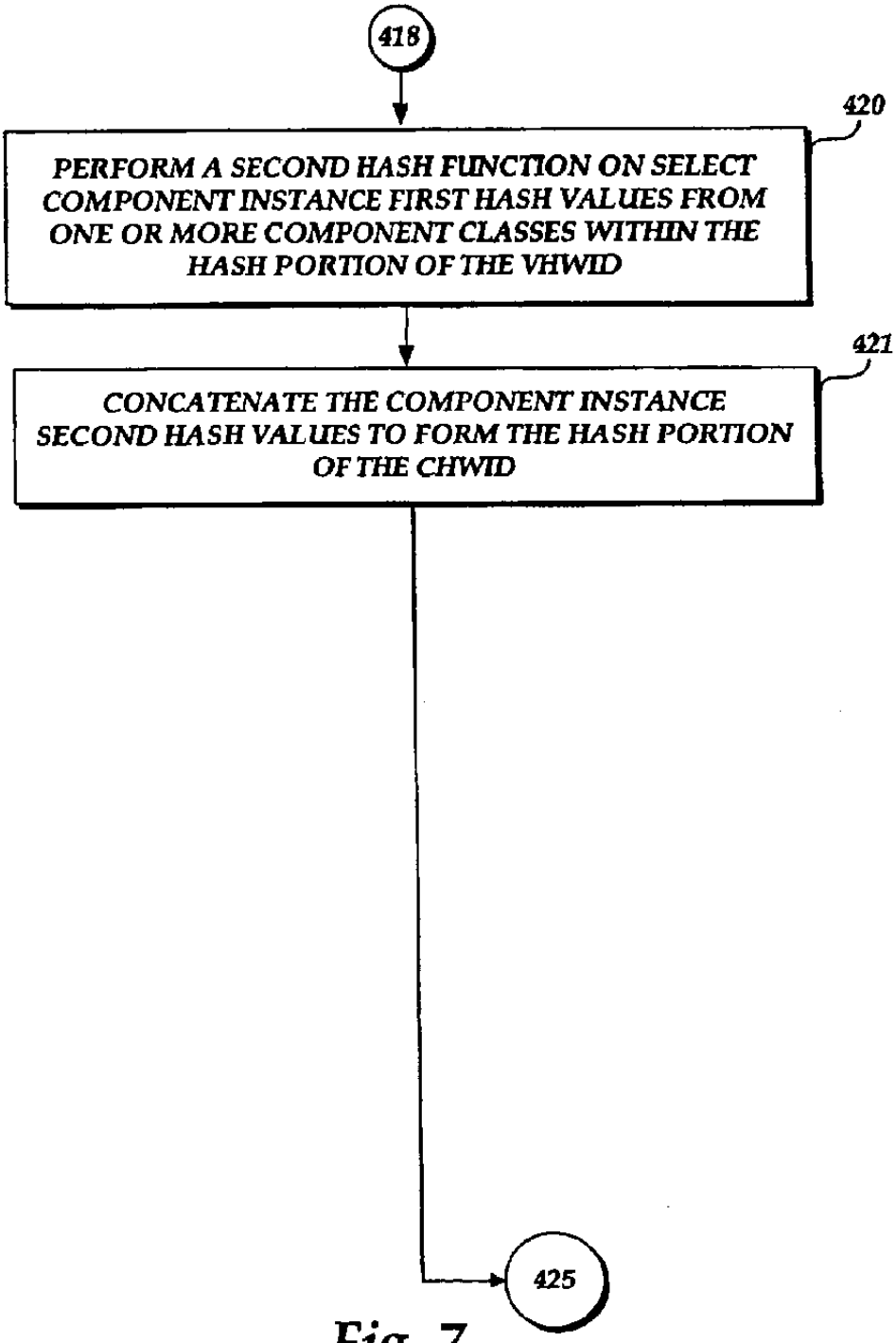


Fig. 7

APPLE EXHIBIT 1102, Page 134 of 1048

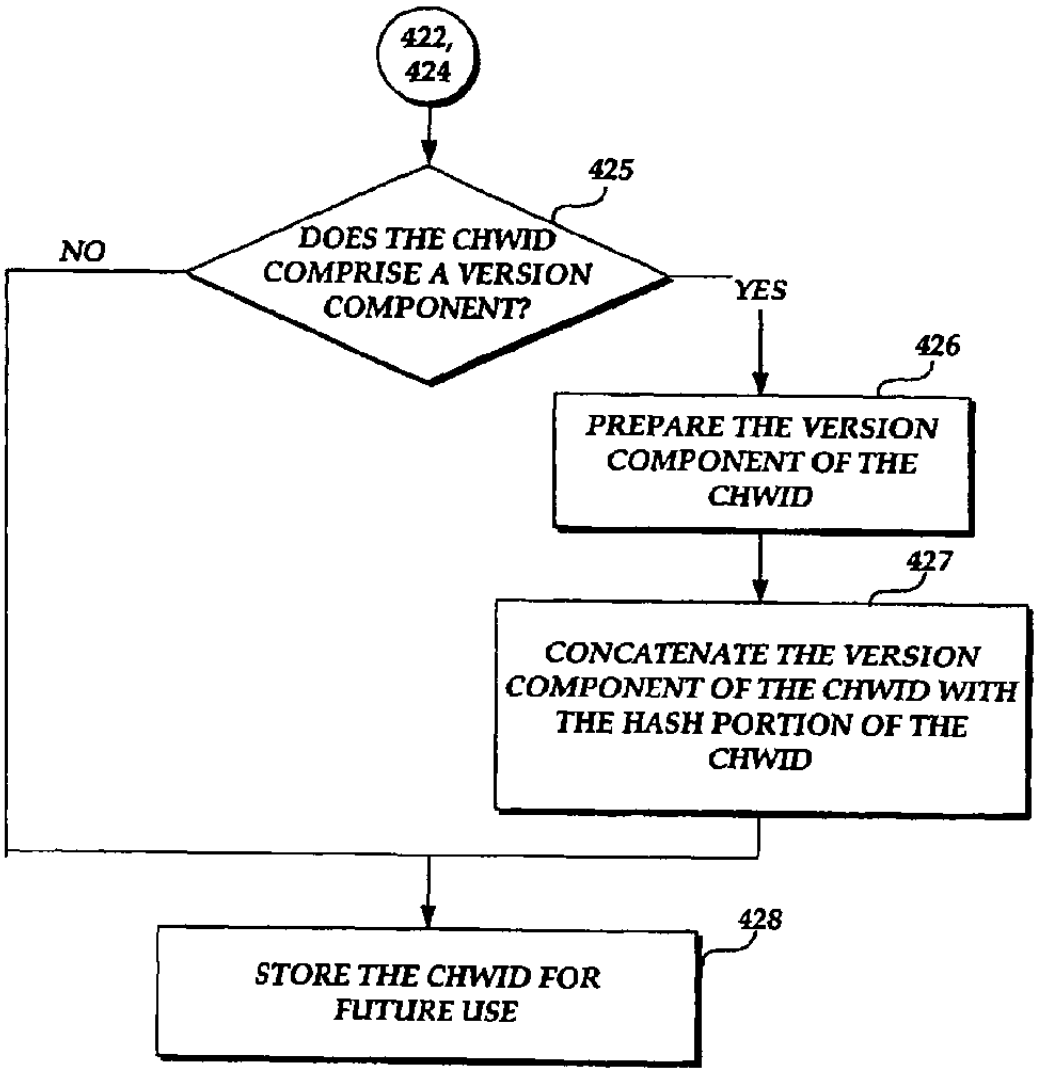


Fig. 8

APPLE EXHIBIT 1102, Page 135 of 1048

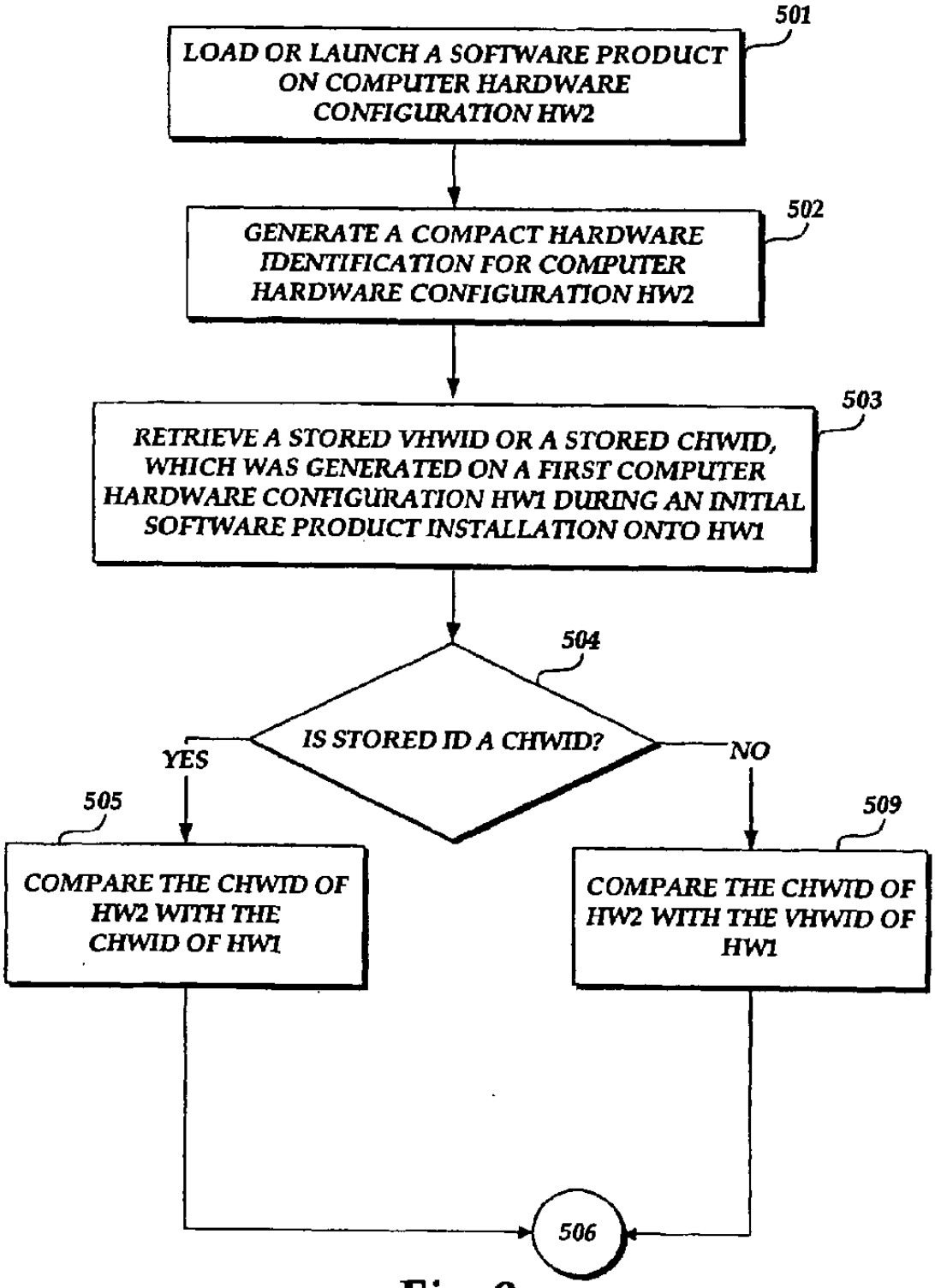


Fig. 9

APPLE EXHIBIT 1102, Page 136 of 1048

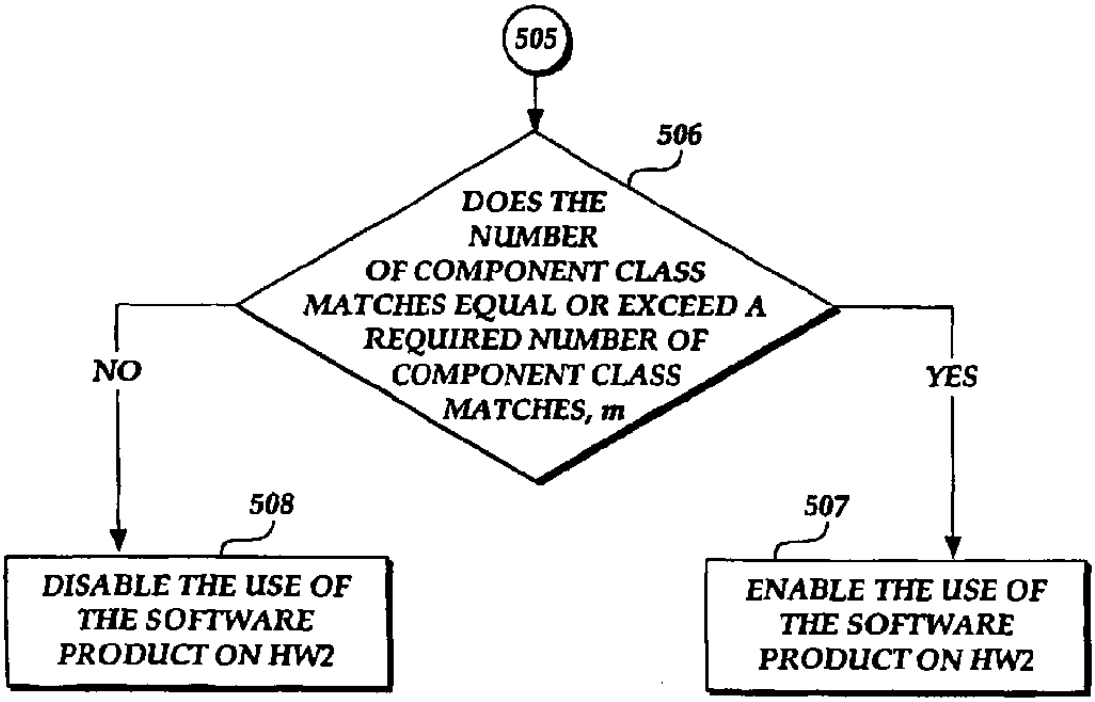


Fig. 10

APPLE EXHIBIT 1102, Page 137 of 1048

(19) Europäisches Patentamt
European Patent Office
Office européen dea brevets

(11) EP 1 637 961 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
22.03.2006 Bulletin 2006/12

(51) Int Cl.:
G06F 1/00 *(2006.01)* ***G06F 9/445*** *(2006.01)*
H04L 29/06 *(2006.01)*

(21) Application number: **05108154.5**

(22) Date of filing: **06.09.2005**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK YU

(30) Priority: **15.09.2004 US 941594**

(71) Applicant: **Microsoft Corporation**
Redmond WA 98052 (US)

(72) Inventors:
- **Holladay, Martin L.**
  98052, Redmond (US)
- **Karki, Mukesh**
  98052, Redmond (US)
- **Parthasarathy, Narayanan**
  98052, Redmond (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser
Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) Deploying and receiving software over a network susceptible to malicious communication

(57) Systems and/or methods that enable secure deployment and/or receipt of an operating system and updates for the operating system to a bare computer across a network susceptible to malicious communication are described. These systems and/or methods can, in one embodiment, securely deploy an image having an operating system and enable secure receipt of an update for the operating system, both via a network susceptible to malicious communication. They can also, in another embodiment, enable a bare computer added to a network to have an operating system deployed to it and updated via the network before the bare computer is subjected to malicious code communicated over the network.

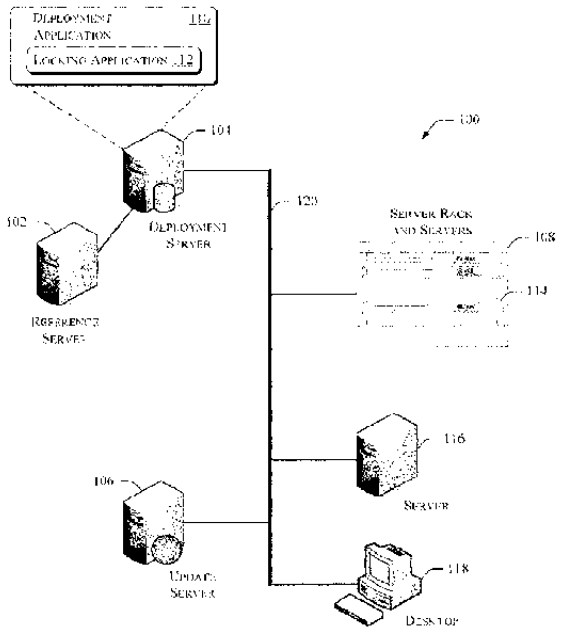


Fig. 1

EP 1 637 961 A2

Description

TECHNICAL FIELD

[0001] This invention relates to deploying and receiving software over a network.

BACKGROUND

[0002] One of the quickest and easiest ways to add a new, bare server (a server not having an operating system) to a network is to plug it into the network and use a deployment server on the network to deploy an image of the operating system to the bare server. The bare server can save this image to its hard disk drive or equivalent storage and then reboot. Once it reboots, it can be running with the newly deployed operating system.

[0003] Operating systems deployed to bare servers with an image are often out of date, however; they need current updates to be optimally secure. A server with an out-of-date operating system, if it is linked to the network, can acquire these updates through the network, usually from an Internet site or an intranet server having current updates.

[0004] But the network, even if it is an intranet, may be susceptible to malicious communication, such as a virus or other network-based attack. Because of this, the server often cannot acquire these updates before being attacked by malicious code via the network. In the amount of time between when the server is first running with its operating system on the network and when it has downloaded and installed current updates, malicious code like a virus or Trojan horse can attack the server. This is a real danger, as many malicious programs take less than a second to corrupt a server running an out-of-date operating system. The MS Blaster virus, for instance, can corrupt a server without an appropriate software update within tenths of a second.

[0005] To partially combat this problem, a bare server can be connected to a deployment server without being connected to a network, such as by manually plugging a cable into both servers. Through this cable, the deployment server can deploy an image having an operating system to the bare server. The server can then be rebooted with the operating system. Once this is done, updates can be installed, usually by hand with compact disks, to make the operating system optimally secure. Once updated, the server can then be plugged into the network. This partial solution may reduce the server's vulnerability to attack, but it is time consuming. An information technology specialist can spend many hours connecting bare servers directly to a deployment server, deploying images, installing updates, disconnecting the servers from the deployment server, and then connecting them to the network.

[0006] Also to partially combat this problem, the operating system and updates can be manually installed on a bare server, usually with many compact disks, prior to connecting the server to the network. Manually installing an operating system and updates, however, is also time consuming and tedious; it can takes hours for each server.

[0007] There is, therefore, a need for a secure way to deploy an operating system and updates to a server over a network that is susceptible to malicious communication.

SUMMARY

[0008] Systems and/or methods ("tools") that enable secure deployment and/or receipt of an operating system and updates for the operating system to a bare computer across a network are described. In one embodiment, for instance, the tools securely deploy an image having an operating system and enable secure receipt of an update for the operating system, both via a network that is susceptible to malicious communication. In a second embodiment, for example, the tools deploy to a computer across a network an operating system that, when run by the computer, prohibits the computer from receiving malicious and/or unsolicited communications via the network. In a third embodiment, for instance, the tools enable a bare computer added to a network to have an operating system deployed to it and updated via the network before the bare computer is subjected to malicious code communicated over the network.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009]

Fig. 1 illustrates an exemplary architecture having exemplary servers, a network susceptible to malicious communication, and bare computers.

Fig. 2 sets forth a flow diagram of an exemplary process for creating a locked image having an operating system.

Fig. 3 sets forth a flow diagram of an exemplary process for deploying and receiving a locked image and updates via a network susceptible to malicious communication.

[0010] The same numbers are used throughout the disclosure and figures to reference like components and features.

DETAILED DESCRIPTION

An Exemplary Architecture

[0011] Referring to Figure 1, an exemplary architecture 100 is shown having a reference server 102, a deployment server 104, an update server 106, and a server rack 108. The reference server, deployment server, and update server are shown as three separate servers, though they can be combined into one or more servers in any combination. The deployment server comprises

APPLE EXHIBIT 1102, Page 139 of 1048

computer-readable media capable of performing one or more of the processes described below. These media can comprise a deployment application 110 and a locking application 112, for instance. The locking application is shown as part of the deployment application, though each can be separate or combined. The update server also comprises computer-readable media, here capable of deploying software patches, fixes, and the like, such as to update an out-of-date operating system for improving its operation, e.g., its security capabilities.

[0012] Three exemplary bare computers are also shown, a bare server 114 in rack 108, a bare stand-alone server 116, and a bare desktop 118. Each of the bare computers has a software or hardware application sufficient to enable the bare computer to request, receive, and follow basic instructions, such as from the deployment application 110.

[0013] The architecture 100 communicates across a network 120. The network is a communication network susceptible to malicious communication, such as network-based attacks. This network can comprise an intranet in communication with an insecure source, such as the Internet or a corrupted computer within the intranet capable of sending malicious code across the network.

Building a Locked Image

[0014] Referring to Figure 2, an exemplary process 200 for building a locked image is shown. This process is illustrated as a series of blocks representing individual operations or acts performed by deployment server 104, such as with locking application 112. This and other processes described herein may be implemented in any suitable hardware, software, firmware, or combination thereof. In the case of software and firmware, these processes represent sets of operations implemented as computer-executable instructions.

[0015] At block 202, deployment server 104, using locking application 112, instructs reference server 102 to prohibit communications with untrustworthy sources but permit communication with at least one trustworthy source, such as the deployment server. The prohibited communications can comprise all communications that are not solicited by the reference server or all communications, solicited or not (other than those permitted from the trustworthy source).

[0016] In one embodiment, the locking application selectively prohibits communication by instructing the reference server to enable a firewall prohibiting communication with any port other than the port used by the deployment server. In another embodiment, the locking application does so by instructing the reference server to enable one or more protocols, such as IPSec ("Internet Protocol Security"), which can prohibit communication with any computer other than the deployment server (and, in some cases, update server 106). In both embodiments, the reference server is instructed to alter its settings to operate securely but permit communication with at least one trustworthy source.

[0017] These settings are stored in the memory of the reference server. Because of this, an image of the reference server's memory can comprise the operating system and these settings. A bare computer booting up this image can run the operating system having these settings, thereby prohibiting potentially dangerous communications but permitting communication with a trustworthy source. If the bare computer that is to receive the image is a desktop or other non-server computer, the reference server can be a reference desktop or other non-server reference computer.

[0018] At block 204, deployment server 104 receives an image having an operating system. In one embodiment, the deployment server performs blocks 204 and 206 and in another embodiment performs blocks 202 and 204, as set described below. This image can be received from the reference server of Figure 1 or another reference computer (not shown). If the image is locked, such as resulting from the actions of block 202, the deployment server does not proceed to block 206. If the image is not locked, the deployment server proceeds to block 206. In another embodiment, the deployment server waits to lock the image until after the image has been saved to the bare server but before the bare server reboots (not shown).

[0019] At block 206, the deployment server, through locking application 112, edits an image having an operating system. This editing can comprise locking the image by altering a security setting to prohibit unsolicited communications except from at least one trustworthy source, such as deployment server 104. The prohibited communications can comprise all communications that are not solicited by the computer running the operating system or all communications, solicited or not (other than those permitted from the trustworthy source). The locking application can do so by editing the image's security setting(s) to add or turn on a firewall like the firewall described in block 202. The locking application can also do so, for instance, by editing the image's security setting (s) to comprise IPSec protocols, such as those described in block 202. Thus, the locking application locks the image to prohibit potentially dangerous communications by a computer running the software in the image but permit communication with a trustworthy source.

Deploying a Locked Image and Updating an Operating System

[0020] Referring to Figure 3, an exemplary process 300 for securely deploying, via a network susceptible to malicious communication, an image having an operating system and enabling secure receipt of an update for the operating system is shown. This process is illustrated as a series of blocks representing individual operations or acts performed by deployment server 104, such as with deploying application 110. An exemplary process 302 for securely receiving the locked image and updates to the

APPLE EXHIBIT 1102, Page 140 of 1048

operating system is also shown. Process 302 is illustrated as a series of blocks representing operations or acts performed by or to bare server 114.

[0021] At block 304, a bare computer is connected to network 120. In the ongoing embodiment, bare server 114 is plugged into the network via rack 108, though other bare computers can instead be connected to the network, such as stand-alone server 116 or desktop 118.

[0022] At block 306, the bare server communicates across the network, requesting an operating system. Without an operating system, the bare server often is not yet vulnerable to malicious code on the network.

[0023] At block 308, deployment server 104 receives the request for an operating system. At block 310, the deployment server, through deployment application 110, securely deplays a locked image having an operating system to the bare server. At this block, the deployment server can, in some embodiments, also deploy software updates. The locked image can be the result of the process 200. In the ongoing embodiment, the locked image is one that, when run by the bare server (which will then no longer be bare), will not permit receipt of unsolicited communication from any source other than the deployment server or any port other than the port used by the deployment server.

[0024] At block 312, the bare server securely receives the locked image via the network and saves it to memory. By securely receiving the locked image, the bare server can receive the locked image without its being subject to malicious communication during transmission. Secure communication of this locked image can also prohibit it from being intercepted or monitored by a third party. In one embodiment, the bare server also receives updates with or as part of the locked image. At block 314, the bare server communicates that it has received the locked image. At block 316, the deployment server receives the communication from the bare server indicating that it has received the locked image. At block 318, the deployment server, through the deployment application, instructs the bare server to boot the locked image.

[0025] At block 320, the bare server reboots, thereby running the image with the operating system and its secure settings. The bare server, now no longer bare as it has an operating system, is running in a secure mode. The bare server, because of settings and/or software in the image, can prohibit untrustworthy or potentially malicious communications. The bare server can operate securely even though it is connected to network 120 and potentially is operating with an out-of-date operating system that could otherwise be vulnerable to malicious communication sent over the network.

[0026] At block 322, bare server 114 informs the deployment server that the operating system is running and/or that the boot was successful.

[0027] At block 324, deployment server 104 receives this information. At block 326, the deployment server, through deployment application 110, instructs the bare server to securely receive and/or install updates. In the ongoing embodiment, the deployment server instructs the bare server to initiate communication with update server 106. In another embodiment, the deployment server securely sends updates to the bare server's operating system and instructs it to add these updates without use of a separate update source like the update server. In still another embodiment, the updates are received along with or as part of the image received at block 312 and sent at block 310. In this embodiment, the deployment server instructs the bare server to install the already received updates. The updates received in any of these embodiments can be effective to update the operating system or other software on the bare server, and can comprise software patches, fixes, and the like. These updates can improve resistance to various malicious code later received by the bare server, described in greater detail below.

[0028] At block 328, the bare server receives the instruction to securely receive updates. In the ongoing embodiment, the bare server receives the instruction from the deployment server.

[0029] At block 330, the bare server initiates secure communication to securely receive updates. In the ongoing embodiment, the bare server solicits communication from update server 106. The bare server's security settings are configured to prevent receipt of unsolicited communication, but the bare server is permitted to solicit communication from the update server. By so doing, updates and other information from the solicited update server can be received by the bare server running the operating system. Other, unsolicited information, can be refused by the bare server because of its security settings, thereby protecting the bare server from unsolicited. malicious code while enabling the bare server to receive updates.

[0030] At block 332, the bare server securely receives and applies updates to its operating system. These updates can be received via the network from the update server solicited at block 330 or from the deployment server directly, for instance. This secure receipt of updates enables the bare server to have an updated operating system via a network that is susceptible to malicious communication without first being vulnerable to malicious code communicated over the network.

[0031] At block 334, the bare server communicates that it has updated its operating system. At block 336, the deployment server receives this communication.

[0032] At block 338, the deployment server instructs the bare server to commence potentially malicious communication. Because the operating system is updated, the bare server is better capable of defending itself against malicious code and attacks communicated across the network. In one embodiment, the deployment server sends and/or instructs the bare server to install a firewall or IPSec protocols to further secure the bare server's operations before commencing potentially malicious communication.

[0033] At block 340, the bare server commences po-

APPLE EXHIBIT 1102, Page 141 of 1048

tentially malicious communication over the network, such as by commencing a production mode of operation. The bare server can do so by opening particular ports, for instance. If the bare server is to be a webserver, for instance, it can open port 80 to enable it to communicate with other servers across the Internet.

[0034] In the ongoing embodiment, most if not all of the acts of the deployment server and the deployment application can be performed automatically and without user interaction. This enables a user to connect a bare server or other bare computer to a network and, without further interaction, have the bare server operating with an updated operating system without having to subject the bare server to malicious code via the network before the operating system is updated.

Conclusion

[0035] The above-described tools enable secure deployment and/or receipt of an operating system and updates across a network that can be susceptible to malicious communication. Although the invention has been described in language specific to structural features and/or methodological acts, it is to be understood that the invention defined in the appended claims is not necessarily limited to the specific features or acts described. Rather, the specific features and acts are disclosed as exemplary forms of implementing the claimed invention.

Claims

1. A method comprising:

   receiving a locked image having an operating system and security settings, the security settings effective to prohibit unsolicited communication via a network that is susceptible to malicious communication other than from a secure source or via a secure port; and
   securely deploying the locked image to a bare computer via the network.

2. The method of claim 1, further comprising:

   instructing the bare computer to securely receive a software update.

3. The method of claim 2, wherein the software update is capable of improving the operating system's resistance to malicious code.

4. The method of claim 2, further comprising receiving an indication that the software update has been applied and instructing the bare computer to commence potentially malicious communication via the network.

5. The method of claim 1, further comprising instructing a reference server having the operating system to prohibit unsolicited communication via the network other than from the secure source or via the secure port.

6. The method of claim 5, wherein the act of instructing the reference server comprises instructing the reference server to enable a firewall.

7. The method of claim 1, wherein the bare computer comprises a bare server.

8. One or more computer-readable media having computer-readable instructions for performing the method recited in claim 1.

9. A system comprising means for performing the method recited in claim 1.

10. A method comprising:

    receiving an image having an operating system and a security setting via a network susceptible to malicious communication, the security setting effective to prohibit unsolicited and potentially malicious communication via the network;
    booting the image, effective to run the operating system at the security setting;
    receiving an update to the operating system; and
    applying the update to the operating system.

11. The method of claim 10, wherein the security setting is effective to permit unsolicited and secure communication from a secure source or via a secure port.

12. The method of claim 11, wherein the act of receiving the update is via the network.

13. The method of claim 10, wherein the act of receiving the update comprises receiving an instruction to solicit communication from an update source and soliciting the update source for the update.

14. The method of claim 10, further comprising:

    permitting unsolicited and potentially malicious communication via the network.

15. The method of claim 14, wherein the act of permitting comprises altering the security setting to permit unsolicited and potentially malicious communication.

16. The method of claim 10, wherein the acts of receiving the image, booting the image, receiving the update, and applying the update are performed without user interaction.

APPLE EXHIBIT 1102, Page 142 of 1048

17. A system comprising means for performing the method recited in claim 10.

18. A method comprising:

securely deploying a locked image to a computer over a network susceptible to malicious communication;
instructing the computer to boot the locked image;
instructing the computer to solicit communication to receive a software update;
receiving from the computer an indication that the software update has been received; and
instructing the computer to permit potentially malicious communication over the network.

19. The method of claim 18, further comprising instructing a reference server to prohibit unsolicited communication via the network other than from a secure source or via a secure port and receiving the locked image from the reference server.

20. The method of claim 19, wherein the act of instructing the reference server comprises instructing the reference server to enable a firewall.

21. The method of claim 19, wherein the act of instructing the reference server comprises instructing the reference server to add IPSec protocols.

22. The method of claim 18, wherein the locked image is capable of prohibiting communication sent across the network that is unsolicited and potentially malicious.

23. The method of claim 18, wherein the locked image is capable of prohibiting unsolicited communication other than from a source from which the locked image was deployed.

24. The method of claim 18, wherein the software update is effective to improve an operating system's resistance to malicious code.

25. The method of claim 18, wherein the computer comprises a bare server.

26. The method of claim 18, wherein the act of instructing the computer to solicit communication comprises instructing the computer to solicit communication from an update server over the network.

27. The method of claim 18, wherein the indication indicates that the software update has been successfully applied.

28. The method of claim 18, wherein the act of receiving and the acts of instructing are communicated via the network.

29. The method of claim 18, wherein the network comprises an intranet capable of communicating with the Internet.

30. One or more computer-readable media having computer-readable instructions for performing the method recited in claim 18.

31. A system comprising means for performing the method recited in claim 18.

32. A method comprising:

securely receiving a locked image having an operating system via a network susceptible to malicious communication;
booting the locked image, the locked image having security settings effective to prohibit unsolicited communication other than from one or more secure sources or via one or more secure ports;
receiving instruction from the secure source(s) or via the secure port(s);
following the instruction to securely receive a software update via the network;
applying the software update effective to improve the security of the operating system; and
permitting potentially malicious communication via the network.

33. The method of claim 32, wherein the locked image and the instruction are received from a deployment server via the network.

34. The method of claim 32, wherein at least four of the acts of securely receiving, booting, receiving instruction, following the instruction, applying, and permitting are performed without human interaction.

35. A system comprising means for performing the method recited in claim 32.

APPLE EXHIBIT 1102, Page 143 of 1048

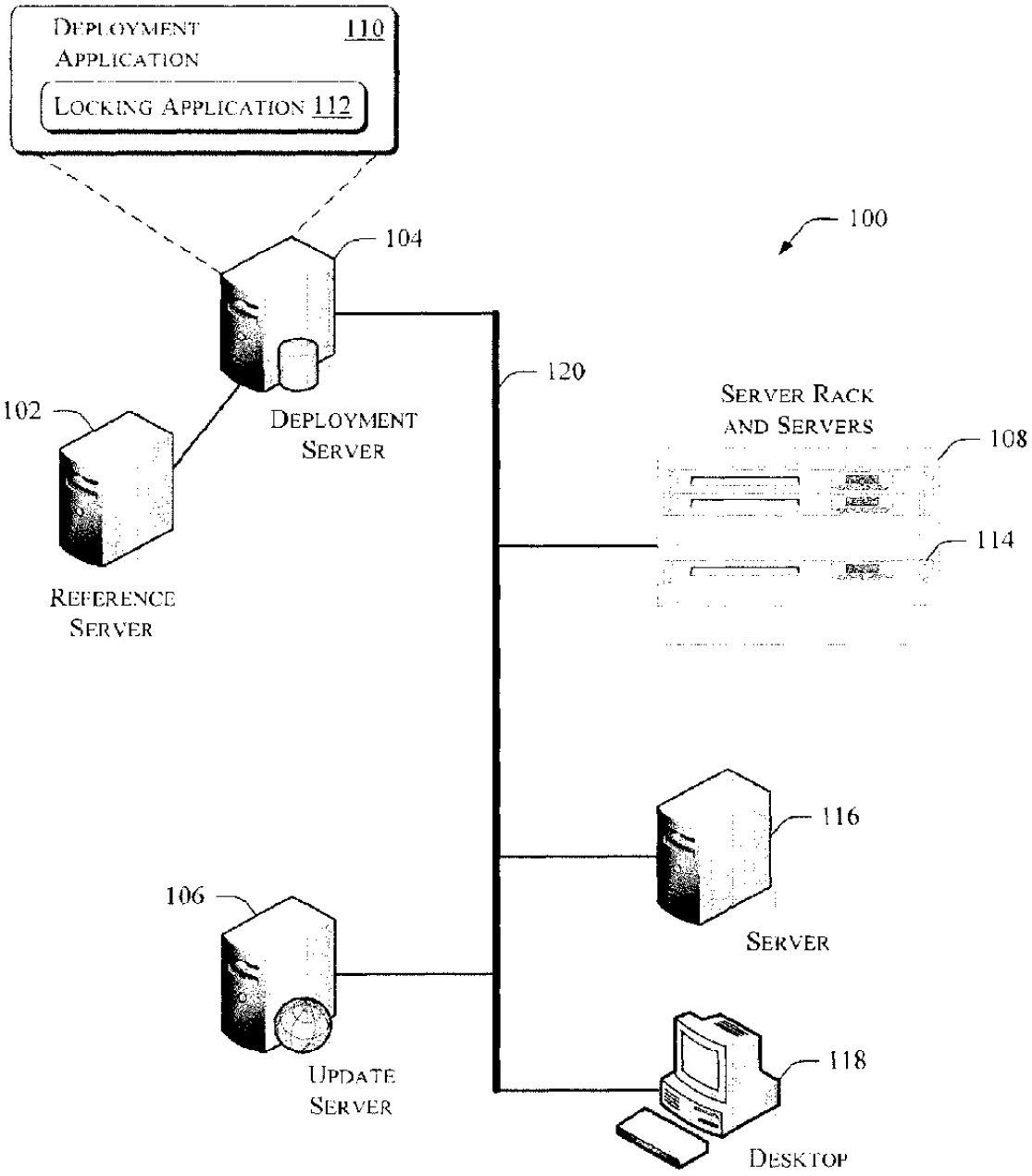


Fig. 1

APPLE EXHIBIT 1102, Page 144 of 1048

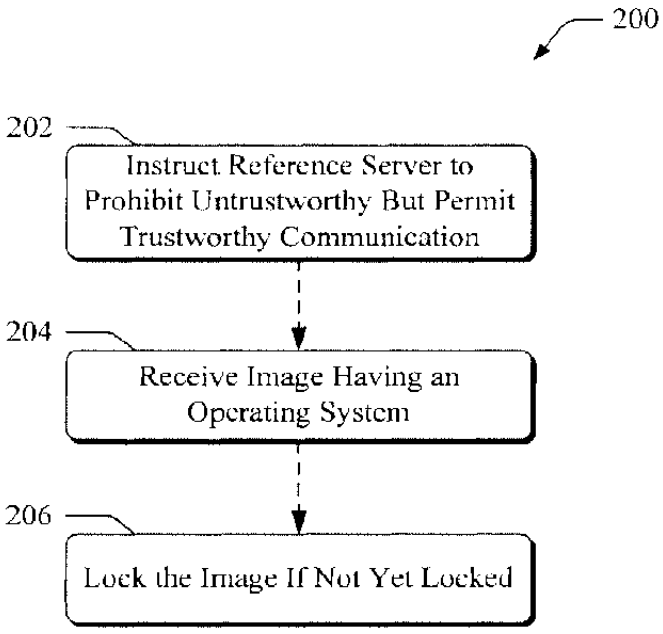


Fig. 2

APPLE EXHIBIT 1102, Page 145 of 1048

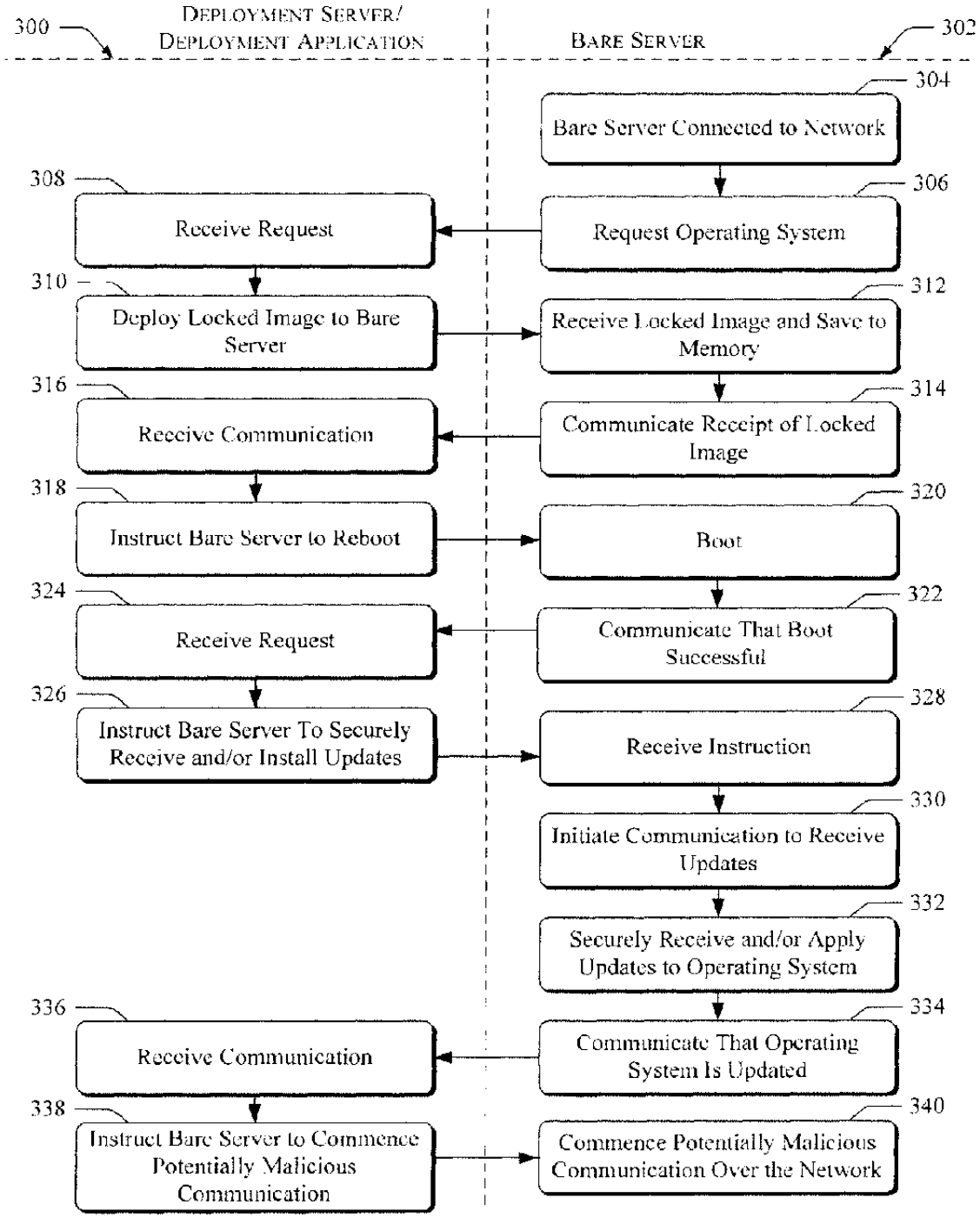


Fig. 3

APPLE EXHIBIT 1102, Page 146 of 1048

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 670 188 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.06.2006 Bulletin 2006/24

(51) Int Cl.:
H04L 12/46 *(2006.01)* ***H04L 29/06*** *(2006.01)*

(21) Application number: **05301029.4**

(22) Date of filing: **08.12.2005**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK YU

(30) Priority: **10.12.2004 US 009917**

(71) Applicant: **Alcatel**
75008 Paris (FR)

(72) Inventors:
- **Krstulich, Zlatko**
  Ontario K2A 2V4, Ottawa (CA)
- **Lee, Cheng-Yin**
  Ontario K2B 6A5, Ottswa (CA)

(74) Representative: **Hervouet, Sylvie et al**
Feray Lenne Conseil,
39/41, avenue Aristide Briand
92163 Anthony Cedex (FR)

(54) Methods and systems for connection determination in a multi-point virtual private network

(57) A method and system for connecting a customer equipment (CE) communication device to a virtual private network (VPN) is provided. A virtual private network membership signal is generated at the customer equipment and transmitted to service provider equipment. The signal includes an identifier which identifies the customer equipment as a member of the virtual private network. On receiving the signal, service provider equipment such as a network element verifies that the customer equipment belongs to the virtual private network based on the customer identifier and only connects the customer equipment to the VPN if the verification is successful. The membership signal may be generated by a customer identification device distributed to the customer and installed in customer equipment to be connected to a virtual private network.

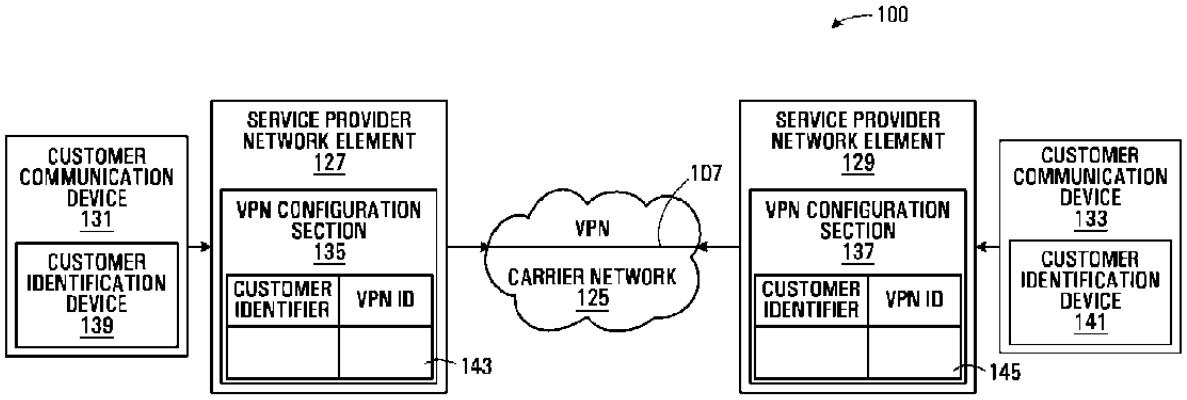


FIG. 3

EP 1 670 188 A2

Description

Field of the Invention

[0001] The present invention relates to methods and systems for connecting customer communication devices to a virtual private network and in particular, but not limited to, methods and systems for connecting communication devices to a multi-point virtual private network (mpVPN).

Background of the Invention

[0002] Virtual private networks allow predefined customer communication devices to be interconnected across a public network to enable private communication between devices which belong to the same VPN. Virtual private networks can be configured and implemented in a variety of different ways. For example, VPNs may be implemented using a link layer protocol such as TDM, FR (frame relay) or ATM (asynchronous transfer mode). These protocols allow point-to-point connectivity between two customer communication devices by forming a direct private connection or dedicated virtual private circuit (VPC) between the two devices, each connection being configured manually. However, VPNs based on these protocols are not generally implemented to allow multi-point connections, i.e. direct connections between all devices on the same virtual private network, with the service provider providing meshed connectivity.

[0003] A multi-point VPN is a service that implements an Ethernet LAN over a virtual layer 2 or layer 3 VPN in the carrier's domain, and typically connects numerous end-customer sites.

[0004] When configuring a virtual private network, it is important to ensure that only the intended subscriber equipment is connected to the VPN so that the network privacy and security of each customer is maintained. VPNs based on TDM, FR or ATM are less vulnerable to improper connection or misconfiguration as they are mostly point-to-point in nature and typically involve uniquely configured or custom data equipment at the customer premises. This implies that random misconnections would not result in an operational link and would very likely result in network alarms or "trouble tickets".

[0005] In contrast, configuring multi-point VPNs correctly and maintaining the configuration as customer drops are added and removed from the VPN instance can be error prone as it involves a number of configuration steps on carrier equipment that is shared across multiple end users, both at the physical layer (shared CPE or data terminating equipment) and the Operational Support System (OSS). The new generation of Ethernet/IP mpVPNs that interconnect customer CPE equipment utilize widely used and well standardized protocols and interfaces so that unwanted connections or "joins" to an mpVPN could easily go undetected and could provide a viable connection to an unintended party. Since the service provider would likely offer mpVPN services to a great number of clients such as enterprises and institutions, the risk and adverse consequences of inadvertently connecting the host node of one client to another client's mpVPN cannot be overlooked.

[0006] U.S. Patent Application Publication No. 2004/0093492 describes generating a digital certificate defining a VPN by aggregating configuration parameters from both a service provider and the customer. The digital certificate is used by the VPN service provider or the VPN customer to verify the VPN configuration or associated configuration logs by comparing information contained in the certificate with data stored at a customer workstation or in the service provider database.

[0007] When a customer communication device is to be connected to a VPN, there is a possibility that the physical connection of the device interface and the provider edge node will be incorrectly implemented so that for example the customer device becomes connected to the VPN of another customer. Although the methods discussed above may allow such a misconfiguration to be detected, none of these methods prevent a customer communication device from being initially connected to an incorrect VPN to thereby prevent any communication between the device and the incorrect VPN.

[0008] U.S. Patent Application Publication No. 2004/0088542 (Daude et al.) describes a method for interconnecting different VPNs. An interconnection device analyzes information contained in digital certificates to identify VPN properties of a device being connected and compares these properties to those contained in another digital certificate of another VPN.

[0009] The interconnection device implements the VPN rules from one or both of the interconnecting VPNs which are necessary to establish a secure interconnection. The interconnection device implements secure interconnection between VPNs without the need for a completely centralized decision-making process.

[0010] Draft-IETF-BONICA-13VPN-AUTH-03.txt "CE to CE Authentication from Layer 3 VPNs", June 2002, and Draft-IETF-13VPN-13VPN-AUTH-00.txt "CE to CE Member Verification_for Layer 3 VPNs" September, 2003, are concerned with the problem of VPN misconfigurations. A customer equipment-based verification mechanism is proposed in which each customer VPN site sends a "magic cookie" or token to the provider edge (PE) router that supports it. Upon receiving the token, the PE router connects the site to the VPN and distributes the token to other customer sites on the VPN, which verify the validity of the token. If the token is not valid, an alarm is raised at the customer VPN sites, and in this way misconfigurations are detected and indicated to the customer. As an optional variant, the first of these references describes an authentication process in which a PE router that receives a magic cookie from a CE transmits an authentication request which includes the magic cookie to a customer controlled server. If the server explicitly rejects the authentication request, the PE router terminates

APPLE EXHIBIT 1102, Page 148 of 1048

the authentication process and will neither accept traffic from the CE nor send traffic to the CE. However, if the customer controlled server cannot be contacted or sends no response at all, the PE router nevertheless joins the CE to the VPN. On the other hand, in the CE to CE based verification method disclosed in the second of these two references, there is no customer controlled authentication server and the PE simply connects the site to the VPN and immediately distributes tokens to other customer sites on the VPN.

[0011] A shortcoming of both of these proposals is that they are incapable of ensuring that a connection of non-VPN member equipment to a VPN is always prevented. Instead, they allow misconfigurations to be detected, and require customer interaction to rectify a carrier error.

Summary of the Invention

[0012] According to one aspect of the present invention, there is provided a customer equipment communication device comprising signal forming means adapted to form a virtual private network membership signal for transmission to and use by service provider equipment, wherein the signal includes an identifier for identifying said customer equipment as a member of a predetermined virtual private network, and is conditioned to cause said service provider equipment to verify that said communication device is a member of said predetermined virtual private network.

[0013] According to another aspect of the present invention, there is provided an apparatus for controlling connection of a customer communication device to a virtual private communication network, comprising means for receiving a signal from a customer communication device, determining means for determining from the signal whether or not the customer communication device is a member of a predetermined virtual private communication network, and controlling means for controlling connection of the customer communication device to the predetermined virtual private network based on the determination made by the determining means.

[0014] According to another aspect of the present invention, there is provided a method of controlling connection of a customer communication device to a virtual private communication network, comprising the steps of receiving at service provider equipment a signal from a customer communication device, determining at the service provider equipment whether or not the customer communication device is a member of a predetermined virtual private communication network based on information contained in the signal, and controlling connection of the customer communication device to the virtual private network based on the result of the determination.

[0015] Advantageously, in this arrangement, a customer communication device, such as a switch, router or host transmits a signal containing a customer identifier to service provider equipment responsible for configuring one or more virtual private networks. The configuration section of the service provider equipment determines from the customer identifier contained in the signal whether or not the customer device is a member of a predetermined virtual private network before connecting the communication device to the VPN. Advantageously, this arrangement enables an incorrect physical connection of a customer communication device at a provider edge node to be detected before data communication between the device and the virtual private network is enabled.

[0016] Furthermore, as the authentication process is performed by equipment under the control of the service provider, rather than requiring a customer controlled authentication server, a customer identifier belonging to one VPN is not passed to the customer of another VPN, so that each customer identifier can remain secret as between one customer and another.

[0017] Moreover, this arrangement allows the service provider equipment to verify whether or not customer equipment should be connected to a VPN so that, unlike the prior art methodologies, the service provider equipment can always ensure that a connection is prevented if the authentication process fails.

[0018] In one embodiment, the authentication process is performed autonomously by the service provider network elements, for example, provider edge nodes, which are connected directly to customer equipment from which the VPN request is transmitted. Advantageously, this arrangement removes the need for element, network, or OSS management systems to partcipate in or orchestrate the authentication process thereby removing the need for modifying element, network or OSS systems to conform to a specific implementation of the authentication process. The simplification provided by this embodiment thereby makes the authentication process more robust and reliable.

[0019] According to another aspect of the present invention, there is provided a method of requesting connection of a customer equipment communication device to a predetermined virtual private network, comprising the steps of: forming at said customer equipment, a virtual private network membership signal for transmission to and use by service provider equipment, wherein the signal includes an identifier for identifying said customer equipment as a member of said predetermined virtual private network and is conditioned to cause said service provider equipment to verify that said communication device is a member of said predetermined virtual private network, and transmitting said signal from said customer equipment communication device to said service provider equipment.

[0020] According to another aspect of the present invention, there is provided a method of controlling connection of a customer communication device to a virtual private communication network comprising: monitoring at service provider equipment, receipt of a predetermined signal from a customer communication device, and controlling connection of said customer communication de-

APPLE EXHIBIT 1102, Page 149 of 1048

vice to a predetermined virtual private communication network based on whether or not said predetermined signal is received at said service provider equipment within a predetermined time.

[0021] In some embodiments, where a connection between the customer communication device and the virtual private communication network is previously established, the step of controlling comprises disabling the connection if the signal is not received within said predetermined time.

[0022] In some embodiments, where a connection between the customer communication device and the virtual private communication network is previously established, the step of controlling comprises continuing to enable the established connection if the signal is received within said predetermined time.

[0023] In some embodiments, the controlling is performed as part of a virtual private network configuration process at the service provider equipment.

[0024] According to another aspect of the present invention, there is provided an apparatus for controlling connection of a customer communication device to a virtual private communication network comprising: means for receiving a signal from a customer communication device, determining means for determining from information in said signal whether or not said customer communication device is a member of a predetermined virtual private communication network, and controlling means for controlling connection of said customer communication device to said predetermined virtual private network based on the determination made by said determining means.

[0025] According to another aspect of the present invention, there is provided a method of detecting member equipment of a virtual private network comprising the steps of: receiving signals which originate from customer equipment communication devices, the signals each containing a customer identifier and a virtual private network identifier, detecting the identifiers in the signals and recording information based on each detected identifier.

[0026] According to another aspect of the present invention, there is provided a customer identification device comprising: a non-volatile memory for storing a customer identifier, signal forming means for forming a signal conditioned for transmission to a virtual private network configuration section of a predetermined carrier network and for causing said configuration section to verify that said device is a member of a predetermined virtual private network, the signal containing said customer identifier, and connection means for connecting said device to a customer communication device.

[0027] According to another aspect of the present invention, there is provided a method of controlling connection of customer communication equipment to a virtual private network, comprising the steps of: receiving at service provider equipment a predetermined customer identifier associated with a virtual private network from a customer equipment communication device, subsequently receiving another customer identifier, determining whether the other customer identifier is sufficiently similar to said predetermined customer identifier that both identifiers belong to the same customer, and controlling connection of service provider equipment based on the result of said determining step.

[0028] According to another aspect of the present invention, there is provided an apparatus for controlling connections to one or more virtual private networks, comprising receiving means for receiving from a customer equipment communication device a predetermined customer identifier associated with a virtual private network, and for receiving subsequent to receipt of said predetermined customer identifier, another customer identifier, and verification means for verifying whether the other customer identifier is sufficiently similar to said predetermined customer identifier that both identifiers belong to the same customer, and connection control means for controlling connection of customer communication equipment to said virtual private network based on the result of the verification by said verification means.

[0029] In some embodiments, the customer identifier comprises a field of characters which is common to all customer equipment of a predetermined customer to be connected to the virtual private network.

[0030] The characters of the field may be selected by the customer.

[0031] In some embodiments, the range of characters from which each character can be selected and/or the number of characters in the field, is sufficient to cause the probability of any virtual private network customer of the service provider selecting the same sequence of characters to be less than a predetermined value, for example less than 1 in a 1,000,000.

Brief Description of the Drawings

[0032] Examples of embodiments of the present invention will now be described with reference to the drawings in which:

Figure 1 shows a schematic diagram of a communication network in which an embodiment of the present invention is implemented;

Figure 2 shows an example of a customer identification packet according to an embodiment of the present invention;

Figure 3 shows a communication network in which another embodiment of the present invention is implemented;

Figure 4 shows a communication network in which another embodiment of the present invention is implemented; and

Figure 5 shows an embodiment of a customer iden-

APPLE EXHIBIT 1102, Page 150 of 1048

tification device according to an embodiment of the present invention.

Description of Embodiments

[0033] Figure 1 shows a schematic diagram of a communication network in which an embodiment of the present invention is implemented. In particular, Figure 1 shows first and second customer communication devices 3, 5 which are to be connected to a virtual private network 7 over a carrier network 9 which is managed by a network management system 11. The customer communication devices may comprise any communication device connectable to a network, for example, a workstation, a host computer, a switch or a router. A device 13, 15 is connected to each customer communication device which contains an identifier for the customer. The identifier is transmitted from the customer communication device to the carrier network 9 and is used by the carrier network to verify that the customer communication device is a member of the virtual private network 7.

[0034] In one implementation, the carrier network 9 is adapted to verify, using the customer identifier transmitted from the communication device, that the communication device is a member of the VPN before the carrier network connects the customer communication device 3, 5 to the VPN 7. Alternatively, or in addition, the customer identifier may be transmitted from the customer communication device to the carrier network after the customer communication device has been connected to the VPN to verify that the communication device is an authorized member of the VPN, and the signal may be transmitted periodically.

[0035] The customer identification device 13, 15 may comprise any suitable device that can be connected to the customer communication device for transmitting, or causing the customer communication device to transmit, a customer identifier to the carrier network. The device may include a memory for storing the customer identifier and may further include a signal generator for generating a signal which includes the customer identifier for transmission to the carrier network. Alternatively, the customer identification device may be adapted to transmit the customer identifier to a data communications processor 17, 19 of the customer communication device and the processor may generate a signal containing the customer identifier for transmission to the carrier network.

[0036] In this embodiment, the network management system 11 includes a virtual private network configuration section 21 which is responsible for the connection of customer communication devices to one or more virtual private networks. The VPN configuration section 21 includes a table 23 containing customer identifiers and an identification of each virtual private network with which they are associated.

[0037] In one implementation, a message or packet (or token) 25, 27 addressed to the VPN configuration section of the carrier network is formed at the customer communication device, which includes the customer identifier recorded in the customer identification device 13, 15, and is transmitted from the customer communication device to the network management system 11. On receiving the message, the VPN configuration section 21 checks the customer identifier against the list of customer identifiers stored in the table 23, and if a match is found, the VPN configuration section permits the customer communication device identified in the message to be connected to the VPN associated with the customer identifier. However, if the customer identifier in the message does not match any customer identifiers contained in the table 23, the VPN configuration section prohibits connection of the customer communication device to any VPN.

[0038] In another implementation, the packet 25, 27 transmitted from the customer communication device may contain a request for the customer communication device to be connected to a particular VPN. In this case, the packet contains the VPN identifier identifying the VPN to which the customer communication device is to be connected, and the customer identifier which may include a group identifier and/or an identification of the customer communication device, such as its network address. On receiving the request packet, the VPN configuration section 21 checks the VPN ID and the customer identifier contained in the packet with those stored in the table 23 and if a match of both parameters is found, the VPN configuration section 21 allows the customer communication device 3 to be connected to the VPN, otherwise connection to the VPN is denied.

[0039] Advantageously, this arrangement, in which an authentication signal is transmitted from a customer communication device to a carrier network, allows the carrier network to verify reliably whether or not the customer communication device is a member of a predetermined virtual private network before the device is connected to the VPN, and therefore prevents VPN misconfigurations. Furthermore, the customer communication device may be adapted to periodically transmit similar packets containing the customer ID to the carrier network to enable the carrier network to periodically check that the customer communication device continues to be a member of the virtual private network after being connected thereto.

[0040] In one embodiment, if a customer communication device becomes disconnected from the VPN, and its reconnection to the VPN is subsequently required, the customer communication device transmits a reconnection request and the customer ID (either separately or together) to the carrier network equipment responsible for VPN membership verification and connection. On detecting the request and customer ID, the carrier network equipment authenticates the customer equipment as belonging to the VPN using the customer ID before allowing reconnection.

[0041] The customer identifier may comprise any suitable identifier and may include several parts. In one embodiment, the customer identifier may simply comprise the name of the customer or another identifier which is

APPLE EXHIBIT 1102, Page 151 of 1048

unique to the customer. The customer identifier may comprise a common or group customer identifier which is used by customer communication devices all belonging to the same customer, and a second identifier which additionally identifies the particular customer communication device. The customer identifier may or may not also be encrypted.

[0042] An example of a VPN membership verification packet is shown in Figure 2. The membership verification packet 41 includes a destination address which enables the packet to be transmitted to the VPN configuration section of the carrier network. The packet also includes a number of fields 45, 47, 49 which, in this embodiment contain the VPN identifier, a group identifier for the customer, and an identifier identifying the particular communication device to be connected to the VPN. Together with an appropriate query (e.g. one or more commands) the customer communication device will transmit an appropriate response containing the verification packet as shown in Figure 2 enabling the customer communication device to be verified by the service provider.

[0043] In other embodiments of the present invention, authentication of a customer communication device to be connected to a particular VPN may be performed by network devices of the carrier network other than the network management system. For example, authentication may be performed by network elements or nodes of the network such as a provider edge (PE) node of the carrier network. An example of such an implementation is described below with reference to Figure 3.

[0044] Referring to Figure 3, a carrier network 125 includes a plurality of PE nodes 127, 129, each of which serves as both ingress and egress nodes to customer communication devices 131, 133 connected thereto. Each PE node 127, 129 includes a VPN configuration section 135, 137 for configuring one or more virtual private networks and which also authenticates customer identification devices to be connected (or reconnected) or which are already connected to a particular VPN.

[0045] Each customer communication device 131, 133 includes a customer identification device 139, 141 connected thereto which transmits or causes transmission of a customer identifier from the customer communication device to a PE node of the carrier network 125.

[0046] When first configuring a new VPN 107, a record identifying the VPN and a customer identifier associated with the VPN is created and stored in the VPN configuration section of a PE node of the carrier network 125. This record may be created in response to a VPN configuration request transmitted from one of the customer communication devices to be connected to the VPN. The request may include the customer identifier and also a VPN identifier which is to be created. Alternatively, the VPN identifier may be determined by the carrier network and transmitted to the customer communication device. On receipt of the request, which includes the customer identifier, the PE node stores the customer identifier together with the VPN identifier and transmits both parameters to one or more other PE nodes of the carrier network 125.

[0047] Each additional customer communication device which is connected to the VPN is provided with a customer identification device which causes a message or packet containing the customer identifier to be transmitted to the PE node of the carrier network to which it is connected to enable the PE node to authenticate the customer communication device as a member of the VPN. The customer identification device connected to each customer communication device may be similar to any of the embodiments described above in connection with Figure 1 and may operate in a similar manner.

[0048] The customer identifier generally includes an identifier which is common to all members of the VPN and may also include an additional identifier which uniquely identifies the particular customer communication device. The customer identifier signal transmitted from each customer communication device enables the PE node to which it is connected to verify that the customer device is a member of the VPN group before allowing the connection, and this arrangement therefore prevents incorrect communication devices from being connected to the VPN. Furthermore, this arrangement uses PE nodes to verify whether or not a particular customer communication device should be connected to a VPN without involving the element management, network management, or the Operational Support System (OSS), and therefore does not involve and is independent of higher layers of software applications. This arrangement is also more robust as it does not rely upon the success of communications to and from the OSS or upon the OSS operating properly, or to have been so modified. to provide the required verification. This arrangement also does not require any pre-configuration regarding the association of a group customer identification to a specific VPN.

[0049] Customer identification devices may be provided to the customer for connection to the customer communication devices when the customer subscribes to a virtual private network service. For example, a quantity of customer identification devices may be issued to the customer by the service provider of the virtual private network service and distributed to each customer site which is to be connected to the service. A customer identification device is connected by authorized personnel such as IT staff, to customer equipment at each site that is to be connected to the VPN service. Each customer identification device causes a customer ID signal to be transmitted to the VPN configuration application or process of the carrier network, which can then verify that the customer equipment at each site should be connected to the VPN before allowing the connection.

[0050] In an alternative embodiment, customer identification devices may be preinstalled in the customer communication devices, for example by the manufacturer or system integrator, rather than at a later time after the communication devices have been installed at the cus-

APPLE EXHIBIT 1102, Page 152 of 1048

tomer site. When a VPN service is required, the customer identification devices could be activated to transmit or cause transmission of the customer ID to the configuration process of the carrier network. Knowledge of the customer ID is independently passed to the configuration process of the carrier network to allow verification that customer equipment should be connected to a VPN.

[0051] Since, in this embodiment, the group identification may be known to a third party, i.e. the manufacturer of the communication device with the preinstalled customer identification device, the customer identification signal may be suitably secured by any appropriate technique such as encryption techniques, of which public key infrastructure (PKI) techniques are one example. In this case, a key or customer signature is provided to the carrier network to allow the carrier network to read and authenticate the customer ID contained in the signal. If the customer key or signature matches, the configuration process of the carrier network allows the connection and enables data communication, otherwise the connection is denied.

[0052] Preinstallation of customer identification devices in customer equipment advantageously eliminates the need to separately distribute special ID devices that are limited to one customer, thereby reducing inventory and distribution concerns.

[0053] In another embodiment of the present invention, the customer may provide the service provider with information that enables the service provider to query and uniquely identify valid equipment before allowing connection to the mpVPN. For example, the carrier network may be provided with the MAC (Media Access Control) addresses of each customer communication device to be connected to a specific VPN instance, together with an appropriate query (e.g. one or more commands) which causes the customer communication device to transmit an appropriate response containing data which enables the customer communication device to be verified by the service provider as a valid member of that specific VPN. The response signal may contain a unique customer identifier and optionally other identifiers such as the VPN identifier to which the communication device is to be connected. In addition, the response signal may be secured, for example, by encryption. On receipt of the response signal by the VPN configuration process of the carrier network, the configuration process uses the signal to verify against its own verification data whether to connect the communication device to the VPN instance and permit data communication.

[0054] In other embodiments of the present invention, when commissioning a new virtual private network for the first time, the service provider equipment (e.g. network management system and/or network elements) may be arranged to connect the customer communication device to the virtual private network from which the customer identifier associated with that VPN is first received by the customer equipment. Advantageously, in this arrangement, the customer equipment needs no prior knowledge of the customer identifier associated with the VPN. On receiving subsequent requests from customer equipment to be connected to that VPN, the VPN configuration section of the service provider equipment simply verifies whether the subsequently received IDs match the first received customer ID and, if so, the connection is allowed, otherwise the connection is denied.

[0055] When a new VPN is first commissioned, the VPN configuration section may record the first received customer ID for future use in verifying subsequently requested connections. The record may be stored permanently or temporarily for a limited time and then deleted. In cases where no record of the customer ID is retained by the service provider equipment, and a connection to the VPN is subsequently requested, the service provider equipment may be adapted to request the customer communication device from which the customer ID was first received, to retransmit the customer ID to enable the VPN configuration section to compare this with the customer ID in the subsequent request to determine whether to allow the new requested connection.

[0056] Alternatively, the customer communication device first connected to the VPN may repeatedly transmit the customer identifier to the service provider equipment to enable the VPN configuration section to use the retransmitted customer ID in verifying a subsequently requested connection.

[0057] Advantageously, either of these two arrangements obviates the need for the service provider equipment to maintain a record of the customer identifier or even needing to know what the customer ID is, thereby significantly reducing the risk of the customer identifier being revealed to unauthorized parties through the service provider equipment.

[0058] The above-described VPN connection verification process is based on a comparison of customer identifiers received from customer equipment communication devices, rather than with any record of a customer identifier maintained by the service provider. The customer identifier may be generated either by the customer or the service provider. Advantageously, if the customer identifier is generated by the customer, the customer identifier need never be retained by the service provider equipment, as the service provider equipment simply performs an equivalency check between two customer identifiers it receives. This also assists in making the customer ID accessible to service provider personnel.

[0059] In any of the embodiments described above, the customer identifier may comprise a plurality of characters in which the range of characters from which each character can be selected and/or the total number of characters in the customer identifier is sufficiently large that it would be improbable for any other VPN customer of the same service provider to choose the same customer ID. For example, the range or number of characters can be selected so that the probability is less than at least 1 in 50, preferably less than at least 1 in 1000 and more preferably less than 1 in a million. This allows the cus-

APPLE EXHIBIT 1102, Page 153 of 1048

tomer ID to be selected by the customer, rather than by the service provider, in a similar manner to selecting a PIN (Personal Identification Number) or password.

[0060] In any of the embodiments described herein, the customer ID may comprise several parts, including a predetermined field which is common to all equipment of the same customer to be connected to a particular VPN. In this case, the service provider equipment may only need to compare this predetermined field of one customer identifier with the corresponding field of another customer identifier. In this way, the customer equipment need only check that two customer identifiers are sufficiently similar to one another, and there is no requirement for the whole customer identifier to be the same as another nor any need to check equivalency of the whole customer identifier. The field or portion of the customer ID selected for comparison must be that portion which is unique to each customer. If the customer ID is selected by the service provider, or otherwise verified as unique, the field may be relatively short. If the characters of the field are selected by the customer, the field shall be sufficiently long to ensure its uniqueness, as described above.

[0061] In embodiments of the invention, more than one customer identification device may be connected to or installed in a customer communication device to provide redundancy in case one customer ID device fails. This is particularly beneficial when the continuation of an allowed connection of a customer communication device to a VPN, once a connection has been established, is dependent on the continued transmission of the customer identification signal from the customer equipment to the carrier network. In this case, where failure to send the signal would otherwise cause the carrier network to disconnect the customer equipment from the VPN, the provision of one or more additional customer identification devices would allow continued transmission of the signal and thereby prevent disconnection of the customer equipment should one customer ID device fail. Transmission of the signal may be monitored by the CPE equipment so that failures can be detected and the auxiliary or backup customer identification device activated, as necessary.

[0062] Figure 4 shows an example of a communication network in which a customer communication device has a plurality of customer identification devices to provide redundancy. The components of Figure 4 are similar to those shown in Figure 3, and like parts are designated by the same reference numerals. In this embodiment, each customer communication device 131, 133 comprises a first customer identification device 139, 141 and a second customer identification device 151, 153. The first customer identification device may constitute the normally active device which provides the customer identifier to the service provider network, and the second customer identification device may constitute the redundant device which is activated if the first customer identification device fails.

[0063] Figure 5 shows a schematic diagram of a customer identification device according to an embodiment of the present invention. The communication device 201 comprises a memory 203 (e.g. a non-volatile memory) which stores the customer identifier used by the service provider equipment to authenticate whether the customer equipment is member equipment of a predetermined virtual private network. The memory may also contain other data such as an identification of the virtual private network to which the customer belongs and/or the address of the service provider equipment which controls authentication and connection to VPNs. The customer identification device may also comprise a processor 205 for generating a packet or other signal containing the customer identifier used for authentication. A communication port 207 is also provided to connect the customer identification device to customer communication equipment at a customer site so that the signal generated by the customer identification device is transmitted to the service provider network. The port may comprise a unidirectional output port or a bi-directional input/output port. The customer identification device may be powered by either an internal or external power source, and in the case of an external power source, the customer identification device may be provided with suitable power receiving terminals and connectors.

[0064] Another embodiment of the customer identification device may comprise simply a memory storing the customer ID, and possibly other data as indicated above, and a suitable port for connection to customer equipment. The memory may comprise a non-volatile memory, so that data can be held therein without the need for a power source. In this case, the customer equipment is adapted to generate a suitable packet (or other signal) containing the customer ID for transmission to the service provider network.

[0065] Advantageously, the embodiments described herein enable a physical connection of a customer communication device to a virtual private network to be detected before data communication between the device and the VPN is enabled. For example, an incorrect connection may occur when VPN provider personnel physically connect a customer communication device intended to be connected to that customer's VPN to the VPN of another customer, by for example, connecting the communication link to an incorrect port. However, before data communication is enabled, the VPN configuration section checks whether the customer identifier transmitted from the customer communication device corresponds to the customer identifier for the VPN associated with that port, and as the customer communication device is connected to the incorrect port, the verification section will deny the connection, and may also provide an indication of the denied connection to the VPN provider personnel so that the misconfiguration can be rectified.

[0066] Changes and modifications to the embodiments described herein will be apparent to those skilled in the art.

APPLE EXHIBIT 1102, Page 154 of 1048

Claims

1. A customer equipment communication device comprising signal forming means adapted to form a virtual private network membership signal for transmission to and use by service provider equipment, wherein the signal includes an identifier for identifying said customer equipment as a member of a predetermined virtual private network and is conditioned to cause said service provider equipment to verify that said communication device is a member of said predetermined virtual private network.

2. A communication device as claimed in claim 1, wherein said identifier comprises at least one of an identifier uniquely identifying said customer equipment and an identifier used to identify a group of equipment belonging to said virtual private network.

3. A communication device as claimed in claim 2, wherein at least one of said unique identifier and said group identifier is encrypted.

4. A communication device as claimed in claim 1, wherein said identifier includes an identifier of said customer equipment and an identifier of said predetermined virtual private network.

5. A communication device as claimed in any preceding claim, wherein said signal forming means is arranged to condition said signal for transmission to service provider equipment adapted to configure said virtual private network.

6. A communication device as claimed in claim 5, wherein said service provider equipment comprises at least one of a service provider network management system and a network element at the edge of said service provider network.

7. A communication device as claimed in any preceding claim, wherein said signal forming means is adapted to form said signal at least one of before and after said communication device is connected to said virtual private network by said service provider.

8. A communication device as claimed in any preceding claim, comprising signal transmission means for transmitting said signal to said service provider equipment.

9. A communication device as claimed in claim 8, wherein said signal transmission means is adapted to transmit said signal at least one of before and after said customer communication device is connected to said virtual private network.

10. A communication device as claimed in claim 8, wherein said signal transmission means is adapted to repeatedly transmit said signal periodically.

11. A communication device as claimed in any preceding claim, further comprising a second signal forming means adapted to form said virtual private network membership signal.

12. A communication device as claimed in claim 11, further comprising detection means for detecting a failure of transmission of said virtual private network membership signal from said customer communication device and for causing a virtual private network membership signal to be formed by said second signal forming means in response to said detected failure.

13. A communication device as claimed in any one of claims 8 to 12, further comprising second signal transmission means for transmitting said virtual private network membership signal to said service provider.

14. A communication device as claimed in claim 13, further comprising detection means for detecting failure of transmission of said signal by said signal transmission means and means for causing said signal to be transmitted by said second transmission means in response to detection of said failure.

15. A communication device as claimed in any preceding claim, wherein said signal forming means is one of (1) preinstalled in said customer equipment communication device before said communication device is first delivered to said customer and (2) connected to said customer equipment communication device after said communication device is first delivered to said customer.

16. A communication device as claimed in any preceding claim, wherein said signal forming means comprises a customer identification device which contains said customer identifier.

17. A communication device as claimed in any preceding claim, further comprising receiving means for receiving a predetermined signal from service provider equipment and wherein said communication device is adapted to transmit said virtual private network membership signal to said service provider equipment in response to said predetermined signal.

18. A method of requesting connection of a customer equipment communication device to a predetermined virtual private network, comprising the steps of:

forming at said customer equipment, a virtual

APPLE EXHIBIT 1102, Page 155 of 1048

private network membership signal for transmission to and use by service provider equipment, wherein the signal includes an identifier for identifying said customer equipment as a member of said predetermined virtual private network and is conditioned to cause said service provider equipment to verify that said communication device is a member of said predetermined virtual private network, and transmitting said signal from said customer equipment communication device to said service provider equipment.

19. A method as claimed in claim 18, further comprising the step of connecting a customer identification device to said communication device to form said virtual private network membership signal.

20. A method of controlling connection of a customer communication device to a virtual private communication network comprising the steps of:

receiving at service provider equipment a signal from a customer communication device,
determining at said service provider equipment whether or not said customer communication device is a member of a predetermined virtual private communication network based on information contained in said signal, and
controlling connection of said customer communication device to said virtual private network based on the result of said determination.

21. A method as claimed in claim 20, wherein said customer communication device initially is not connected to said virtual private communication network, and wherein the step of controlling connection comprises enabling connection of the customer communication device to said virtual private communication network if, by said determining step, the customer communication device is determined to be a member of the virtual private communication network.

22. A method as claimed in claim 20, wherein said customer communication device is previously connected to said predetermined virtual private communication network, and the step of controlling connection comprises at least one of (1) permitting continued enablement of said connection if, by said determination step, the customer device is determined to be a member of the predetermined virtual private communication network, and (2) prohibiting a connection of said customer communication device to said predetermined virtual private communication network, if by said determining step, the customer communication device is determined not to be a member of said virtual private communication network.

23. A method as claimed in any one of claims 20 to 22, further comprising the step of monitoring at said service provider equipment receipt of a subsequent predetermined signal from said customer communication device, and controlling connection of said customer communication device to said virtual private communication network in response to said monitoring.

24. A method as claimed in claim 23, wherein the step of controlling said connection in response to said monitoring comprises disabling said connection if said further signal is not received within a predetermined time.

25. A method as claimed in any one of claims 20 to 24, wherein said service provider equipment comprises at least one of a network management system and a provider edge network element.

26. A method as claimed in any one of claims 20 to 25, further comprising the step of transmitting from said service provider equipment a customer identifier identifying said customer and a VPN identifier identifying said predetermined virtual private network to one or more provider edge network elements if, by said determining step, said customer communication device is determined to be a member of said predetermined virtual private network.

27. A method as claimed in any one of claims 20 to 26, wherein said determining step is performed as part of a virtual private network configuration process in said service provider equipment.

28. A method as claimed in claim 20, comprising receiving at said service provider equipment a signal requesting reconnection of a previously connected but subsequently disconnected customer communication device, and subsequently performing said determining and controlling steps in response to said signal containing said information.

29. A method as claimed in any one of claims 20 to 31, further comprising the step of providing said customer with a customer identification device for use in generating said signal from said customer communication device.

30. A method as claimed in claim 18, further comprising providing first and second independently operable customer identification devices each capable of forming said virtual private network membership signal, monitoring said first customer identification device from said virtual private network membership signal if said first customer identification device fails.

31. An apparatus for controlling connection of a customer communication device to a virtual private commu-

APPLE EXHIBIT 1102, Page 156 of 1048

nication network comprising:

means for receiving a signal from a customer communication device,
determining means for determining from information in said signal whether or not said customer communication device is a member of a predetermined virtual private communication network, and
controlling means for controlling connection of said customer communication device to said predetermined virtual private network based on the determination made by said determining means.

32. An apparatus as claimed in claim 38, wherein said controlling means is adapted to enable connection of said customer communication device to said predetermined virtual private network if said determining means determines that the customer communication device is a member of said predetermined virtual private communication network.

33. An apparatus as claimed in claim 31 or 32, wherein said controlling means is adapted to prohibit connection of the customer communication device to said predetermined virtual private network if said determining means determines that said customer communication device is not a member of said predetermined virtual private network.

34. An apparatus as claimed in any one of claims 31 to 33, wherein said information comprises at least one of (1) a customer identifier, and (2) an identifier identifying said predetermined virtual private communication network.

35. An apparatus for controlling connection of a customer communication device to a virtual private communication network comprising:

monitoring means for monitoring receipt of a predetermined signal from a customer communication device, and
controlling means for controlling connection of said customer communication device to a predetermined virtual private communication network based on whether or not said predetermined signal is received within a predetermined time.

36. An apparatus as claimed in claim 35, wherein said controlling means is adapted to at least one of (1) disable a previously established connection of said customer communication device to said virtual private network if said predetermined signal is not received within said predetermined time, and (2) permit a previously established connection between a customer communication device and said predetermined virtual private network to continue if said predetermined signal is received within said predetermined time.

37. An apparatus as claimed in claim 35 or 36, further comprising indicator means for providing an indication to an operator if said predetermined signal is not received within said predetermined time.

38. A customer identification device comprising:

a non-volatile memory for storing a customer identifier, signal forming means for forming a signal conditioned for transmission to a virtual private network configuration section of a predetermined carrier network and for causing said configuration section to verify that said device is a member of a predetermined virtual private network, the signal containing said customer identifier, and
connection means for connecting said device to a customer communication device.

39. A method of controlling connection of customer communication equipment to a virtual private network, comprising the steps of:

receiving at service provider equipment a predetermined customer identifier associated with a virtual private network from a customer equipment communication device,
subsequently receiving another customer identifier,
determining whether the other customer identifier is sufficiently similar to said predetermined customer identifier that both identifiers belong to the same customer, and
controlling connection of service provider equipment based on the result of said determining step.

40. A method as claimed in claim 39, wherein said predetermined customer identifier is the first customer identifier associated with said virtual private network to be received, and connecting the customer equipment communication device from which said first customer identifier is received to said virtual private network.

41. A method as claimed in claim 40, wherein said other customer identifier is received from another customer equipment communication device, and connecting said other customer equipment communication device to said virtual private network if said other customer identifier is determined to be sufficiently similar to said predetermined customer identifier.

APPLE EXHIBIT 1102, Page 157 of 1048

42. A method as claimed in claim 40, wherein said other customer identifier is received from another customer equipment communication device, and denying connection of said other customer equipment communication device to said virtual private network if the other customer identifier is determined to be insufficiently similar to said predetermined customer identifier.

43. A method as claimed in any one of claims 39 to 42, further comprising requesting the customer equipment communication device from which said predetermined customer identifier is received to send said predetermined customer identifier to said service provider equipment again in response to said service provider equipment receiving said other customer identifier, and wherein said determining step is performed based on the retransmitted predetermined customer identifier.

44. A method as claimed in any one of claims 39 to 43, comprising repetitively receiving said predetermined customer identifier which is retransmitted from said customer equipment communication device and wherein said determining step is performed based on a retransmitted predetermined customer identifier.

45. A method as claimed in any one of claims 39 to 44, wherein said predetermined customer identifier includes a field of characters which is common to all customer equipment of a predetermined customer to be connected to a predetermined VPN.

46. A method as claimed in claim 45, wherein the characters of said field are selected by said customer.

47. A method as claimed in claim 45 or 46, wherein at least one of (a) the range of characters from which each character in said field can be selected and (b) the number of characters in said field is sufficient to cause the probability of any other customer selecting the same sequence of characters to be less than a predetermined value.

48. A method as claimed in claim 47, wherein said predetermined value is 1 in a million.

49. A method as claimed in any one of claims 45 to 48, wherein said determining step comprises comparing said field with a field contained in said other customer identifier.

50. Apparatus for controlling connections to one or more virtual private networks, comprising receiving means for receiving from a customer equipment communication device a predetermined customer identifier associated with a virtual private network, and for receiving subsequent to receipt of said predetermined customer identifier, another customer identifier, and verification means for verifying whether the other customer identifier is sufficiently similar to said predetermined customer identifier that both identifiers belong to the same customer, and connection control means for controlling connection of customer communication equipment to said virtual private network based on the result of the verification by said verification means.

51. An apparatus as claimed in claim 50, wherein said connection control means is adapted to connect to said virtual private network the customer equipment communication device from which a customer identifier associated with said virtual private network is first received by said apparatus.

52. An apparatus as claimed in claim 51, wherein said connection control means is adapted to connect a customer equipment communication device from which said other customer identifier is received if said verification means determines that the other customer identifier is sufficiently similar to said first received customer identifier.

53. An apparatus as claimed in claim 52, further comprising transmitting means for transmitting to said first connected customer communication device a request for said predetermined customer identifier in response to receiving said subsequent customer identifier and wherein said verification means is adapted to verify whether said other customer identifier is sufficiently similar to said predetermined customer identifier transmitted from said customer equipment in response to said request.

54. An apparatus as claimed in any one of claims 50 to 53, wherein said customer identifier comprises a field of characters which is common to all customer equipment of a predetermined customer to be connected to said virtual private network.

APPLE EXHIBIT 1102, Page 158 of 1048

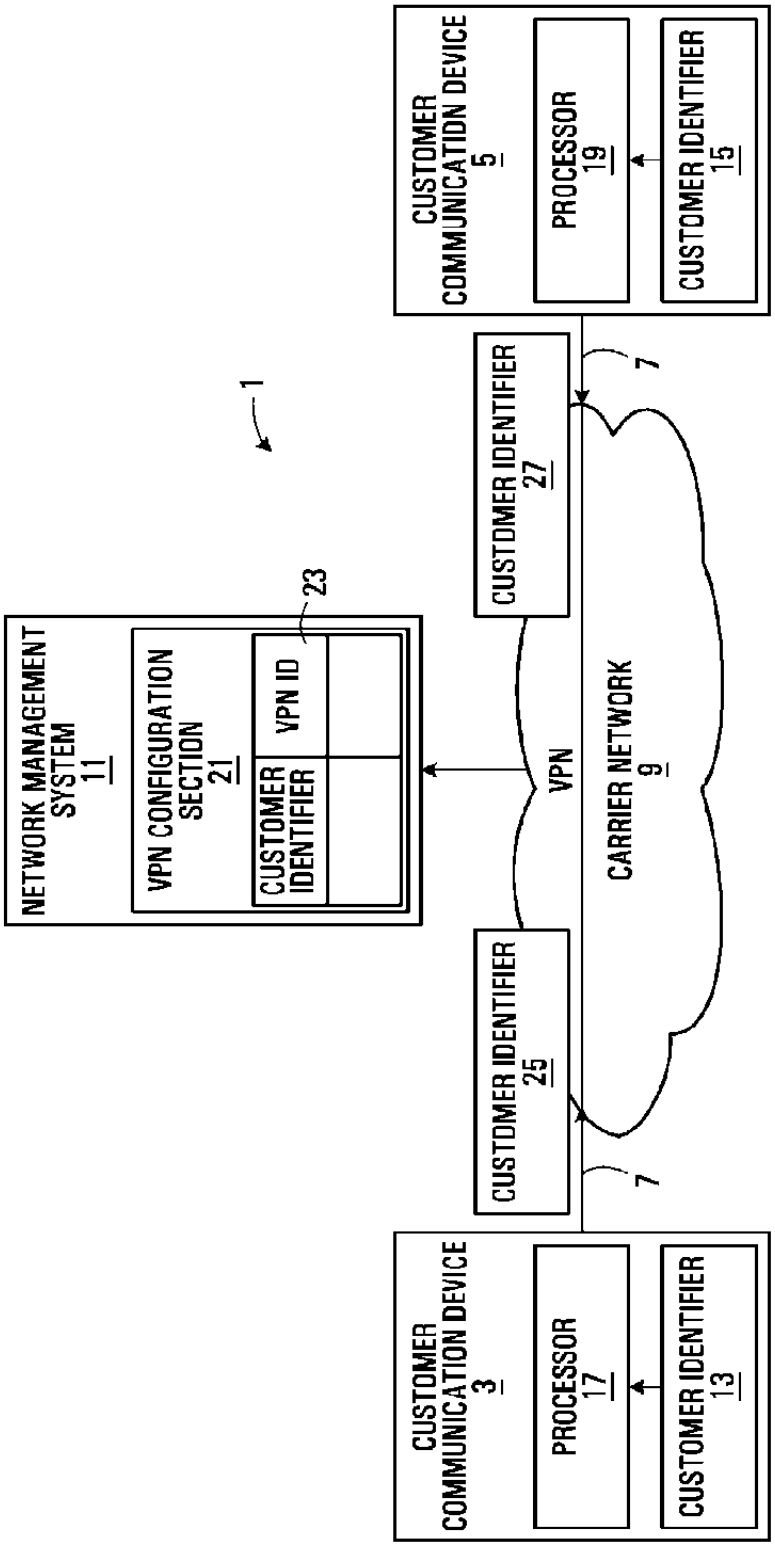


FIG. 1

APPLE EXHIBIT 1102, Page 159 of 1048

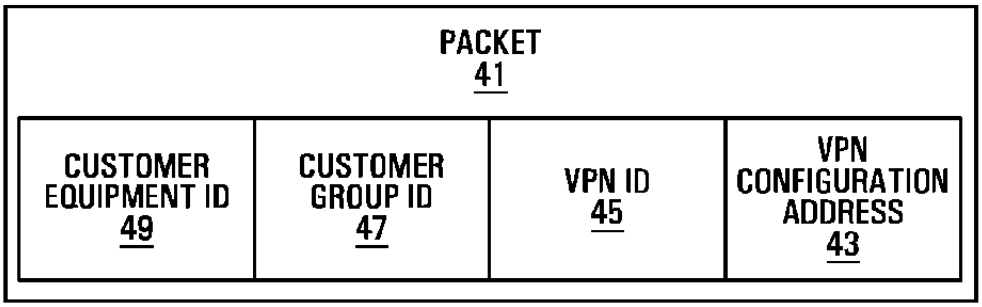


FIG. 2

APPLE EXHIBIT 1102, Page 160 of 1048

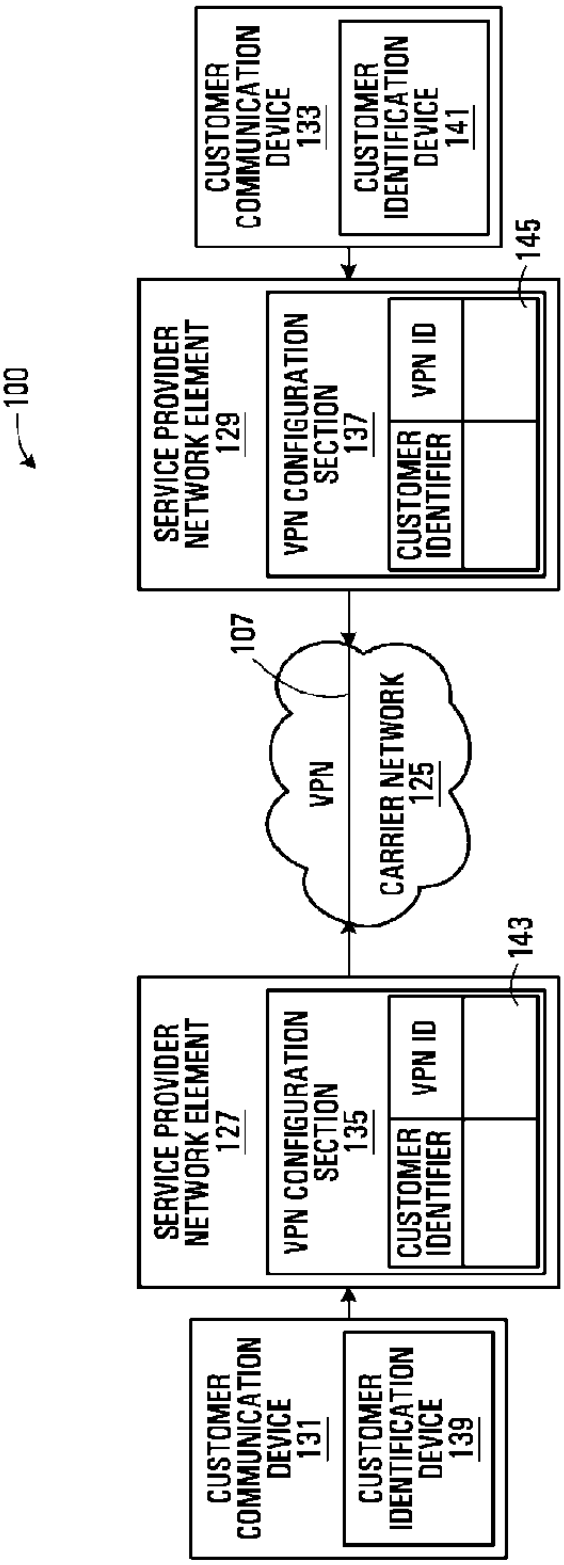


FIG. 3

APPLE EXHIBIT 1102, Page 161 of 1048

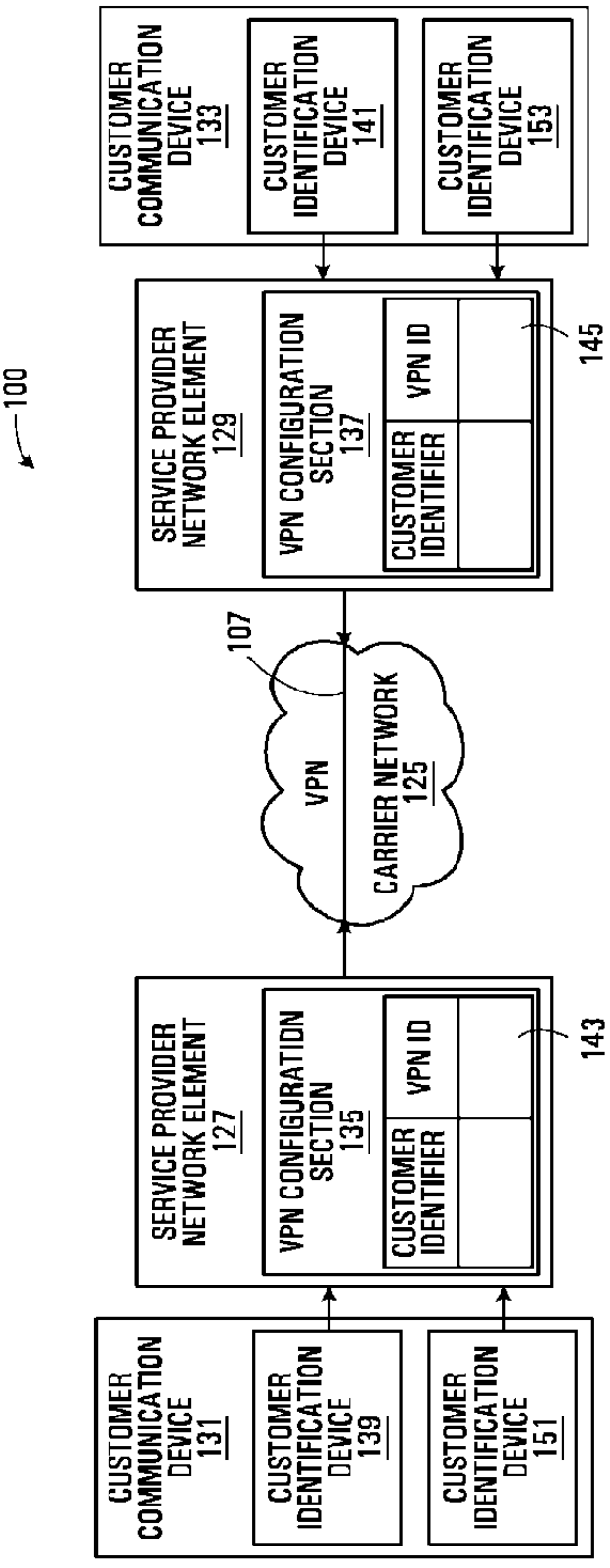


FIG. 4

APPLE EXHIBIT 1102, Page 162 of 1048

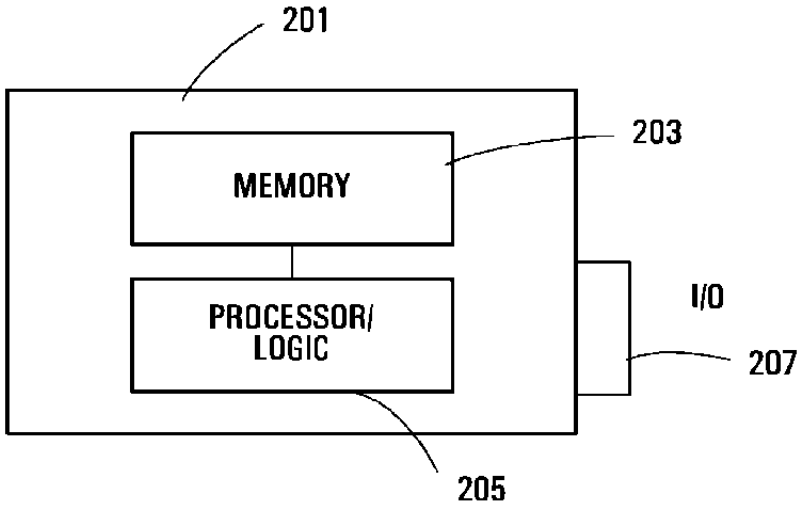


FIG. 5

APPLE EXHIBIT 1102, Page 163 of 1048

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 1/00 — A1

(11) International Publication Number: **WO 00/67095**

(43) International Publication Date: 9 November 2000 (09.11.00)

(21) International Application Number: PCT/US00/11545

(22) International Filing Date: 26 April 2000 (26.04.00)

(30) Priority Data:

60/131,769	30 April 1999 (30.04.99)	US
09/328,737	9 June 1999 (09.06.99)	US

(71) Applicant: TRYMEDIA SYSTEMS [US/US]; 1516 Folsom Street, San Francisco, CA 94103 (US).

(72) Inventor: TORRUBIA–SAEZ, Andres; Rambla Mendez Nunez, 34–1, E–03002 Alicante (ES).

(74) Agents: GLENN, Michael, A. et al.; Glenn Patent Group, Suite L, 3475 Edison Way, Menlo Park, CA 94025 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHODS AND APPARATUS FOR SECURE DISTRIBUTION OF SOFTWARE

(57) Abstract

Software is securely distributed with limited usage rights. The software may be an executable program and/or one or more data files such as image or multimedia data files. The software includes an access control object which prevents at least some usage of the software without use of a first access control code. The first access control code is produced based on selected information characteristic of the user's computer system. The access control code is produced in a server computer to which the user directs a request for the access control code. The user makes a payment to receive the access control code, which is then downloaded to the user's computer system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHODS AND APPARATUS FOR SECURE DISTRIBUTION OF SOFTWARE

Background of the Invention

The present invention relates to secure methods for distributing software and data objects, as well as to access-controlled software and data objects, and computer systems which practice or utilize any of the foregoing.

Commercial distribution of software and data (such as media files and reports) by data communication is a very rapidly growing form of commerce. It is both efficient and convenient as compared to traditional distribution methods.

Distribution of software and data on a "Try and Buy" basis permits the user to run or "demo" the product before committing to buy it. This assumes that the software licensor or media distributor somehow exercises control over the use of the product at least until the recipient buys the right to use it. The widespread availability of data communication, especially via the Internet, also emphasizes the need for the software licensor and the media distributor to exercise control over their products.

One technique for controlling access to executables involves "wrapping" the executable to be controlled within a second program, termed a "wrapper". In effect, the executable to be controlled and the wrapper are joined into one executable, in which the wrapper is executed first and controls access to the wrapped executable.

However, conventional software protection systems based on wrapping are easily circumvented by class attacks which destroy the security otherwise afforded by a given type of wrapper. This is achieved through a modification of only a single part of the wrapper which is identical in all wrappers of that type. Generic unprotectors can easily be obtained via the Internet.

Another form of attack is the so-called "dump attack" in which the attacker waits for the wrapped application to be decompressed and/or decrypted in memory, and then dumps it to a hard disk in its original, unprotected state. Programs to carry out dump attacks also are easily obtained via the Internet.

A widely used security device injects new code into an existing executable in order to control access to the latter. When the executable is run, a specially-designed DLL executable is loaded for controlling access to the existing executable. The presumed "security" afforded by this scheme is circumvented by eliminating the call to the DLL or by modifying the DLL itself.

It has been proposed to package data objects with executables which carry out such control functions.

A dedicated user program is required to decrypt, decompress and format the data for display by a monitor and/or an audio reproduction device. Consequently, it is necessary to provide a different user program for each data format which may be encountered. For example, a different program is required to play an avi file than is used to display a bmp or JPEG file.

It would, therefore, be desirable to provide methods, software and computer systems which control access to data objects, but do not require different programs to display or present objects in various formats. It would also be desirable to provide methods, software and computer systems which control access to executables but which are not subject to class attacks or dump attacks.

Summary of the Invention

As used in this application, the following terms shall have the indicated meanings:

Software: includes both data and programming instructions.

Package: any software to be stored, accessed, loaded, assembled, prepared for transmission or received as a unit.

Object: any software to be run, utilized or displayed as a unit.

Feature: a "feature" of an object is any function, instruction, capability, or information included therein, or controlled or enabled thereby.

Computer System: includes a single computer or multiple cooperating computers, and includes one or more PC's, mainframes, digital processors, workstations, DSP's or a computer network or networks, or a computer internetwork.

Wrapping: joining one executable with another executable in a package, one of the executables (termed the "Wrapper") being executed first and controlling access to the other executable.

Watermark: includes information in software which either enables identification of an owner, licensee, distributee or another having rights in or an obligation in connection with the software, or enables identification of a version or copy of the software. Usually, but not necessarily, the watermark is imperceptible and preferably is difficult to remove from the software.

Padding Area: a space within a software object or package which does not contain required code or data.

In accordance with an aspect of the present invention, a method of securely distributing software with limited usage rights is provided. The method comprises: supplying software for distribution to a user, the software including an access control object for preventing at least some usage thereof on a computer system without the use of a first access control code; producing the first access control code based on

selected information characteristic of the predetermined computer system; and supplying the first access control code to the predetermined computer system to enable the at least some usage of the software.

In accordance with another aspect of the present invention, an executable object is provided, comprising: a first code portion comprising first predetermined instructions; and a second code portion comprising loading instructions required for loading the first code portion in a memory of a computer system to be programmed thereby, the second code portion being operative to control the computer system to erase the loading instructions from memory upon loading the first code portion in memory.

In accordance with still another aspect of the invention, a software package is provided, comprising: a first executable object, and a wrapper for the first executable object, the wrapper being operative to erase predetermined software from the first executable object when it has been loaded in running format in memory.

In accordance with a further aspect of the present invention, a computer system is provided, comprising: a processor; a memory; an instruction input device; and an executable stored in the computer system, the executable having a first code portion comprising first predetermined instructions for execution by the processor, and a second code portion including loading instructions, the processor being operative upon receipt of a predetermined instruction from the instruction input device to load the second code portion in the memory, the processor being operative under the control of the loading instructions to load the first code portion in the memory and operative under the control of the second code portion to erase the loading instructions from the memory upon loading the first code portion in memory.

In accordance with yet another aspect of the present invention, a software package comprises: a first object providing a first set of a plurality of features; a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set; and an access control portion affording selective access to the first software object and/or the second software object.

In accordance with still another aspect of the present invention, a software package is provided, comprising: a first executable object, and a wrapper for the first executable object, the first executable object being operative, while running, to access a feature of the wrapper; the wrapper being operative to supply the feature to the first executable object when the feature is accessed thereby.

In accordance with yet another aspect of the invention, a software package is provided, comprising: a first executable object, and a wrapper for the first executable object, the first executable object being operative, while running, to access a feature of the wrapper; the wrapper being operative to supply the feature to the first executable object when the feature is accessed thereby.

In accordance with yet another aspect of the invention, a software package is provided comprising: a first executable object, and a wrapper for the first executable object, the first executable object being operative to call a predetermined feature external thereto; the wrapper being operative upon a call of the predetermined feature by the first executable object to transfer program execution control to a predetermined address within the wrapper to control access by the first executable object to the predetermined feature.

In accordance with a still further aspect of the present invention, a computer system is provided, comprising: a processor; a memory; an instruction input device; and a software package stored in the computer system, the software package having a first object providing a first set of a plurality of features, a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set, and an access control portion; the processor being operative to load the software package in the memory, the processor being further operative to request access to a selected one of the first and second objects in response to a predetermined instruction from the instruction input device, the access control portion being operative to selectively control access to the selected object.

In accordance with still another aspect of the present invention, a software package is provided, comprising: a first object providing a first set of a plurality of features, the first object being encrypted; and a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set, the second object being unencrypted.

In accordance with yet still another aspect of the present invention, a driver executable is provided, comprising: first code for accessing a requested file from a storage device; second code for detecting the presence of a predetermined identifier in the accessed file; and decryption code for decrypting at least a portion of the accessed file in response to detection of the identifier therein.

In accordance with a still further aspect of the invention, a software package is provided, comprising: a software object having a first set of features and a second set of features, the first set of features being encrypted and the second set of

features being unencrypted; and a signature readable by a predetermined executable serving to control access to the encrypted first set of features.

In accordance with a yet still further aspect of the present invention, a computer system is provided. The computer system comprises: a processor; a memory; an instruction input device; a storage device storing a file; an operating system; a driver executable; and a device driver serving to control access to the storage device; the instruction input device being operative to input a first request for access to the file; the operating system serving to control the processor to direct a second request for the file to the driver executable in response to the first request for access; the driver executable being operative in response to the second request to control the processor to direct a third request for the file to the driver; the driver being operative in response to the third request to control the processor to read the file from the device to the memory and thereupon return control of the processor to the driver executable; the driver executable being operative upon return of control thereto to control the processor to examine the file in memory to detect the presence of a predetermined identifier in the file and to decrypt at least a portion of the file in response to detection of the predetermined identifier therein.

The foregoing, as well as further aspects of the invention and advantages thereof, will be apparent in the following detailed description of certain illustrative embodiments thereof which is to be read in connection with the accompanying drawings forming a part hereof, and wherein corresponding parts and components are identified by the same reference numerals in the several views of the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a computer system having a single CPU;

Figure 2 is a flow diagram illustrating a method of producing software in the form of a package including a first object, a second object produced from the first object and usage authorization information governing use of the first and second objects;

Figures 3A through 3C illustrate image objects to be included in a package and produced in multiple versions each including a respectively different amount of information, produced by varying the amounts of noise therein;

Figures 3D through 3F illustrate multiple versions of the same image object of Figure 3A in which the amount of information in each version is varied by removing lines and/or portions of lines from certain versions;

Figures 3G through 3I illustrate multiple versions of the image object of Figure 3A in which the amount of information in each version is varied by filtering certain versions;

Figures 3J through 3L illustrate multiple versions of the image object of Figure 3A in which the amount of information is varied by encrypting portions of certain versions;

Figure 4A is a spectral diagram of a segment of an audio signal to be included as a data object in a package, while Figure 4B is a spectral diagram of another version of the segment having relatively less information than the segment of Figure 4A;

Figure 5A illustrates a data format for use in storing usage authorization information governing the use of various objects in a package, while Figures 5B and

5C are tables providing examples of the types of data included in such usage authorization information;

Figure 6 is a diagram illustrating a package produced according to the method of Figure 2 wherein a first object whose use is restricted is encrypted;

Figure 7 is a flow diagram of another method for producing software in the form of a package, wherein multiple objects are watermarked, compressed and encrypted and usage authorization information is watermarked and encrypted;

Figures 8A through 8D are used to describe methods for watermarking software carried out in the method of Figure 7; Figures 8A and 8B schematically illustrate a portion of an executable object and a portion of a code section, to be watermarked; Figures 8C and 8D schematically illustrate methods for watermarking executable objects and code sections of the type illustrated in Figures 8A and 8B;

Figures 9A through 9I are used to describe methods for compressing and encrypting software carried out in the method of Figure 7;

Figure 10 is a diagram of software in the form of a package produced by the method of Figure 7;

Figure 11A is a diagram of software in the form of a package including first and second executable or program objects; Figure 11B is a diagram of an executable notifier included in the package of Figure 11A, while Figure 11C is a diagram of the compressed program objects and access control information of the package of Figure 11A;

Figure 12 is a flow diagram of a method for secure distribution of software by data communication;

Figure 13 is a flow diagram of a method for secure distribution of software stored in a storage medium;

Figure 14 is a schematic diagram illustrating the use of a driver executable for controlling access to predetermined data objects in a computer system;

Figure 15 is a flow diagram of a method of printing a data object to which access is controlled;

Figure 16 illustrates the software package of Figures 11A through 11C when it is first loaded in the memory of a user's computer system;

Figure 17 illustrates portions of the software package of Figure 16 after the executable notifier has loaded a selected one of the program objects in running condition in the memory of the user's computer system; and

Figure 18 illustrates a method for controlling the usage of a given program by means of code in the executable notifier.

DETAILED DESCRIPTION OF CERTAIN PREFERRED EMBODIMENTS

With reference to Fig. 1, a computer system 100 is illustrated schematically having one or more central processing units (CPU) or processors 110, a display 120, other input/output (I/O) apparatus 130 (such as a network or internet connection and a keyboard and/or mouse), and a memory 140 in which executable files 150 and data files 160 may be loaded for execution or use by processor 110. The computer

system 100 also includes a non-volatile mass storage apparatus, such as a hard disk drive (not shown for purposes of simplicity and clarity).

Computer system 100 functions to produce software and to distribute the produced software to users, as well as to produce and distribute various other types of executables and data for controlling access to the produced software and carry out associated license purchasing transactions with users' computer systems. The manner in which system 100 carries out these functions will be apparent from the following discussion in connection with the associated drawings.

Figure 2 illustrates an exemplary method for producing a software package for distribution either on a record medium or by data communication, for example, via the world wide web or a dial-up service. The product thus generated includes multiple objects which either are data objects, such as media or multi-media objects, or are executable objects, such as games, applications or utilities. The method of Figure 2 is especially useful for generating try-and-buy packages.

In the method of Figure 2, a first object is used to produce one or more second objects in a step 210. In certain embodiments of this particular method, the one or more second objects are produced by removing features from the first object. In certain other embodiments, one or more first objects instead are produced from a second object by adding features to the second object.

Various embodiments of step 210 are illustrated in Figures 3A through 3L in which a first data object in the form of a digitized picture is used to produce multiple second objects having progressively less picture information.

In a first embodiment, a first picture object 310 shown in Figure 3A is used to produce a somewhat degraded version 316 as shown in Figure 3B by the addition of

noise to object 310. A further degraded version of object 310 is illustrated in Figure 3C as picture object 320 which is produced either through the addition of noise to object 310 or the addition of further noise to object 315.

A second embodiment of step 210 is illustrated in Figures 3D through 3F. The first picture object 310 is shown again in Figure 3D and is used to produce the moderately degraded version 325 as shown in Figure 3E by removing lines or portions of lines from the data object 310. A further degraded version 330 of object 310 shown in Figure 3F is produced by removing a relatively greater number of lines or portions of lines from object 310 or by removing still further lines from version 325. In still other embodiments the degraded versions are produced by removing multiple contiguous lines.

A further embodiment of step 210 is illustrated in Figures 3G through 3I in which the object 310 is subjected to low-pass filtering in order to remove fine details, such as the edges of objects. A moderately degraded version 335 as shown in Figure 3H is produced by low-pass filtering of object 310 with a relatively high frequency cut-off point, while a further degraded version 340 shown in Figure 3I is produced by low-pass filtering of object 310 with a relatively lower frequency cut-off point.

Yet another embodiment of the step 210 is illustrated in Figures 3J through 3L in which the object 310 is used to produce a somewhat degraded version 345 shown in Figure 3K by encrypting groups of contiguous horizontal lines with a first encryption key. When the object is displayed without decryption, it will appear as version 345 as shown in Figure 3K in which the encrypted portions are displayed as noise. Additional portions are encrypted to produce the still further degraded version

350 as shown in Figure 3L, the additional portions being encrypted with a second key or with the same key used to encrypt the portions shown in Figure 3K. Differently defined regions, such as blocks or vertical lines or regions, or else arbitrarily defined regions, may be selected for encryption.

In still other embodiments, either one, three or more degraded versions of a first picture object are produced.

In yet still further embodiments, further versions of a first picture object are produced by adding features thereto. For example, new elements can be added to the first picture object from other sources.

In other embodiments, the further versions are produced by substituting pixels having further information, such as finer detail or additional picture elements.

An embodiment of step 210 for producing multiple versions of an audio object is illustrated in Figures 4A and 4B. Figure 4A provides an exemplary spectral energy distribution 410 for a segment of a first audio object. A modified or degraded version of the Figure 4A segment is illustrated in the spectral energy distribution 420 of Figure 4B. In Figure 4B, the hatched-line frequency bands 430 represent portions of the energy spectrum which are removed, for example, by filtering, by removal of certain energy bins from an FFT transformed version of the segment, by removal of certain coefficients from a discrete cosine transformation of the segment, or otherwise. In still other embodiments, subbands of the audio signal in MP3 format are easily removed or encrypted to produce a degraded version thereof.

In the case of an executable object, step 210 is carried out in any of a number of ways. In one embodiment, the overall coding of a first executable object is modified to produce a modified executable object lacking one or more features of the

first. This may be done by removing the routines necessary to perform the disenabled features or bypassing such routines. In another embodiment, only one section of the first executable object is modified. For example, executable objects often are provided with resource sections which are readily modified to enable or disable its functions.

In the method of Figure 2, once the first and second objects have been prepared/obtained, the first object is encrypted to provide one means of controlling access thereto. In a try-and-buy transaction, as will be seen in greater detail below, the user is permitted free access to the second object having fewer than all of the features he needs, in order to assess his interest in acquiring rights to the first object which has all of the features he requires. Encryption is a relatively strong protection. The encryption step 220 is carried out so that a unique key or decryption executable is required to decrypt the first object. The key or decryption executable is produced by a server using selected information characteristic of the user's computer system, so that in order to decrypt the first object, both the key and decryption executable as well as the selected information are required. This key or decryption executable is stored in the system 100 and is not included in the package produced in the method of Figure 2. Rather, once the user has purchased the right to use the first object, the system 100 transmits the key or executable to the user's system which stores the key or executable in a package other than that of the first object.

In Step 230 of the Figure 2 method, data specifying permitted uses for each object and their price, if any, are produced and assembled according to each object. That is, for each object included in the package (or external to the package and

referenced thereby) and for each permitted user thereof, a record 510 such as that illustrated in Figure 5A is produced or accessed from storage in the system 100.

In a first field 520 of the record 510, data is provided identifying the object to which the record pertains. In a second field 530, the particular usage of the object for which the record is provided is identified. Examples of various usage types which can be identified in field 530 are listed in the table of Figure 5B.

A third field 540 of the record 510 specifies the extent of the permitted usage for the price specified in a fourth field 550 of the record 510. As indicated in the left-hand column of the table provided in Figure 5C, the extent of usage may be expressed in various ways, for example, by duration of use or numbers of usages. The price specified in the fourth field 550 corresponds to the authorized extent of usage, as can be seen from the table of Figure 5C. For example, if the extent of authorized usage is N times, the price may represent a specified amount of money for each time or for a number of times.

In step 240 of Figure 2, the first and second objects, and the usage authorization information are assembled in a package with a notifier section and, in packages having data objects, a signature. An exemplary structure for the package thus produced is illustrated in Figure 6, wherein the notifier, indicated as element 610 is arranged as the first section of the package.

The notifier 610 can take the form of one or more data objects or an executable object, depending on the type of package. Where the package contains data objects in the form of media objects such as digital images, video data or audio data produced in a standard format, the notifier includes at least one unencrypted and uncompressed image to be displayed to the user, as needed. As will be

explained in greater detail below, packages having data objects in standard formats preferably are accessed in the user's system by means of a driver executable in accordance with one aspect of the present invention. The first (or only) image stored in the notifier provides a message to the user that he needs to download the driver executable in order to make use of the data objects in the package. The notifier can also include a version of an object in the package having less information than such object, but which is unencrypted and readily displayed by the user's system. Once the driver executable has been downloaded and installed, it presents a dialog box to the user indicating the available objects, their authorized usages and the prices of each.

The driver executable is able to detect the type of accessed package as one including data objects requiring access control by the driver executable based on the package's signature which, in the embodiment of Figure 6, is appended at the end of the package. Where the driver executable detects that the accessed package has no recognizable signature or instead includes executable objects, it simply passes such packages on to the operating system without exercising any form of access control.

Packages including executable objects have notifiers including executables which serve both to control access to the executable objects in the package and to display necessary images to the user. These functions of the executable notifiers will be described in greater detail below. Since the driver executable is only required for accessing packages having data objects, there is no need to include a signature in a package having only executable objects.

Figure 7 illustrates another method for producing a software package including data or executable objects. In a first step 710 of the Figure 7 method, it is assumed that first, second and third objects, as well as an appropriate notifier and usage authorization information have been provided. In step 710, a watermark is placed in each of the foregoing objects, notifier and usage authorization information to provide a means of identifying the licensed user if any of these should be redistributed by him without authorization.

Data objects may be watermarked by any of a number of known methods which add data to the objects or modify the original data in order to embed the data of the watermark. However, watermarking of executable objects has, until now, been impractical, since any change to the code in the objects will interfere with the proper operation of the executable, and will likely render it inoperable. In addition, it is necessary for any such watermarking methodology for executable objects to enable the production of many variations in the watermark (at least one for each user) and, thus, in the anatomy of the executable, but wherein each variation of the executable is semantically equivalent to all other variations.

A further requirement is resistance to collusion attacks in which two or more dishonest purchasers combine their versions of the executable to derive one copy from which the watermark has been eliminated. To be considered resistant to such attacks, the number of different buyers whose individual revisions are all required to produce a watermark-free version or a version in which the watermark is useless, should be made impractically large.

In a further aspect of the present invention, watermarks are embedded in executable objects so that the watermarks are highly resistant to collusion attacks.

Advantageous watermarking techniques in accordance with certain features of the invention are illustrated in Figures 8A through 8D. In general, the method comprises: determining a location of at least one padding area in an executable object, and inserting a predetermined watermark in the at least one padding area. In certain embodiments, the watermark is encoded. A particularly advantageous form of encoding the watermark comprises including a plurality of software portions copied from the executable object or which mimic the same in the padding area to represent the encoded watermark.

Example of padding areas are provided with reference to Figures 8A and 8B. Figure 8A schematically illustrates a portion of an executable object in a storage medium, the object including a header 810, an executable code section 820 and a data section 830. The executable object of Figure 8A is formatted so that each section begins at a predetermined boundary. For example, the formats of an executable in the Win 32 platform would align the beginnings of the sections 820 and 830 at a 4 Kbyte boundary. Similar alignment conventions have been devised for other software formats, such as the Common Object File Format (COFF) used in UNIX and the Portable Executable format (PE) which is an extension of the COFF utilized in Windows™ platforms. The technique of aligning the beginning of each section at a predetermined boundary is convenient for programming purposes.

As a result, padding areas 812, 822 and 832 are formed between the ends of the sections 810, 820 and 830, respectively, and the following boundaries.

The padding areas either contain code or data which is unimportant or are simply empty.

Padding areas also exist within sections. With reference to Figure 8B, a schematic diagram of a code section is illustrated having instructions 1, 2, 3, ..., n, (n+1), ...

In this example padding areas are located after instruction 10 as well as after instruction (n+1). Such padding areas may be produced, for example, by a compiler which is designed so that each routine or calling point is arranged according to cache-line size. Codes designed to run on Intel™ processors include sequences of opcodes 0 x 90 (NOP) in these padding areas, so that it is relatively easy to locate such areas.

There are a number of ways to include watermarks in the padding areas as shown in Figures 8A and 8B. In certain embodiments, the watermark data is inserted in the padding areas in an unencoded form. Less knowledgeable users and licensees are not likely to take steps to locate and remove such watermarks. However, in more secure embodiments, the watermark is generated as a random number or selected as a pseudorandom number so that it is not easily recognized in order to remove or alter it.

However, padding areas associated with executable code sections or routines normally are filled with code which is not to be executed but rather serves only as filler. To substitute a random number for such codes would likely arouse suspicion by a would-be software pirate. Accordingly, in particularly advantageous embodiments, the watermark is encoded in software which mimics software present in the object before the watermark is inserted. An efficient way to carry out this method is to copy portions of the preexisting software (code or data) to represent the watermark. In certain embodiments the copied code is modified to encode the

watermark. Preferably, however, the copied portions are unmodified, but rather are selected to replace the existing contents of the padding area in a sequence representing the watermark. This is carried out in certain embodiments by selecting the copied portions according to their parities, so that a predetermined watermark can be recovered from the watermarked object simply by calculating the parities of the objects' contents until a known random or psuedo-random number constituting a predetermined watermark, is found.

Examples of this encoding technique are illustrated in Figures 8C and 8D. Figure 8C illustrates a technique for inserting watermarks in the padding areas 822 and 832 in the executable of Figure 8A. Once the padding areas 822 and 832 have been located, their contents are substituted with software from the adjacent segments 820 and 832 to encode the watermark. In order to encode the watermark in padding area 822, the parities of various code blocks from the code section 820 are determined. Then the blocks are inserted in the padding area 822 based on their parities, so that when the parities of these blocks are later determined, they reveal the watermark, preferably a random-generated or pseudorandom number.

As an example, if the watermark to be inserted in area 822 is 1011, a block 823 is selected having a parity of "1" and is inserted in area 822. Then a block 824 having a parity of "0" is inserted in the area 822, followed in turn by blocks 825 and 826 having parities "1", and "1", respectively. Similarly, blocks 833, 834, 835 and 836 are inserted in area 832 to continue the watermark.

Figure 8D provides an example of a method for encoding a watermark in the padding areas between routines in a code section of the type illustrated in Figure 8B. Routines Ø, 1 and 2, also identified by reference numerals 850, 860 and 870, are

separated by padding areas 852, 862 and 872. The watermark is inserted in the identified padding areas 852, 862 and 872 by copying portion of the sections, 850, 860 and 870 and inserting these in the padding areas. In the example of Figure 8D, an initial portion of routine Ø is inserted in a first portion of padding area 852 and a concluding portion of routine 1 is inserted in a final portion of padding area 852. Similar selections and insertions are made in padding areas 862 and 872. In this example, the watermark is encoded in the selection of the portions of the routines inserted in the various padding areas.

Various other encoding techniques are available. In other embodiments, NOP opcodes are replaced by opcodes having the same effect, just in case the NOP's are actually executed. For example, opcodes such as [mov al, a1], [mov c1, c1] [mov ah, ah] and [fnop] have the same effect as an NOP opcode and may be substituted therefor in order to encode a watermark.

In still other embodiments, the lengths of the blocks and/or fake routines are selected to encode all or part of the watermark.

In a subsequent step 720 of the method as illustrated in Figure 7, the first, second and third objects are compressed in accordance with still another aspect of the present invention. In a third step 730 of the method as shown in Figure 7, each of the blocks and assembly information representing the compressed first, second and third objects, as well as the Usage Authorization Information is encrypted. Preferably each is encrypted using a respective, unique key. The keys are not included in the resulting software package, but are retained to be distributed subsequently to authorized users.

The inventive compression technique carried out in step 720 of Figure 7, as well as the encryption step 730 thereof, are illustrated in greater detail in Figure 9A. As shown therein, software objects I through n, identified by 910, which may take the form of separate software packages, are subject to an inventive macrocompression method 920 to convert the objects I-n into one or more blocks 937 and assembly information objects 935, one for each object I-n, each indicating how to reconstruct the various strings of the respective one of the objects I-n from the one or more blocks 937. In summary, the macrocompression method 920, (1) produces matches of reference strings within the software objects 910 with comparison strings therein, the reference strings and the comparison strings having a predetermined minimum length, each comparison string within the same package as a matching reference string being separated therefrom by a predetermined minimum distance within the package, (2) expands the sizes of matching strings by including adjacent, matching software therein, and (3) forms compressed software objects comprising at least one software block corresponding to a selected one of the expanded, matching strings and assembly information indicating how to reconstruct others of the matching strings from the at least one software block. In certain embodiments, the software objects 910 comprise data. In other embodiments the software objects 910 comprise executables. While Figure 9A shows multiple objects I-n, the macrocompression method 920 also serves to compress a single object in certain embodiments.

The macrocompression method 920 is illustrated in greater detail in Figure 9B. String matching is carried out on the contents of the 1 through n objects 910, as indicated in a step 932. In certain embodiments, the string matching step is

facilitated by producing a hash head table grouping possible string matches together according to their hashing functions.

A hashing function of a given string calculates a hashing value based on the values of the bytes in the string. In certain embodiments of the present invention, a minimum string length of n bytes is employed and a hashing function is selected to calculate a hashing value for each string of n bytes. In general, the hashing value for each string of n bytes in each of the objects to be compressed is carried out, although this is not essential. In the general case, the hashing function is carried out for each string in the object $[p_0, p_1, \ldots, p_{n-1}]$, $[p_1, p_2, \ldots, p_n]$, . . $[p_i, p_{i-1}, \ldots, p_{i+n-1}]$, etc. where p_i represents a value of the i'th byte in the object. As the hashing value of each string having an offset j is determined, its offset j is added to a hash head table, indexed according to its hash value.

An exemplary hash head table is illustrated in Figure 9C and stores data identifying each string of n bytes in three objects M_1, M_2, and M_3 indexed according to the hashing value of each string. As shown in Figure 9C, all strings having a hashing value h equal to zero are identified by offset and object numbers in an initial record of the hash head table, and so on, until a final record is provided to identify those strings whose hashing value is a maximum among all hashing values in this case, h_{max}. It will be appreciated that the maximum possible number of different hashing values in this case will be $(L_1-n) + (L_2-n) + (L_3-n)$ which will occur in the event that each string yields a different hashing value. Accordingly, this is the maximum possible length of the hash head table for which memory space need be set aside in memory 140.

A particularly advantageous hashing function calculates the hashing value of each string of n bytes as a summation of their values:

$$h(j) = \sum_{i=j}^{j+n-1} p_i,$$

Wherein h(j) represents the hashing value of the jth string in the object and p_i is the value of the i'th byte of the object. One advantage flows from the commutative property of this function. That is, the function is commutative since it may be carried out using the byte value p_i in any arbitrary order. Consequently, in certain advantageous embodiments, once the hash value h(j) has been calculated as above for the string (p_j, p_{j+1}, . . . p_{j+n-1}), the hashing value for the next string is determined using relatively fewer operations (and processing time) as follows:

$$H_{(j+1)} = h_{(j)} - p_j + p_{j+n}$$

Also, the contents of most objects yield hashing values which are clumped, that is, unevenly distributed over the range of hashing values. This tends to reduce the usefulness of the hashing function as a means of separating strings which do not match from those which possibly do match. Where the invention implements a hashing function of the type:

$$h(j) = \sum_{i=j}^{j+n-1} p_i,$$

in certain embodiments utilizing this function, clumping is reduced by increasing the range of hashing values. That is, where the hashing function is carried out in the form illustrated above for the strings of length n bytes in an object having a total of L

bytes, the maximum number of different hashing values is (L-n). In the presently described embodiments, the hashing function is modified so that it takes the form:

$$h = K_1 h_1 \text{ (bytes a)} + K_2 h_2 \text{ (bytes n-a)},$$

wherein (bytes a) are the first (a) bytes within the string, so that a<n; (bytes n-a) represents the following (n-a) bytes within the same string; a selected one of K_1 and K_2 is equal to 1 and the other of K_1 and K_2 is an integer greater than 1; the function h_1 is calculated: $h_1 = \_$ (bytes a); and the function h_2 is calculated: $h_2 = \_$ (bytes n-a).

In a particularly advantageous form of this embodiment, memory space is conserved by assigning the value (255a+1) to the other of K_1 and K_2, so that the maximum value of h_1, which is (255a), immediately precedes the minimum non-zero value of K_2, which is (255a+1). As a consequence, there is no wasted memory space between these two possible hashing values.

Still other types of hashing functions may be employed in place of the above-described summation function. In particular, other commutative hashing functions are similarly advantageous. For example, an appropriate commutative hashing function h can take the form:

$$h(j) = p_j \times p_{j+1} \times \ldots \times p_{j+n-1},$$

or the form:

$$h(j) = p_j \oplus p_{j+1} \ominus \ldots \oplus p_{j+n-1}.$$

Since these functions are commutative, they can also be implemented in a simplified fashion as

$$H(j+1) = h(j) \text{ (inv\_op) } p_j \text{ (op) } P_{j+n},$$

where (op) represents a selected commutative operation (such as addition, multiplication or exclusive OR) and (inv_op) represents the inverse of such operation.

As noted above, the hash head table produces records containing possible matches. So, once the table is produced, the string matching process continues by searching for matches within each record of the table on the condition that, to qualify as an acceptable match, two matching strings within the same package (such as strings from the same file) must be separated by a predetermined minimum distance within the package. The following Table 1 provides an example of a possible sequence by byte values within a given package wherein each row of byte values is a continuation of the preceding row of values:

TABLE 1

Column

	1	2	3	4	5	6	7	8	9
Row 1	3	2	5	1	7	9	10	5	7
Row 2	10	11	31	2	5	1	7	9	10
Row 3	9	21	24	0	0	0	0	X_1	X_2
				. . .					
Row k	X_N	2	5	1	7	9	Y_1	Y_2	Y_3

From Table 1 it will be seen that four different strings of five bytes each have the hashing value h(j) = 24 where

$$h(j) = \sum_{i=j}^{j+4} p_i,$$

namely, (a) the string (a) from row 1, column 2 to row 1, column 6 having the values (2, 5, 1, 7, 9), (b) the string (b) from row 2, column 4 to row 2, column 8 having the values (2, 5, 1, 7, 9), (c) the string (c) from row 3, column 3 to row 3, column 7 having the values (24, 0, 0, 0, 0), and the string (d) from row k, column 2 to row k, column 6 having the values (2, 5, 1, 7, 9). While strings (a) and (c) have the same hashing values, they clearly do not match. Also, since to qualify as an acceptable match, the matching strings must be separated at least by a minimum distance if within the same package, strings (a) and (b), while matching, will not qualify if the minimum distance exceeds 11 bytes. Typically, the minimum distance will be substantially greater than 11 bytes in order to provide the ability to compress further through microcompression, as explained in greater detail below. If it is assumed that the matching strings (a) and (d) are separated at least by such minimum distance, therefore, strings (a) and (d) form a qualifying match.

An example of a search for matching strings in multiple packages is now provided with reference to Figure 9C. Packages M_1, M_2 and M_3 are illustrated therein having two types of exemplary strings of length n bytes, strings A and B. Where matching strings are contained in different packages, as in the case of strings B in packages M_1 and M_3, there is no need to require a minimum distance between them, as they would not be matched in the subsequent microcompression process. However, if it is assumed that the minimum distance between strings is q bytes as shown in Figure 9C, then the two strings A in M_1 will not form a qualifying match as they are offset by less than q bytes. However, the two strings A in M_2 will form a qualifying match as the strings of this pair are separated within package M_2 by more than q bytes.

Once all of the qualifying matches of a given type of string have been found, their identifiers are collected under a common group designation. When all of the qualifying matches of each type of string in the package or package being compressed, have been found and so grouped, the sizes of the matching strings are expanded by including adjacent matching bytes therein. An exemplary string expansion technique is explained in connection with Figure 9D which is a schematic illustration of a portion of a package or object having various types of strings K, L, P and Q, in which the matching process has located three qualified matching strings 1, 2 and 3 of type K. In order to expand these strings in one embodiment, each of the strings 1, 2 and 3 is expanded to the right by one byte and then the various combinations of matching string pairs (1 and 2, 2 and 3, 1 and 3) are compared for a match. If a match is still found for a given pair, the strings of the matching pair are repeatedly expanded by one byte and compared until a match is no longer found. At that point the identity of the pair and its matching length is entered in a table of the various string pair combinations, as shown in Figure 9E.

In other embodiments, the matching strings of each group instead are expanded to the left, while in still other embodiments the matching string are expanded in both directions.

Once the expanded matching pairs have been entered in the table of Figure 9E, they are removed from the hash head table.

When all of the matching strings have been expanded as explained above, the software blocks and the assembly information constituting the compressed package or packages are produced in a step 935 of Figure 9B. Preferably, representative ones of the largest expanded, matching strings are selected as the

software blocks, represented schematically at 937 in Figure 9B, and copied as indicated in step 939. Then the assembly information is produced as information referencing the remaining strings to all or a portion of each of the software blocks, as their contents correspond. This step is illustrated by the example of Figures 9D through 9F. As described above, in this example, the matches for each pair of strings (1, 2), (1, 3) and (2, 3) as seen in Figure 9D were separately expanded to produce the data shown in the table of Figure 9E. From Figure 9E it will be seen that the largest expanded, matching strings are strings 2 and 3. In this example, string 2 is selected as a software block for reference in reproducing each of the expanded strings 1, 2 and 3, since the contents of each is either contained in or corresponds to the contents of expanded string 2. The assembly information necessary to reconstruct strings 1, 2 and 3 is arranged in the table in Figure 9F. For example, string 1 is identified by its offset in the original package or object and its contents are reproduced from string 2 (software block) as the source, based on the offset within string 2 at which its contents is located (the source offset) and the length of such contents within string 2. In this manner, relatively large blocks of data from the original, uncompressed package or object can be represented as only a few bytes within the assembly information in the compressed form thereof, resulting in substantial reductions in the amount of data required to represent the package or object when it has been compressed according to the macrocompression method of step 920.

Where it is desired to remove information from a given package, for example, in order to produce images such as those illustrated by Figures 3E and 3K, or a sound segment such as that shown in Figure 4B, a technique as illustrated in

Figures 9G and 9H is advantageous. In Figure 9G, it is assumed that a segment B is to be removed from a package P and substituted with zero values throughout, or else by some other constant or by noise. As shown in Figure 9G, the segment B is located at an offset 2 and has a length L_B. Segment B is flanked by a segment A located at an offset 1 and a segment C located at an offset 3.

The desired result is illustrated in Figure 9H wherein the segment B is replaced by zero-value data, represented by double cross-hatching. The resulting package P' is achieved by specifying the source for each of the three segments, as shown in the table T of Figure 9H, wherein the source for the segment at offset 2 extending for a length L_B is specified as the constant value zero, which thus replaces the original contents of segment B.

Once the new package P' has thus been specified, macrocompression is carried out only for the first and third segments at offsets 1 and 3. This is achieved preferably by constructing a hash head table only for the strings in the first and third segments A and C, and prohibiting the use of any strings in the second segment in producing the hash head table. Thereafter, both the macrocompressed segments at offsets 1 and 3 and the uncompressed segment at offset 2, may be compressed by microcompression as discussed below.

This technique is useful not only in producing degraded objects and packages, but also for preparing a partially compressed package or object having an uncompressed portion which is thus readily modified.

Returning to Figure 9A, after the macrocompression method 920 has been carried out, the resulting blocks and assembly information are further compressed by microcompression, as indicated by step 950. As used herein, microcompression

identifies a software compression technique which compares strings having a predetermined maximum size with other strings of the same size which are located no more than a predetermined distance or window from one another in the same package, in order to eliminate redundant strings. An example of a microcompression executable is the PK Zip™ utility. The result of microcompression is further compressed assembly information AI* and software blocks BLKS* as shown in Figure 9A.

Preferably, the window used in the microcompression process is smaller than the minimum distance between qualified matching blocks in the macrocompression method of step 920. In this manner, different strings are compared in the two compression techniques, thus affording more effective compression. In accordance with another aspect of the invention, a method of compressing software in one or more packages comprises: producing first compressed software by matching strings selected so that matching strings within the same package are separated at least by a minimum predetermined distance within the package, and producing second compressed software by matching strings of the first compressed software within the same package and within a maximum predetermined distance of one another. Preferably, the minimum predetermined distance is greater than the maximum predetermined distance.

The further compressed assembly information AI* and software blocks BLKS*, along with the Usage Authorization Information, are then encrypted in a step 960 so that the Usage Authorization Information and the assembly information AI* for each object 1 through n, is encrypted using a respectively different encryption key. Preferably, each of the blocks BLKS* is also encrypted with a respectively different

encryption key. As will be explained in greater detail below, each encryption key is produced based on information characteristic of the user's computer system, and so that decryption requires the use of both the encryption key and such characteristic information. This ensures that the encrypted information and software cannot be decrypted using a system other than the user's particular system.

In accordance with a further aspect of the invention, a method of encrypting software representing a plurality of compressed objects is provided. The software includes at least one software block and assembly information for each of the objects, the assembly information for each object enabling the reconstruction thereof from the at least one software block. The method comprises: encrypting each of the software blocks with an encryption key; and encrypting the assembly information for each object using a respectively different encryption key. Preferably, a respectively different encryption key is used to encrypt each of the software blocks.

The encrypted assembly information AI** and the encrypted software blocks BLKS**, together with the encrypted Usage Authorization Information, are formed into a single composite package 970.

In a final step 740 of the method as shown in Figure 7, an appropriate notifier and signature (if necessary) are added to the encrypted blocks, assembly information and usage authorization information to complete the software package.

An advantageous format for the software package is illustrated in Figure 10, wherein the notifier 1010 is placed at the head of the package. Where the package includes data objects, placing the notifier at the head of the package will result in the display of the correct image when the package is first accessed. Where the package includes executable objects, the first portion of the package may simply be a header

indicating the entry point for an executable notifier located anywhere in the package. Packages including data objects have a signature 1020 appended thereto. Placing the signature at the end of the package enables the executable driver to readily locate the signature in order to determine if it is to exercise access control over data objects in the package as well as perform other functions, such as decryption and decompression of the data objects. Although the signature 1020 is shown appended at the end of the package, in the alternative, it may be located elsewhere, such as at the beginning of the package or after the notifier.

Between the notifier 1010 and the signature 1020, the encrypted sections 1030 (indicated by cross-hatching) are arranged in a predetermined order to be accessed by the driver executable or the executable notifier, as the case may be.

Figures 11A through 11C illustrate the structure of a software package including multiple program objects. Figure 11A provides an overall view of the software package illustrating the arrangement of an executable notifier 1110 at the head of the package, an optional signature section 1120 at the end of the package, with encrypted and compressed program objects 1 and 2 and encrypted access control information 1130 arranged between the executable notifier 1110 and the signature section 1120.

The executable notifier 1110 is illustrated in greater detail in Figure 11B. As shown therein, the executable notifier 1110 includes a header section 1135 at the beginning of the software package, followed in turn by an executable code section 1140 and a data section 1145. The data section 1145 is followed sequentially by a resource section 1150 and an import table 1155. The resource section 1150 supplies various resources which may be employed by the executable code of

section 1140, such as dialog boxes or menus. The import table 1155 includes links to various routines supplied by the operating system, such as print, copy, readfile, createfile, etc.

Figure 11C illustrates the encrypted portions of the software package, including the encrypted access control information 1160 and the compressed program objects in the form of N blocks 1165 and respective assembly information sections 1170 for each program object.

With reference again to Figure 11B, the executable code section 1140 of the executable notifier 1110, in general, exercises control over access to the program objects 1 and 2 and performs certain ancillary functions, as follows:

(1) When the user's system first loads the software package in memory, the executable code section 1140 runs a setup routine utilizing displays and dialog boxes supplied from the resource section 1150. The setup routine performs normal setup functions, such as a display of the relevant user license and securing the user's agreement to the license terms. The executable code section 1140 refers to information in the operating system of the user's computer to determine the language (e.g., English, French, German) in which the displays and dialog boxes are presented.

(2) The executable code section 1140 solicits and evaluates the user's requests for access to the program objects. This is achieved by displaying a dialog box when the software package is accessed by the user. The dialog box explains the user's options, such as which programs and/or program options are available without charge, which are available for a fee and which of the latter have been purchased and are still available to be used. To provide such a display, the

executable code section references both the access control information section 1160 (after decrypting section 1160) and a purchase status file which is produced when the user purchases rights to use one or more objects.

(3) Where a requested use is either free, or already purchased, if not free, the executable code section 1140 decrypts and decompresses the relevant program or data object, and then loads it in memory to be run so that the requested use may be carried out. The section 1140 prevents access to unavailable uses by hooking the functions referenced in the import table of the running program object to control routines in the executable code section 1140, as explained below.

(4) The executable code section 1140 serves to deter dump attacks by erasing from memory certain necessary information from the program object when it loads the program object in running format in memory. Consequently, even if the decrypted and decompressed program object is somehow copied from the memory to some storage device, it could not be reloaded in running format in memory and, thus, is useless after a dump attack.

It will be understood that the executable code section 1140 functions as a "wrapper" or access control executable but without being susceptible to various types of attacks that prior art wrappers have been subject to.

Fig 12 is a flow diagram of a method for secure distribution of software by data communication. For the purposes of Figure 12, it will be assumed that a user's computer has been connected to a server computer by a data communication channel, such as the internet. According to an initial step 1210 in Fig. 12, the server sends a software product, which is either an executable object or a data object, to

the user's computer, in response to a request sent to the server from the user's computer.

If the software product is a data object, the user's computer will require a driver executable in order to make use of the data. If the user's computer lacks the required driver executable, the user's attempt to access the data object will result only in the display of a notification to download the driver executable from the server computer. When the server computer receives such a request, it responds as indicated in step 1220 by sending the driver executable to the user's computer where it is installed to operate between its operating system and the appropriate disk or other mass storage driver thereof, as explained below in connection with Figure 14.

Then, at step 1230, and in response to input from the user, an access control executable portion of the software product (if an executable object) or of the driver executable (if the software product is a data object) causes the user's computer to transmit a purchase request for partial or full access to the software product, and the server receives the purchase request. Step 1240 follows, at which the server sends to the user's computer a program which generates system identification information based on data that is specific to the user's computer. For example, the data used to generate the system identification information may include serial numbers of such components of the user's computer as the hard disk, the network interface card, the motherboard, and so forth. The user's computer then sends to the server the resulting system identification information, as well as information, such as a credit card number, which is required to complete the transaction. This information is received at the server, as indicated at step 1250.

Following step 1250 is step 1260, at which the server validates the credit card information and generates a decryption key and/or a decryption executable program on the basis of the system identification information received from, and specific to, the user's computer. According to one method of implementing the invention, the required decryption key is split into two parts, of which one part is calculated in the server, and the other is calculated in real time in the user's computer, using the data which is specific to components of the user's computer. The decryption key and/or decryption executable program are then transmitted to the user's computer from the server, as indicated at step 1270. The decryption key and/or decryption executable program are then used in the user's computer to decrypt the software object to which the user has just purchased usage rights. In certain embodiments, a watermark is added to the software object to store data indicative of the transaction in which the usage rights were purchased.

According to certain embodiments of the invention, the software product sent at step 1210 includes three objects, of which a first object has all of the features of a second object plus at least one additional feature. A third of the three objects has all of the features of the first object plus at least one additional object. Access to the second object is free, but access to the first and third objects requires two separate payments. If a payment arrangement is made for both of the first and third objects, the server computer provides different access control codes, such as different decryption keys, for the first and third objects, respectively. The different control codes are based on different respective information characteristic of the user's computer system.

Fig. 13 is a flow diagram of a method for secure distribution of software stored in a storage medium.

According to a first step 1310 in Fig. 13, software which is distributed on a storage medium is acquired by the user of a computer and installed on the user's computer. This step 1310 may have taken place a substantial period of time prior to the subsequent steps. Next, at step 1320, a server computer receives a request from the user's computer to purchase partial or full access to a software object which was installed on the user's computer in step 1310. It again is assumed that the user's computer has been connected by a communication channel to the server. Preferably the information received by the server at step 1320 includes an identification code (such as a CD serial number) which identifies the particular storage medium on which the software was distributed.

Following step 1320 are steps 1330, 1340, 1350 and 1360. These steps may be identical to steps 1240-1270 which were described above in connection with Fig. 12, except that the decryption key generated by the server at step 1350 may be based in part on the storage medium identification code. In view of the previous discussion of the corresponding steps in Fig. 12, no further explanation of Fig. 13 is necessary.

Fig. 14 is a schematic diagram illustrating the use of a driver executable controlling access to data objects stored in a computer system. The software architecture illustrated in Fig. 14 includes a media player application 1405 which is provided to read or play data objects such as images. Also included is a conventional operating system 1410 and a driver executable program 1415 of the

type referred to in connection with step 1220 in Fig. 12, or which is distributed on the storage medium referred to at step 310 in Fig. 13.

Also illustrated in Fig. 14 are a conventional driver program 1420 which is provided for managing a storage device, and a storage device 1425 on which one or more data objects are stored.

Fig. 14 also illustrates a process by which a data object stored on the storage device 1425 is accessed by the media player application 1405, as well as a process for requesting printing of the accessed object.

When the user of the computer system enters an input to request access to a data object stored on the storage device 1425, a request to that effect is passed from the media player application 1405 to the operating system 1410, as indicated at reference numeral 1430 in Fig. 14. In response to the request 1430, the operating system 1410 passes a second request (represented by reference numeral 1432) to the driver executable 1415. In response to the request 1432, the driver executable 1415 passes a third request (reference numeral 1434) to the storage device driver 1420. In response to the request 1434, the storage device driver 1420 retrieves the desired data object from the storage device 1425. The desired object is then passed from the storage device driver 1420 to the driver executable 1415 either in encrypted form, as indicated at 1436, or in unencrypted form. If the user has satisfied the condition for access to the data object (e.g., by paying the purchase price for access), then the driver executable decrypts the encrypted data object and passes the decrypted data object to the operating system 1410, as indicated at 1438. The decrypted data object is then passed from the operating system to the media player application, as indicated at 1440.

If the user wishes to print the data object, then a request 1442 is passed from the media player application to the driver executable, which then passes another print request 1444 to the operating system.

Fig. 15 is a flow diagram which shows additional details of a method of printing a data object to which access is controlled. In response to input from the user of the computer, the media player transmits the print request (reference numeral 1442 in Fig. 14), as represented by step 1510 in Fig. 15, to the driver executable. The driver executable then examines the object to determine whether identifier data such as a signature is present in the object to indicate that printing of the object is subject to some restriction (step 1520). If at step 1520 no such identifier is found, then, as indicated at step 1530, the driver executable provides the data object in an unmodified form to the operating system.

If at step 1520 the driver executable finds the signature which identifies the object as one for which access is controlled, step 1540 follows. At step 1540 the driver executable saves or modifies the target address in the media player application to which the application directs calls for a print routine. Consequently, as indicated at step 1550, when the media player calls a print routine, the call is directed to the driver executable. However, if step 1540 has already been carried out as a result of a previous print request from the media player, this step need not be repeated.

At step 1560, and in response to the call for the print routine from the media player application, the driver executable determines whether the customer has satisfied the conditions required to authorize printing of the data object. If not, the driver executable causes the computer system to display a suitable notice to indicate

to the user that printing is denied, and to invite the user to purchase the right to print the data object (step 1570), as described hereinabove.

If at step 1560 the driver executable determines that printing is authorized, then the driver executable calls the print routine provided by the operating system (step 1580).

Fig. 16 illustrates the software package of Figs. 11A-11C when the software package is first loaded into the working memory of a user's computer system. As before, the executable notifier 1110 is made up of a header section 1135, followed in turn by a executable code section 1140, a data section 1145, a resource section 1150 and an import table 1155.

Following the executable notifier 1110 are the encrypted and compressed program objects and encrypted access control information, all indicated by reference numeral 1130, and the signature section 1120, which were referred to above in connection with Fig.11A.

If the user requests access to one of the program objects, say object 1, and if access to the object has been authorized, then the executable notifier decrypts and decompresses the program object and causes the program object to be written in executable form as indicated in Figure 17. As seen from Fig. 17, the decrypted, decompressed program object includes a header section 1710, followed in turn by an executable code section 1720, a data section 1730, a resource section 1740, and an import table 1750.

After the program object has been written in memory in executable form as shown in Fig. 17, the executable notifier modifies the program object in a manner to defeat dump attacks. This is achieved by erasing or modifying certain portions of the

program object after it is written in memory. In certain embodiments, one or more of the program object's relocation information, directory pointers or its entry point pointer are erased or modified for this purpose. In other embodiments, one or more of the references to exterior routines in the import table of the program object are modified to enable the executable notifier to control access to such routines. This modification of the program object is referred to as "hooking" routine calls by the program objects. This is done by modifying the import table 1750 so that routine calls are routed through the executable notifier instead of directly to the operating system. Details of the "hooking" process will now be described with reference to Figure 18.

As indicated at 1810 in Fig. 18, the executable notifier erases portions of the import table that identify the routines to be called by the corresponding virtual address such as "read file", "create file", or "print". Instead of addresses to the operating system routines, the executable notifier inserts virtual addresses in the import table which cause jumps to the code section 1140 of the executable notifier. The code section 1140 is programmed to interpret each jump to determine the particular routine requested by the program object. The executable notifier then determines whether the user has satisfied the conditions to perform the function in question. If so, the executable notifier calls the appropriate routine in the operating system. To elaborate details of the "hooking" process shown in Fig. 18, the executable notifier stores in an address record portion of the import table 1750 addresses within the executable notifier in place of the addresses of the relevant routines in the operating system. Instead of erasing part of, and making substitutions for, the import table 1750 of the program object, the executable notifier

may erase and substitute for other portions of the program object, such as relocation information, a directory pointer or an entry point pointer.

The above description of the invention is intended to be illustrative and not limiting. Various changes or modifications in the embodiments described may occur to those skilled in the art. These can be made without departing from the spirit or scope of the invention.

CLAIMS

1. A method of securely distributing software with limited usage rights, comprising:

supplying software for distribution to a user, the software including an access control object for preventing at least some usage thereof on a computer system without the use of a first access control code;

producing the first access control code based on selected information characteristic of a predetermined computer system; and

supplying the first access control code to the predetermined computer system to enable the at least some usage of the software.

2. The method of claim 1, wherein the step of supplying software comprises supplying the software to the predetermined computer system, the software having a first object and a second object, the access control object comprising an access control executable controlling access to the first and second objects by referencing the first access control code and the selected information in the computer system.

3. The method of claim 2, wherein the step of supplying the first access control code comprises supplying a usage authorization package including the first access control code and information identifying authorized usages of the software, the access control executable being operative to reference the usage authorization package in controlling access to the first and second objects, the software being operative to store the usage authorization package apart from the first and second objects.

4. The method of claim 2, wherein the first object provides a first set of a plurality of features, the second object provides a second set of a plurality of features including some, but less than all, of the features included in the first set and the access control executable is operative to prevent access to the first object in the absence of the first access control code and the selected information but to enable access to the second object without reference to the first access control code or the selected information.

5. The method of claim 4, wherein the software includes a third object providing the first set of a plurality of features together with a third set including at least one feature not included in the first set, the executable being operative to prevent access to the third object in the absence of a second access control code different from the first access control code and further selected information characteristic of the predetermined computer system, the second access control code being produced based on the further selected information, the method further comprising:

supplying the second access control code to the predetermined computer system.

6. The method of claim 2, wherein the access control executable comprises a wrapper for the first and second objects.

7. The method of claim 1, wherein the first access control code is a decryption key produced from the selected information.

8. The method of claim 1, wherein the first access control code is an executable required for decrypting at least a portion of the software.

9. The method of claim 1, wherein the first access control code is a watermark in an object supplied to the predetermined computer system.

10. The method of claim 1, wherein the software includes transaction information relating to a transaction by which the software is supplied to the user, and the access control object is operative to prevent the at least some usage of the software in the absence of the transaction information in the software.

11. The method of claim 1, wherein the transaction information is supplied as a watermark in the software.

12. The method of claim 1, further comprising the steps of storing the first access control code at a location in the computer system apart from a location at which the software is stored therein.

13. The method of claim 1, further comprising receiving at a server a request from the user to purchase rights to predetermined usage of the software, receiving at the server the selected information characteristic of the predetermined computer system, obtaining payment information from the user assuring payment for the rights and supplying the first access control code from the server to the predetermined computer system in response to receipt of the payment information.

14. The method of claim 13, further comprising supplying a system information collection code from the server to the predetermined computer system, the system information collection code being operative to obtain the selected information characteristic of the predetermined computer system.

15. The method of claim 13, wherein the software includes data defining a notifier which the software causes to be displayed by means of the predetermined computer system, the notifier conveying information required by the user for ordering rights to predetermined usage of the software and enabling entry of first transaction information required for the purchase of the rights, the software being operative to obtain the selected information from the predetermined computer system in response to entry of the payment information and to cause the predetermined computer system to transmit the selected information and the first transaction information to the server.

16. The method of claim 15, wherein the software is operative to reference information in an operating system of the predetermined computer system identifying a language selected for providing outputs to a user and to cause the software to provide such outputs in the selected language.

17. The method of claim 15, wherein the software is supplied as a software copy on a storage medium for distribution to the user, the software including an identification code identifying the software copy, the software is operative to transmit the identification code to the server, the server being operative to produce the first access control code based on the identification code.

18. The method of claim 15, wherein the software is supplied by data communication from a server to the predetermined computer system in response to a request, the request including second transaction information, the server being operative to insert transaction identification information in the software based on the second transaction information.

19. The method of claim 18, wherein the server is operative to insert the transaction identification information in the software as a watermark.

20. The method of claim 18, wherein the server is operative to produce the first access control code based on the second transaction information.

21. The method of claim 20, wherein the server is operative to supply the first access control code with an identifying watermark.

22. The method of claim 1, wherein the software comprises a first data object.

23. The method of claim 22, wherein the first data object includes a first set of features and a second set of features, the first set of features being encrypted and the second set of features being unencrypted, and wherein the step of supplying software comprises supplying a driver executable to the predetermined computer system, the driver executable including first code for accessing the first data object and decryption code for controlling decryption of the first data object.

24. The method of claim 22, wherein the first data object is encrypted, the step of supplying software including supplying a driver executable to the predetermined computer system, the driver executable including first code for accessing the first data object and decryption code for decrypting the accessed first data object.

25. The method of claim 24, wherein the driver executable is operative to receive a request for the first data object from an operating system of the predetermined computer system and to transfer the request to a preexisting driver of

the predetermined computer system, the driver executable being further operative to receive the first data object from the preexisting driver, to decrypt the first predetermined object and supply the decrypted first predetermined object to the operating system.

26. The method of claim 25, wherein the first data object includes a predetermined identifier therein and the driver executable includes second code for detecting the presence of the predetermined identifier in the first data object, the decryption code being operative to decrypt the first data object in response to the presence of the predetermined identifier therein.

27. The method of claim 26, wherein the driver executable is operative to transfer a file from the preexisting driver to the operating system without modification in the absence of the predetermined identifier in the file as received from the preexisting driver.

28. The method of claim 24 wherein the first data object provides a first set of a plurality of features, the software further comprising a second data object providing a second set of a plurality of features including some, but less than all, of the features included in the first set, and a usage authorization package including information identifying authorized usages of the first and second data objects, the driver executable being operative to selectively enable usage of the first and second data objects based on the usage authorization package.

29. The method of claim 28, wherein the driver executable is operative upon a first request for access to the first or second data object to return a dialog box

object for displaying a dialog box to the user, the dialog box providing the user with options for accessing the first and second data objects on a pay and/or no-pay basis.

30. The method of claim 29, wherein the driver executable is operative upon the first request for access to the first or second data object to reference information in the operating system identifying a display language selected for producing displays to a user and to provide the dialog box with text in the display language.

31. The method of claim 22, wherein only a portion of the first data object is encrypted, and wherein the step of supplying software comprises supplying a driver executable and a usage authorization package to the predetermined computer system, the usage authorization package including information identifying authorized usages of the first data object, the driver executable being operative to access the first data object and to transfer the first data object to an operating system of the predetermined computer system, wherein the driver executable selectively decrypts the portion of the first data object before transferring the first data object to the operating system based on the usage authorization package.

32. The method of claim 22, wherein the step of supplying software comprises supplying a driver executable to the predetermined computer system, the driver executable including first code for accessing the first data object in response to a request from an operating system of the predetermined computer system and being operative to determine whether a requested action utilizing the first data object is authorized, the driver executable being operative to block execution of the requested action when the same is not authorized.

33. The method of claim 32, wherein the driver executable is operative to block execution of the requested action by hooking a routine of an external executable required for performing the requested action.

34. The method of claim 22, wherein the software comprises a second data object, wherein the method further comprises producing the second data object from the first data object by reducing the information content of the first data object.

35. The method of claim 34, wherein the second data object is produced by eliminating data from the first data object.

36. The method of claim 34, wherein the second data object is produced by adding noise to the first data object.

37. The method of claim 34, wherein the second data object is produced by filtering the first data object.

38. The method of claim 34, wherein the second data object is produced by encrypting portions of the first data object.

39. The method of claim 22, wherein the software comprises a second data object, and wherein the method further comprises producing the first data object from the second data object by adding data to the second data object.

40. The method of claim 1, wherein the software comprises a first executable object.

41. A software package comprising:

a first object providing a first set of a plurality of features;

a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set; and

an access control portion affording selective access to the first software object and/or the second software object.

42. The software package of claim 41, wherein the first and second objects are executables.

43. The software package of claim 41, wherein the first and second objects are data objects.

44. A software package, comprising:

a first object providing a first set of a plurality of features, the first object being encrypted; and

a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set, the second object being unencrypted.

45. A driver executable, comprising:

first code for accessing a requested file from a storage device;

second code for detecting the presence of a predetermined identifier in the accessed file; and

decryption code for decrypting at least a portion of the accessed file in response to detection of the predetermined identifier therein.

46. A software package, comprising:

a first executable object; and

a wrapper for the first executable object, the wrapper being operative to erase predetermined software from the first executable object when it has been loaded in running format in memory.

47. The software package of claim 46, wherein the predetermined software comprises an import table.

48. The software package of claim 46, wherein the predetermined software comprises relocation information.

49. The software package of claim 46, wherein the predetermined software comprises a directory pointer.

50. The software package of claim 46, wherein the predetermined software comprises an entry point pointer.

51. The software package of claim 46, further comprising usage information defining at least one permitted use of the first executable object, the wrapper being operative to reference the usage information to control usage of the first executable object.

52. The software package of claim 51, wherein the first executable object is operative to access a file external to the software package and the wrapper is operative to control usage of the first executable object to access the external file based on the usage information.

53. The software package of claim 52, wherein the external file is encrypted and the wrapper is operative to control decryption of the external file.

54. The software package of claim 51, wherein the wrapper is operative to detect an unauthorized request for access to the external file and upon such detection to control the display of a dialog box to a user soliciting payment for the requested access.

55. A software package, comprising:

a first executable object; and

a wrapper for the first executable object;

the first executable object being operative, while running, to access a feature of the wrapper;

the wrapper being operative to supply the feature to the first executable object when the feature is accessed thereby.

56. The software package of claim 55, wherein the first executable object is operative to access a print control feature of the wrapper, and the wrapper is operative to control execution of a print feature in response to access of the print control feature thereof by the first executable object.

57. The software package of claim 55, wherein the first executable object is operative to access a copy control feature of the wrapper, and the wrapper is operative to control execution of a copy feature in response to access of the copy control feature thereof by the first executable object.

58. The software package of claim 55, wherein the first executable object is operative to access a read-file control feature of the wrapper, and the wrapper is

operative to control execution of the read file feature in response to access of the read-file control feature thereof by the first executable object.

59. The software package of claim 55, wherein the wrapper is operative to control execution of a decryption feature in response to the access by the first executable object.

60. The software package of claim 55, wherein the first executable object includes at least one record storing an address of the wrapper, the first executable object being operative to access the feature of the wrapper by transferring program execution control to the address of the wrapper.

61. The software package of claim 60, wherein the wrapper is operated to supply the feature by calling an executable routine external to the software package.

62. A software package, comprising:

a first executable object, and

a wrapper for the first executable object;

the first executable object being operative to call a predetermined feature external thereto;

the wrapper being operative upon a call of the predetermined feature by the first executable object to transfer program execution control to a predetermined address within the wrapper to control access by the first executable object to the predetermined feature.

63. The software package of claim 62, wherein the first executable object stores an address record for an address of software providing the predetermined feature, and the wrapper is operative to store an address of the wrapper in the address record.

64. The software package of claim 63, wherein the first executable object stores the address record in an import table including a plurality of address records for calling a plurality of routines external to the software package, the wrapper being operative to resolve the addresses of the plurality of routines and to insert a predetermined one of the routine addresses as the address record in the import table.

65. The software package of claim 64, wherein the wrapper is operative to insert each of a plurality of the routine addresses in a respective one of the plurality of address records in the import table.

66. A software package, comprising:

a software object having a first set of features and a second set of features, the first set of features being encrypted and the second set of features being unencrypted; and

a signature readable by a predetermined executable serving to control access to the encrypted first set of features.

67. The software package of claim 66, wherein the software object comprises an executable.

68. The software package of claim 66, wherein the software object comprises a data object.

69. The software package of claim 68, wherein the data object comprises music data.

70. The software package of claim 69, wherein the music data is encoded as a plurality of frequency coefficients produced by a discrete cosine transform, the coefficients including relatively low frequency coefficients and relatively high frequency coefficients, the relatively high frequency coefficients being encrypted and the relatively low frequency coefficients being unencrypted.

71. The software package of claim 69, wherein the music data is arranged in a plurality of subbands, at least some of the subbands of data being encrypted and others of the subbands being unencrypted.

72. The software package of claim 69, wherein the music data is frequency domain data, at least some of the frequency domain data being encrypted and other of the frequency domain being unencrypted.

73. The software package of claim 68, wherein the data object comprises image data.

74. The software package of claim 73, wherein the image data is encoded as a plurality of frequency coefficients produced by discrete cosine transform, the coefficients including relatively low frequency coefficients and relatively high frequency coefficients, the relatively high frequency coefficients being encrypted and the relatively low frequency coefficients being unencrypted.

75. The software package of claim 73, wherein the image data comprises color components, at least some of the color components being encrypted.

76. The software package of claim 75, wherein at least one of the color components is unencrypted.

77. The software package of claim 75, wherein the image data further comprises an unencrypted luminance component, and wherein all of the color components are encrypted.

78. The software package of claim 73, wherein the image data comprises line data, at least some of the line data being encrypted.

79. The software package of claim 73, wherein the image data is arranged in a plurality of lines, at least portions of some of the lines being encrypted and at least other portions of other lines being unencrypted.

80. The software package of claim 73, wherein the image data is arranged at least in part as a plurality of blocks, at least one of the blocks being encrypted.

81. An executable object, comprising:

a first code portion comprising first predetermined instructions; and

a second code portion comprising loading instructions required for loading the first code portion in a memory of a computer system to be programmed thereby, the second code portion being operative to control the computer system to erase the loading instructions from memory upon loading the first code portion in memory.

82. A computer system, comprising:

a processor;

a memory;

an instruction input device; and

an executable object stored in the computer system, the executable object having a first code portion comprising first predetermined instructions for execution by the processor, and a second code portion including loading instructions, the processor being operative upon receipt of a predetermined instruction from the instruction input device to load the second code portion in the memory, the processor being operative under the control of the loading instructions to load the first code portion in the memory and operative under the control of the second code portion to erase the loading instructions from the memory upon loading the first code portion in memory.

83. A computer system, comprising:

a processor;

a memory;

an instruction input device; and

a software package stored in the computer system, the software packing having a first object providing a first set of a plurality of features, a second object providing a second set of a plurality of features including some, but less than all, of the features included in the first set, and an access control portion; the

processor being operative to load the software package in the memory, the processor being further operative to request access to a selected one of the first and second objects in response to a predetermined instruction from the instruction input device, the access control portion being operative to selectively control access to the selected object.

84. A computer system, comprising:

a processor;

a memory;

an instruction input device;

a storage device storing a file;

an operating system;

a driver executable; and

a device driver serving to control access to the storage device;

the instruction input device being operative to input a first request for access to the file;

the operating system serving to control the processor to direct a second request for the file to the driver executable in response to the first request for access;

the driver executable being operative in response to the second request to control the processor to direct a third request for the file to the driver;

the driver being operative in response to the third request to control the processor to read the file from the device to the memory and thereupon return control of the processor to the driver executable;

the driver executable being operative upon return of control thereto to control the processor to examine the file in memory to detect the presence of a predetermined identifier in the file and to decrypt at least a portion of the file in response to detection of the predetermined identifier therein.

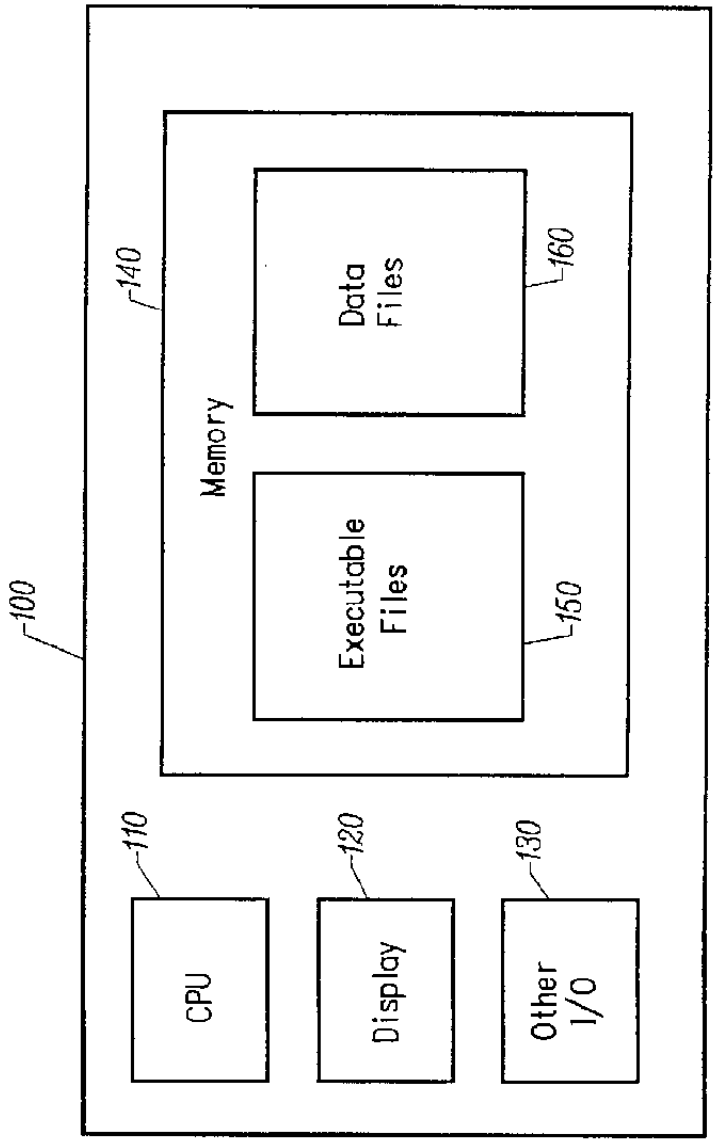

100
140
Memory
Data Files
160
Executable Files
150
110
CPU
120
Display
130
Other I/O

FIG. 1

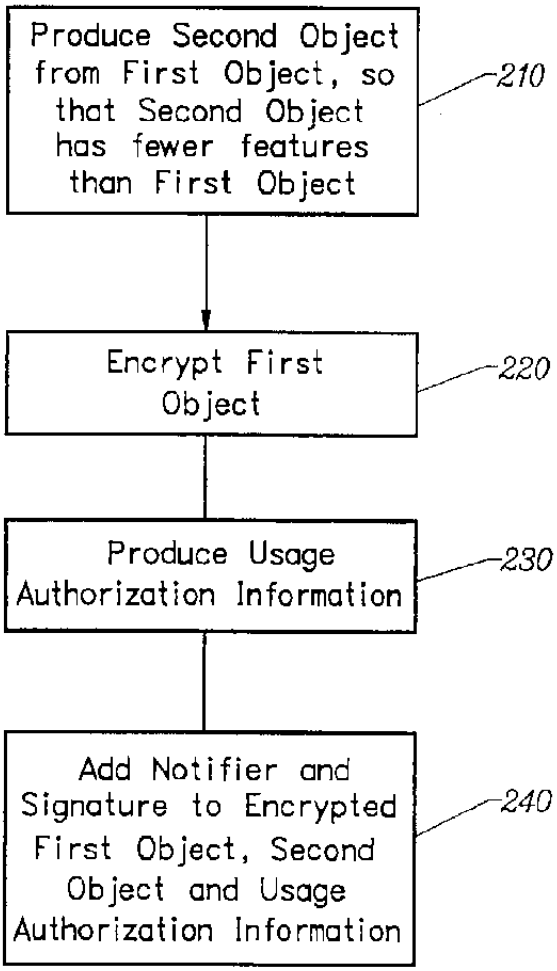

Produce Second Object from First Object, so that Second Object has fewer features than First Object
210
Encrypt First Object
220
Produce Usage Authorization Information
230
Add Notifier and Signature to Encrypted First Object, Second Object and Usage Authorization Information
240

FIG. 2

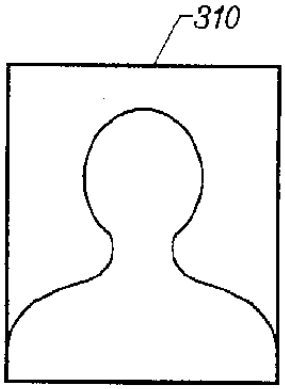


FIG. 3A

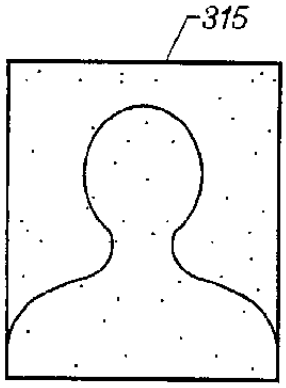


FIG. 3B

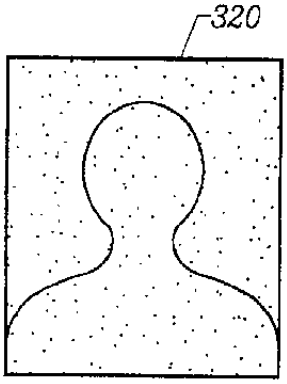


FIG. 3C

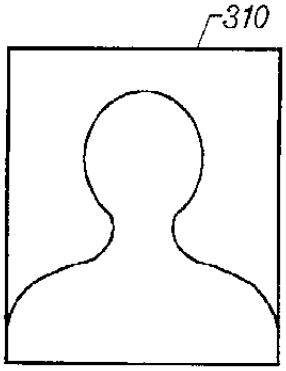


FIG. 3D

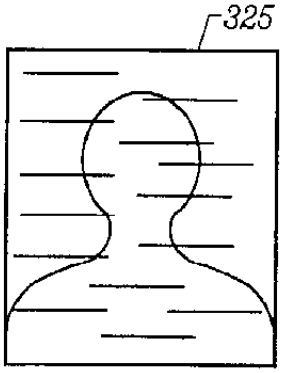


FIG. 3E

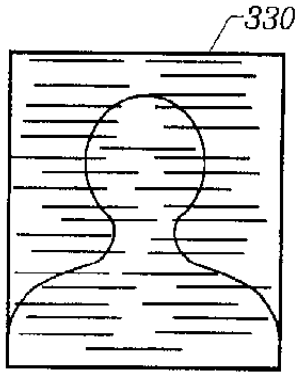


FIG. 3F

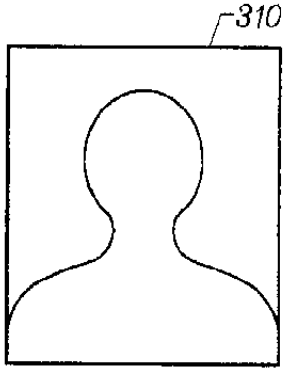


FIG. 3G

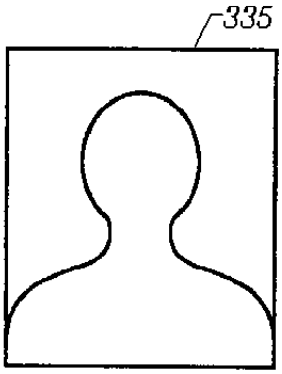


FIG. 3H

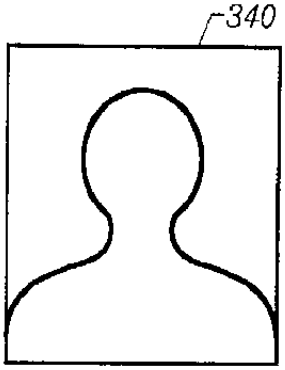


FIG. 3I

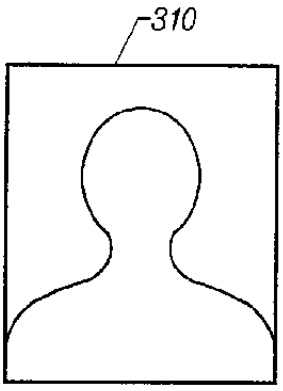


FIG. 3J

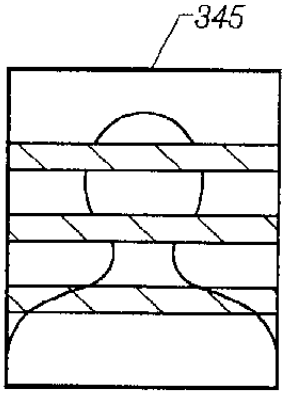


FIG. 3K

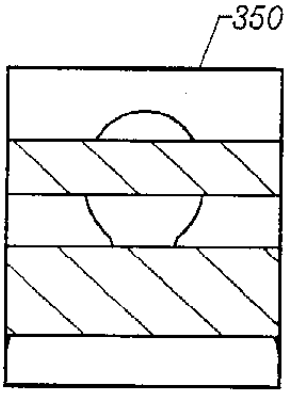


FIG. 3L

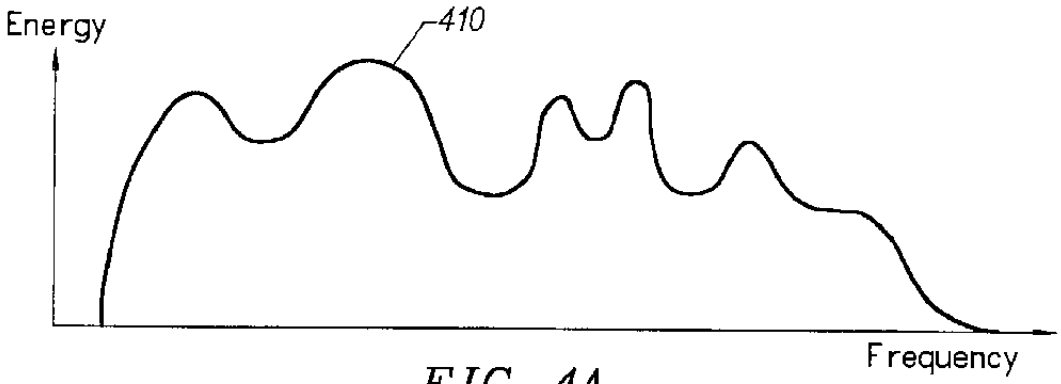


FIG. 4A

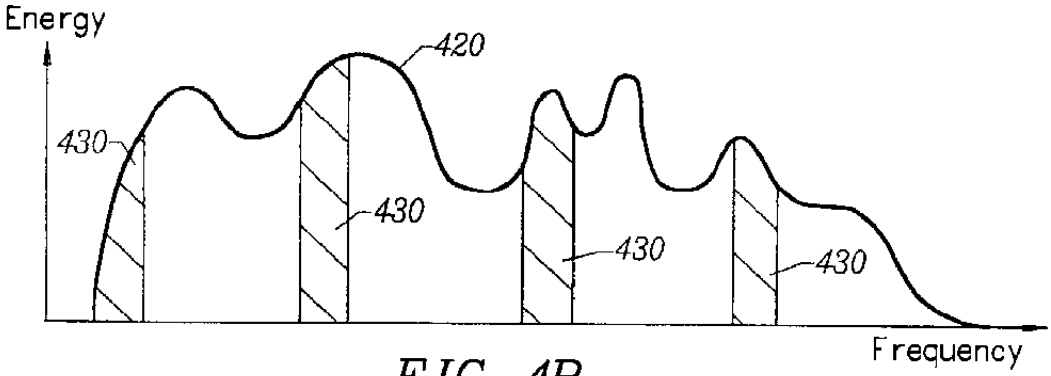


FIG. 4B

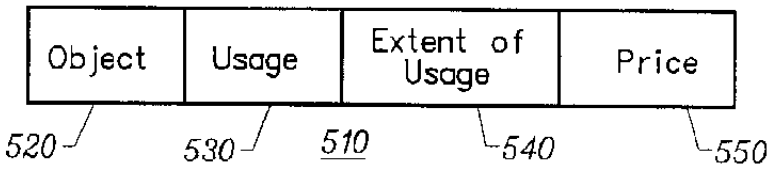


FIG. 5A

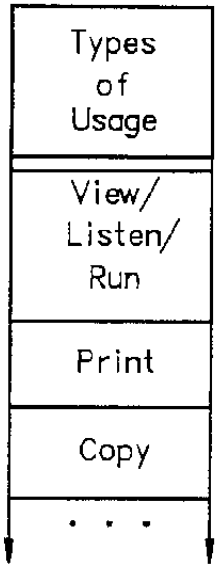


FIG. 5B

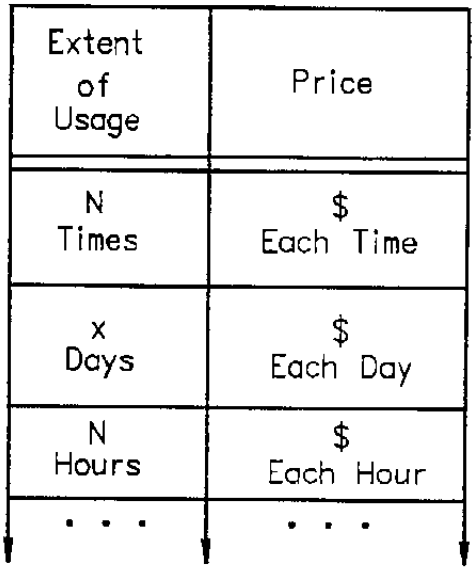


FIG. 5C

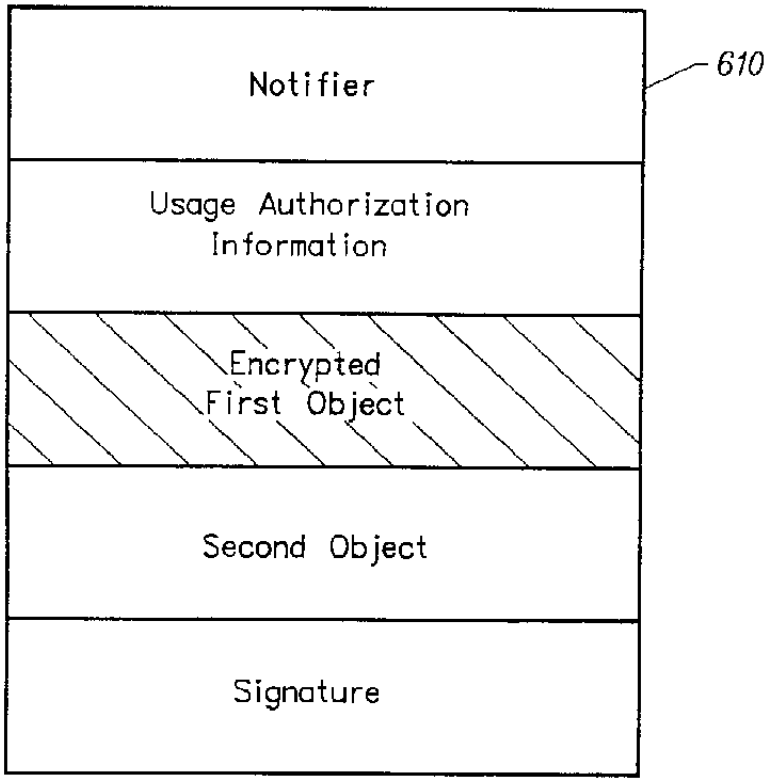

Notifier
610
Usage Authorization Information
Encrypted First Object
Second Object
Signature

FIG. 6

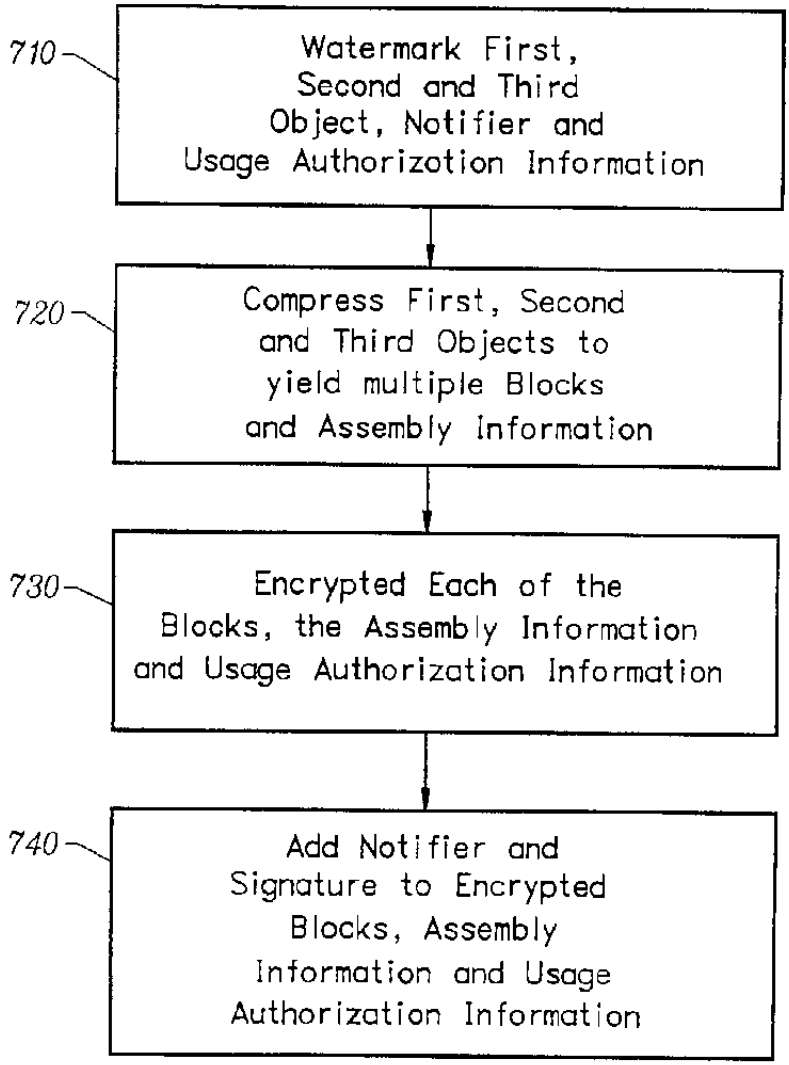

710
Watermark First, Second and Third Object, Notifier and Usage Authorization Information
720
Compress First, Second and Third Objects to yield multiple Blocks and Assembly Information
730
Encrypted Each of the Blocks, the Assembly Information and Usage Authorization Information
740
Add Notifier and Signature to Encrypted Blocks, Assembly Information and Usage Authorization Information

FIG. 7

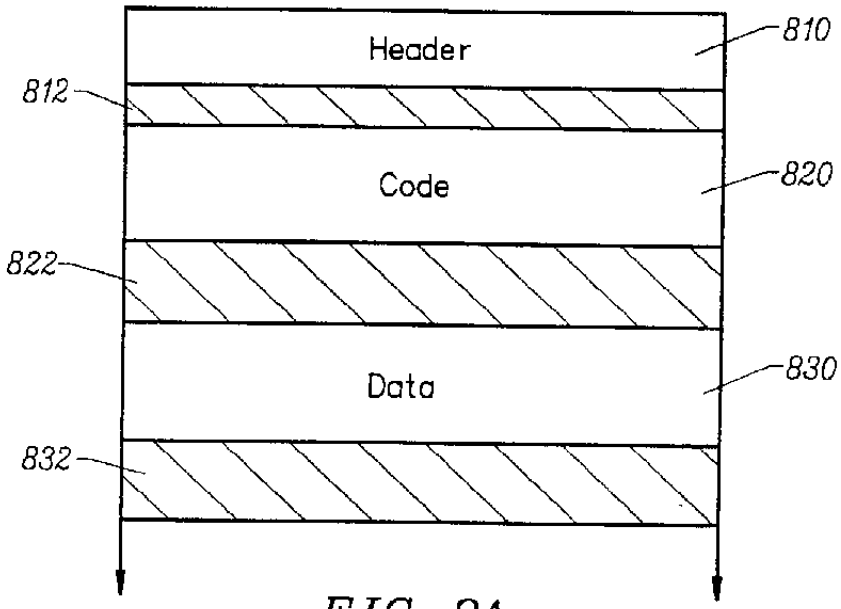

Header
Code
Data
810
812
820
822
830
832

FIG. 8A

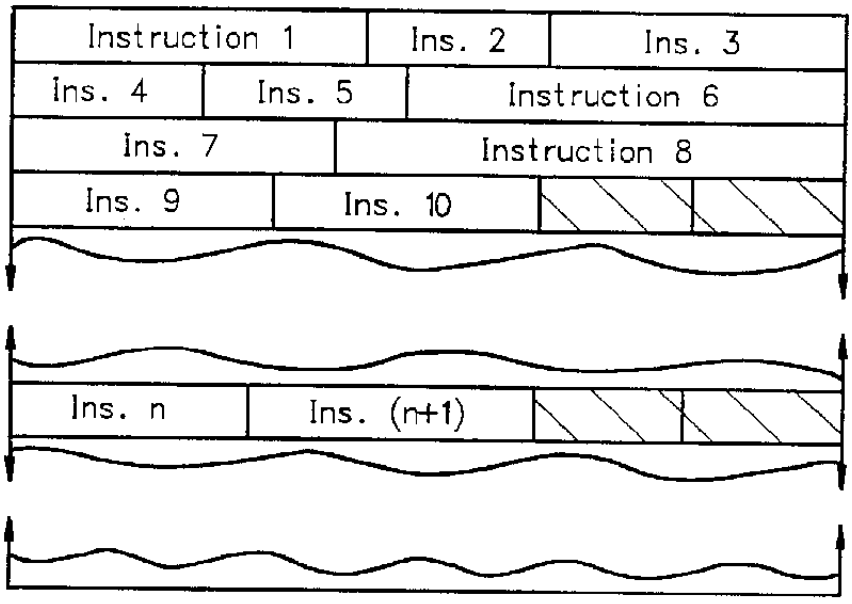

Instruction 1
Ins. 2
Ins. 3
Ins. 4
Ins. 5
Instruction 6
Ins. 7
Instruction 8
Ins. 9
Ins. 10
Ins. n
Ins. (n+1)

FIG. 8B

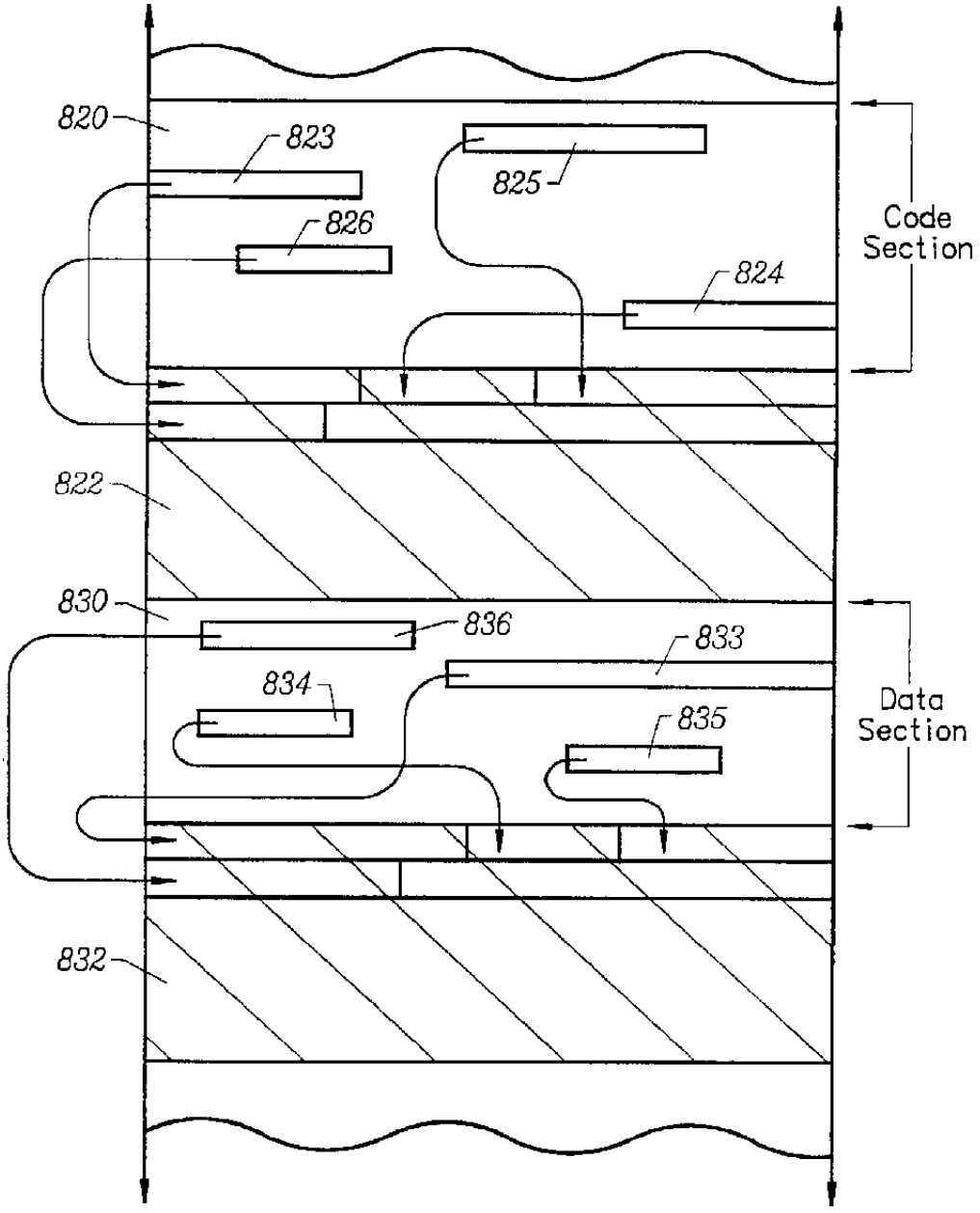

820
823
825
826
824
Code Section
822
830
836
833
834
835
Data Section
832

FIG. 8C

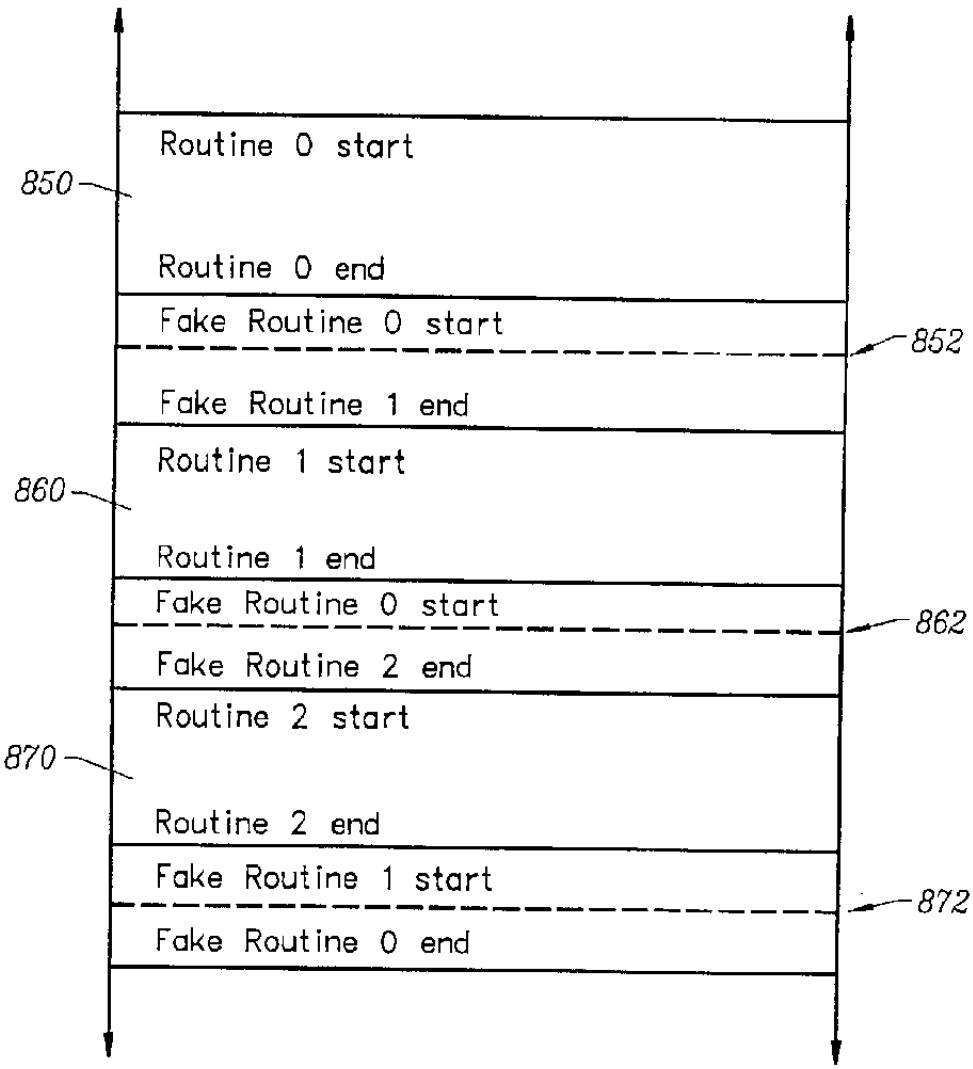

Routine 0 start
850
Routine 0 end
Fake Routine 0 start
852
Fake Routine 1 end
Routine 1 start
860
Routine 1 end
Fake Routine 0 start
862
Fake Routine 2 end
Routine 2 start
870
Routine 2 end
Fake Routine 1 start
872
Fake Routine 0 end

FIG. 8D

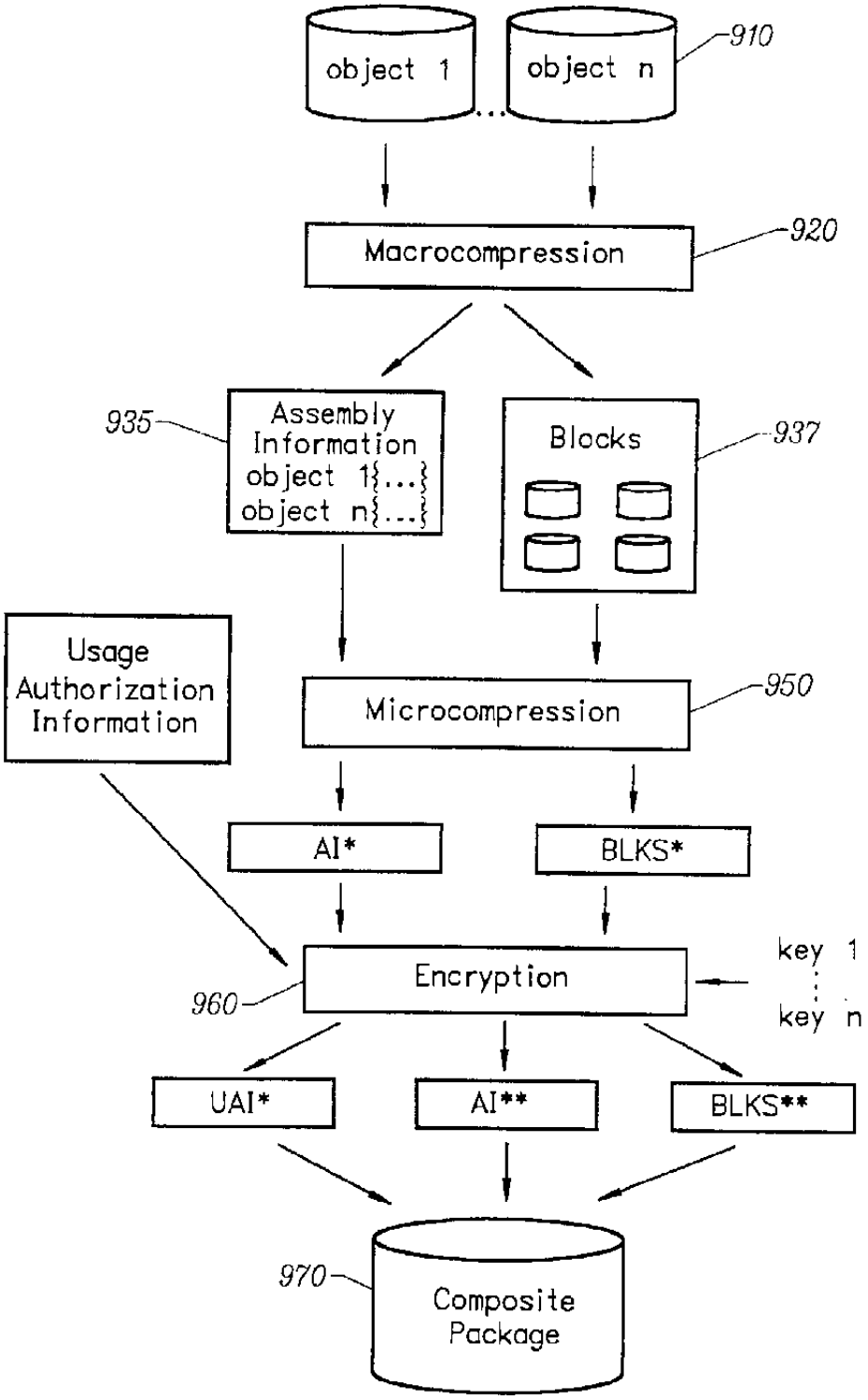

object 1
object n
910
Macrocompression
920
935
Assembly Information
object 1{...}
object n{...}
Blocks
937
Usage Authorization Information
Microcompression
950
AI*
BLKS*
Encryption
960
key 1
key n
UAI*
AI**
BLKS**
970
Composite Package

FIG. 9A

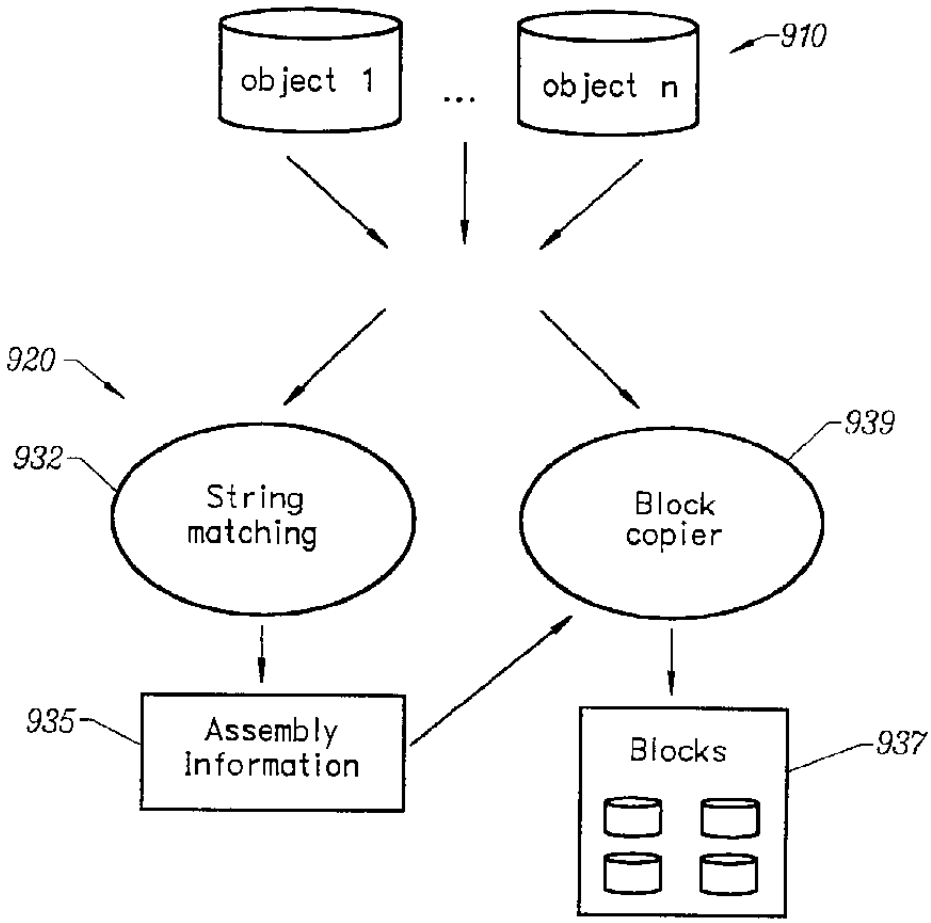

object 1
...
object n
910
920
932
String matching
939
Block copier
935
Assembly Information
Blocks
937

FIG. 9B

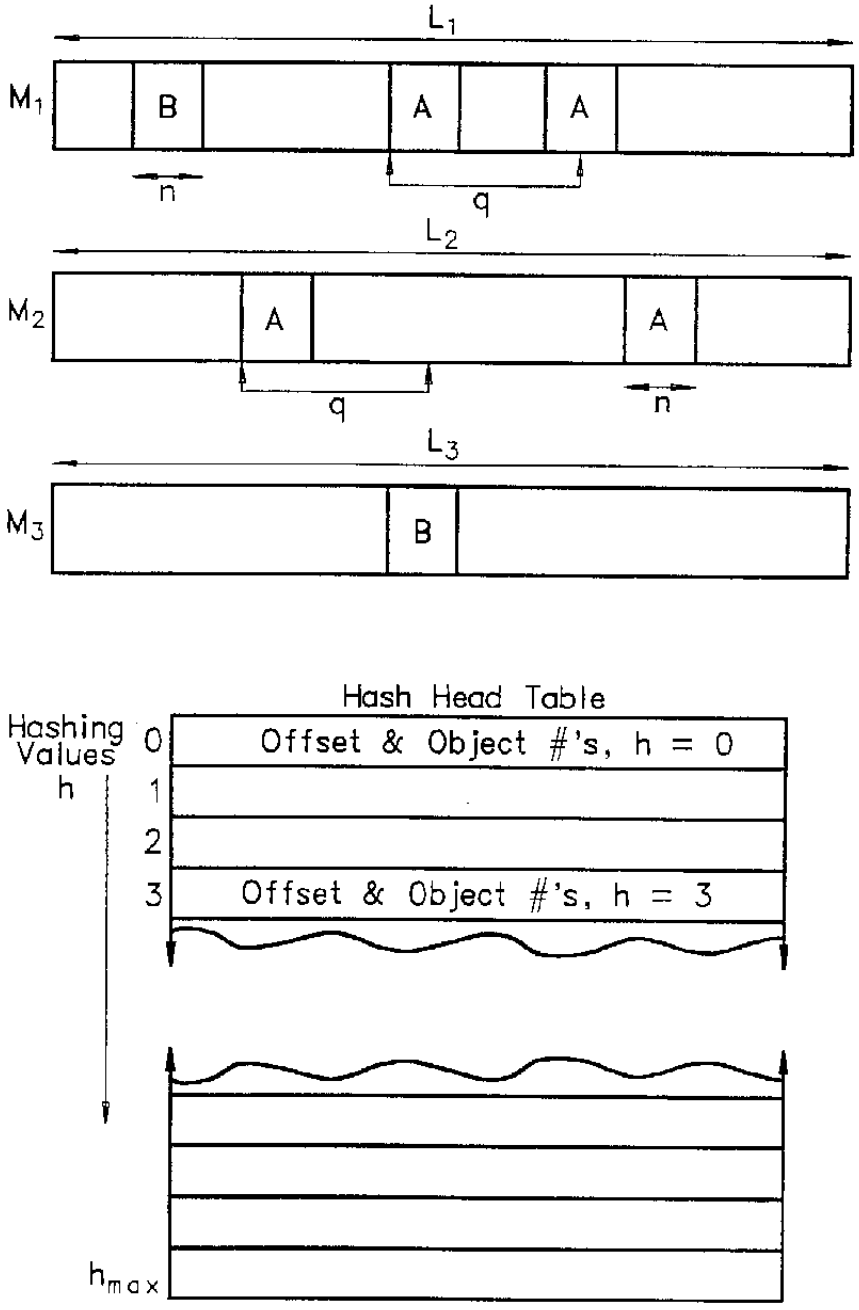

L1
M1
B
A
A
n
q
L2
M2
A
A
q
n
L3
M3
B
Hash Head Table
Hashing Values
h
0
Offset & Object #'s, h = 0
1
2
3
Offset & Object #'s, h = 3
hmax

FIG. 9C

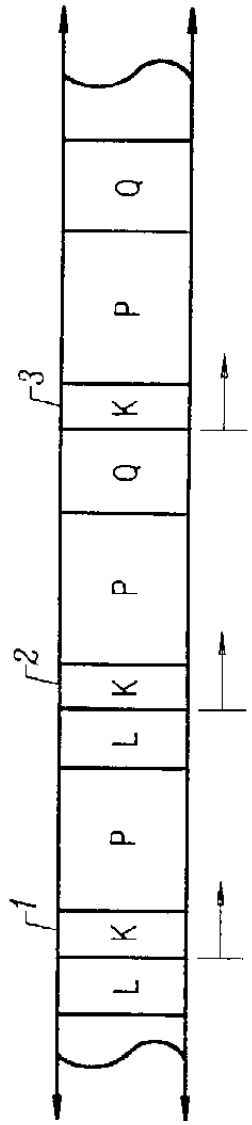


FIG. 9D

String ID	Source Offset	Length
1(Offset)	2(Offset)	K+P
2(Offset)	2(Offset)	K+P+Q
3(Offset)	2(Offset)	K+P+Q

FIG. 9F

Strings	Matching Length
1, 2	K+P
1, 3	K+P
2, 3	K+P+Q

FIG. 9E

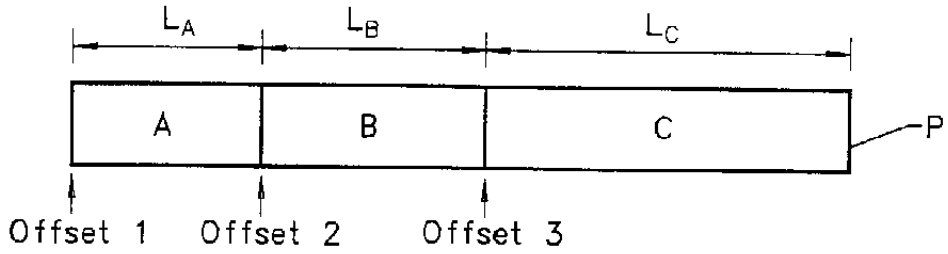


FIG. 9G

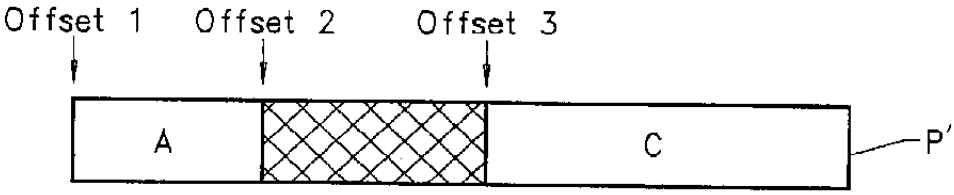


FIG. 9G

Segment	Source	Length
Offset 1	Off 1, P	L_A
Offset 2	0's	L_B
Offset 3	Off 3, P	L_C

T

FIG. 9H

Segment	Source	Length
Offset 1	Offset 1, P	L_A
Offset 2	Source Fct	L_B
Offset 3	Offset 3, P	L_C

FIG. 9I

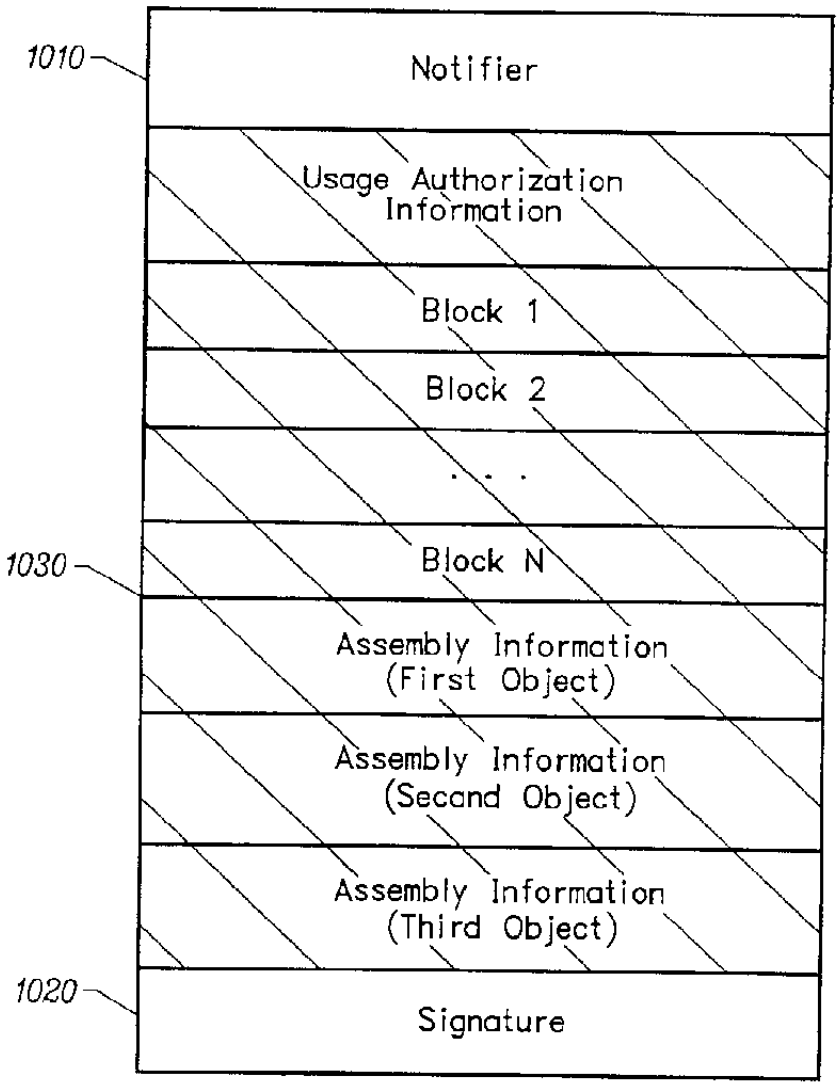

1010
Notifier
Usage Authorization Information
Block 1
Block 2
. . .
Block N
1030
Assembly Information (First Object)
Assembly Information (Second Object)
Assembly Information (Third Object)
1020
Signature

FIG. 10

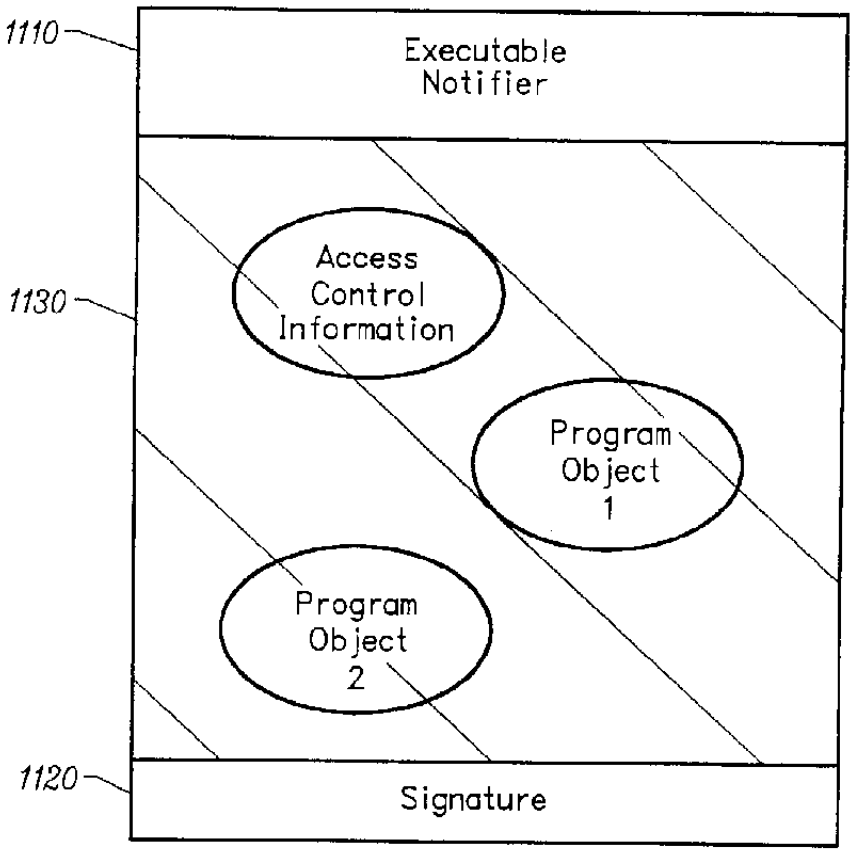

1110
Executable
Notifier
1130
Access
Control
Information
Program
Object
1
Program
Object
2
1120
Signature

FIG. 11A

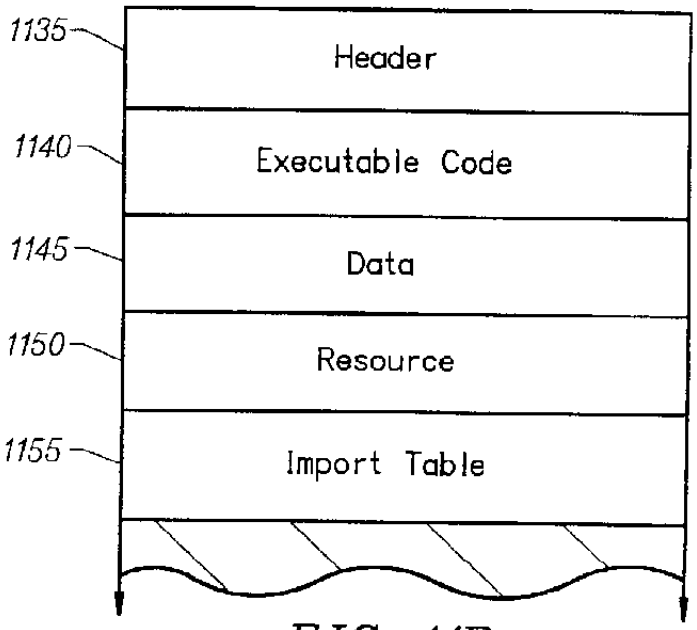


FIG. 11B

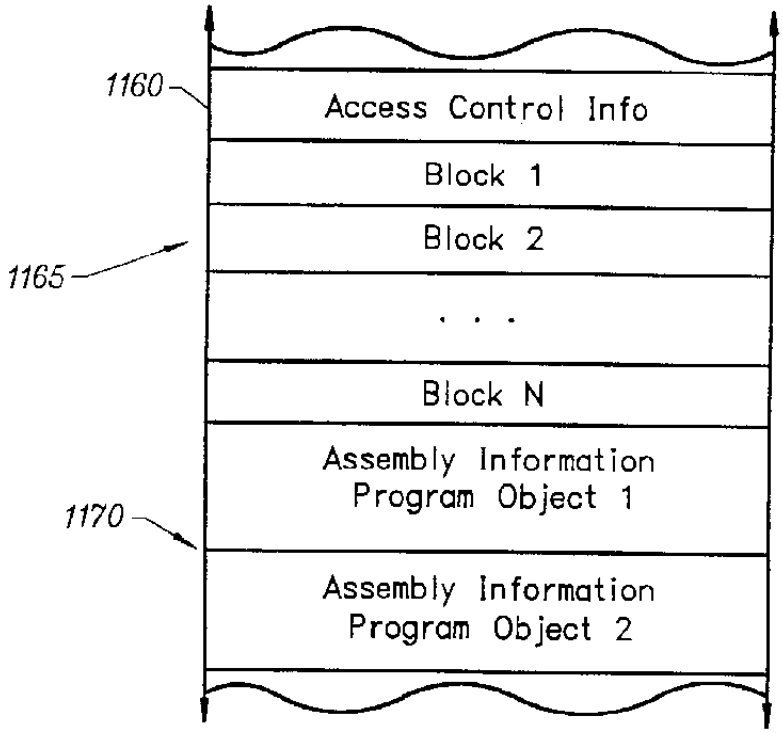


FIG. 11C

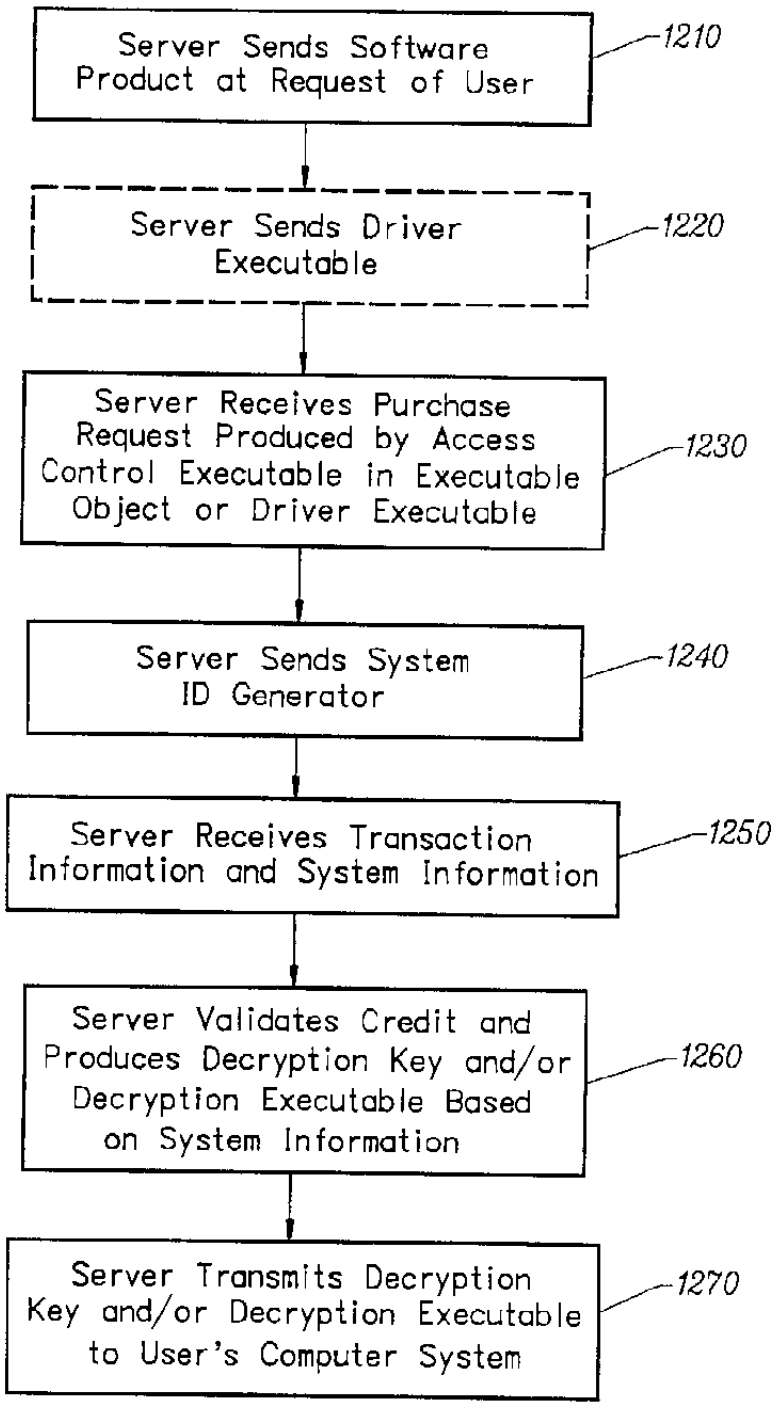

Server Sends Software Product at Request of User
1210
Server Sends Driver Executable
1220
Server Receives Purchase Request Produced by Access Control Executable in Executable Object or Driver Executable
1230
Server Sends System ID Generator
1240
Server Receives Transaction Information and System Information
1250
Server Validates Credit and Produces Decryption Key and/or Decryption Executable Based on System Information
1260
Server Transmits Decryption Key and/or Decryption Executable to User's Computer System
1270

FIG. 12

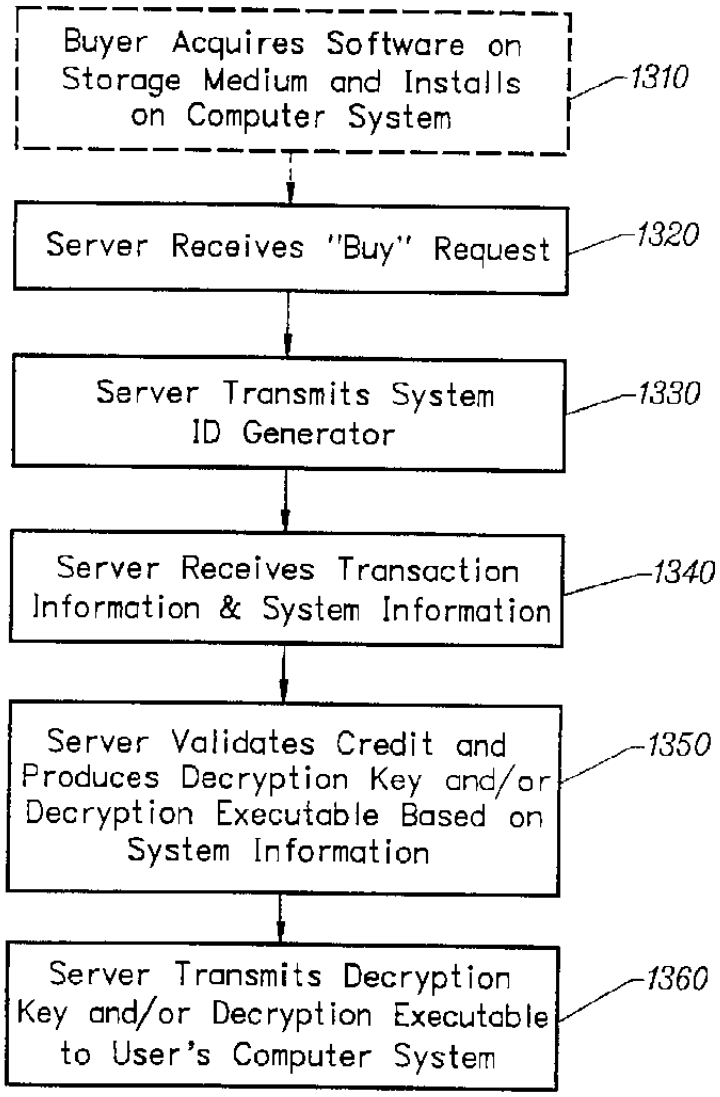

Buyer Acquires Software on Storage Medium and Installs on Computer System
1310
Server Receives "Buy" Request
1320
Server Transmits System ID Generator
1330
Server Receives Transaction Information & System Information
1340
Server Validates Credit and Produces Decryption Key and/or Decryption Executable Based on System Information
1350
Server Transmits Decryption Key and/or Decryption Executable to User's Computer System
1360

FIG. 13

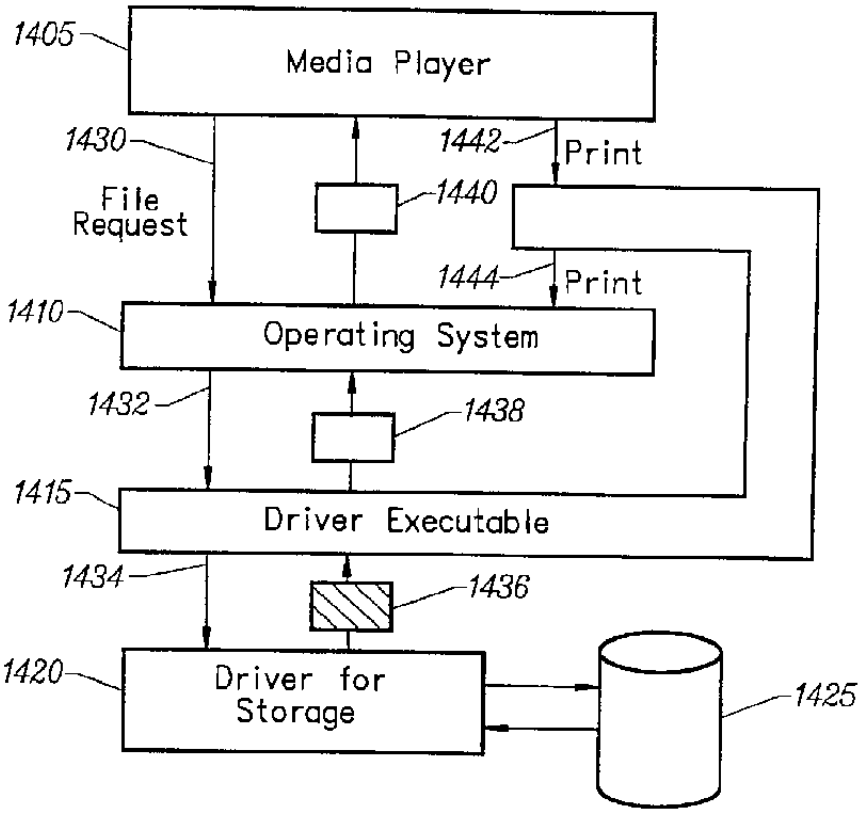

1405
Media Player
1430
File Request
1442
Print
1440
1444
Print
1410
Operating System
1432
1438
1415
Driver Executable
1434
1436
1420
Driver for Storage
1425

FIG. 14

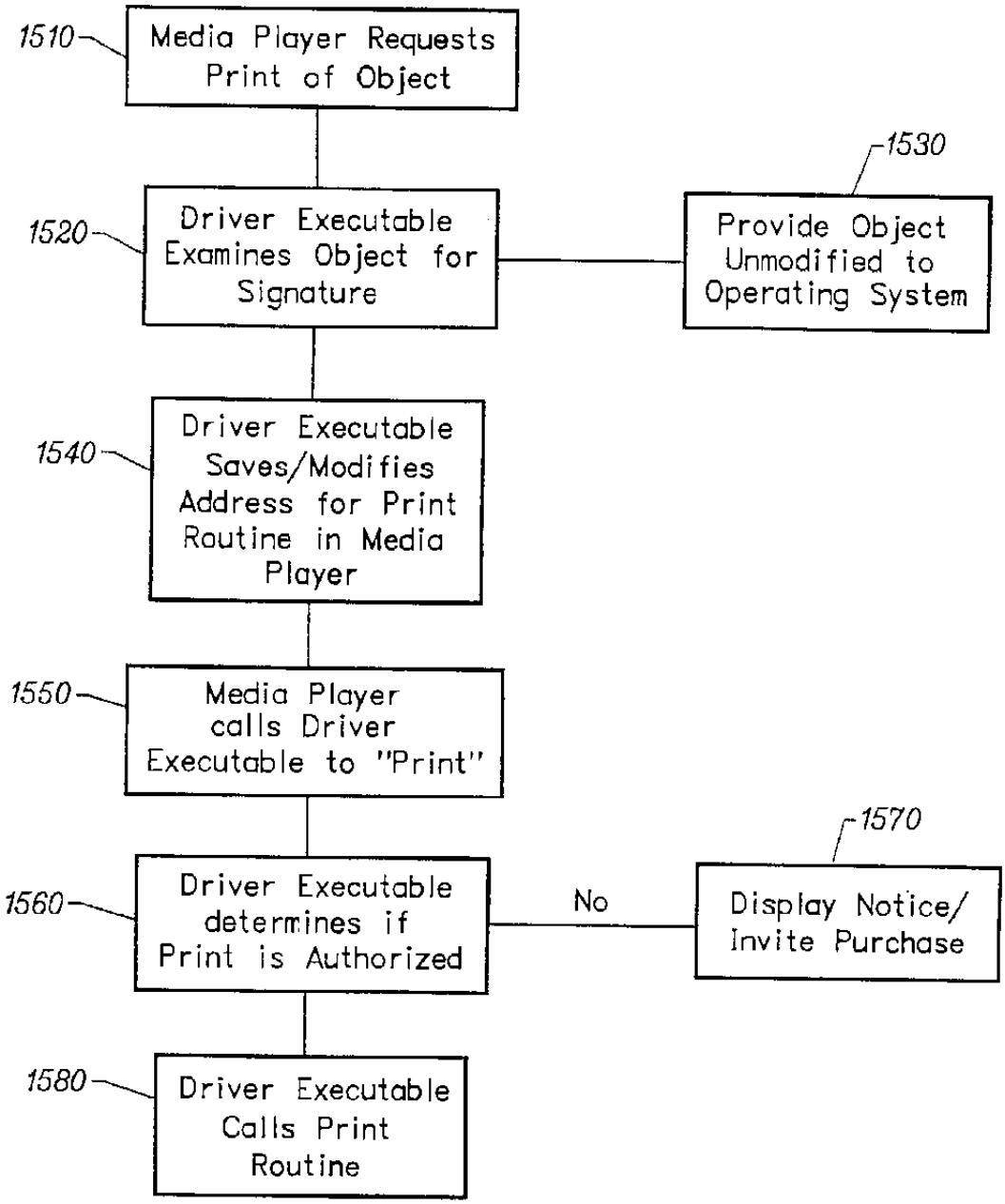

1510
Media Player Requests Print of Object
1520
Driver Executable Examines Object for Signature
1530
Provide Object Unmodified to Operating System
1540
Driver Executable Saves/Modifies Address for Print Routine in Media Player
1550
Media Player calls Driver Executable to "Print"
1560
Driver Executable determines if Print is Authorized
No
1570
Display Notice/ Invite Purchase
1580
Driver Executable Calls Print Routine

FIG. 15

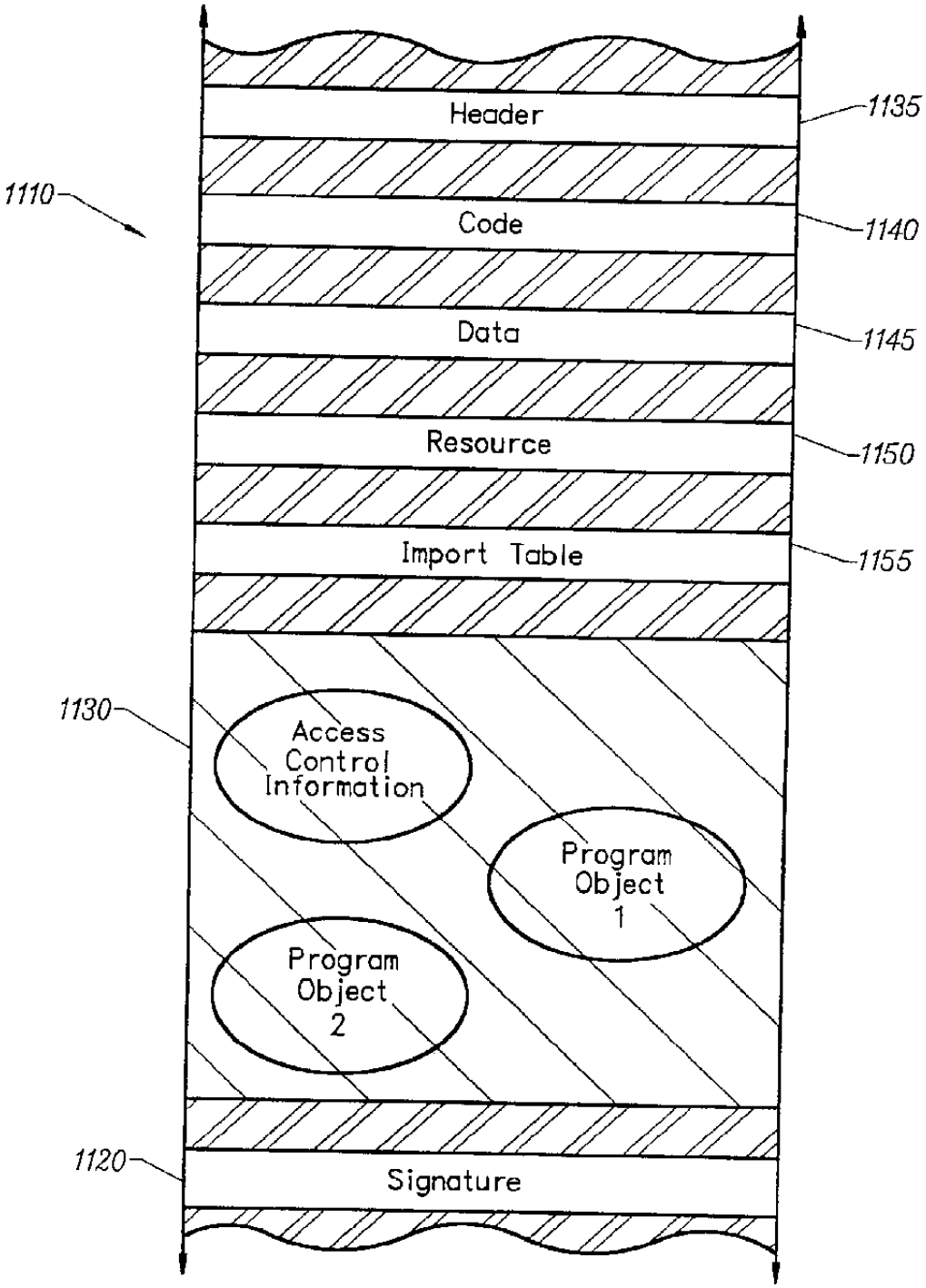

Header
1135
1110
Code
1140
Data
1145
Resource
1150
Import Table
1155
1130
Access Control Information
Program Object 1
Program Object 2
1120
Signature

FIG. 16

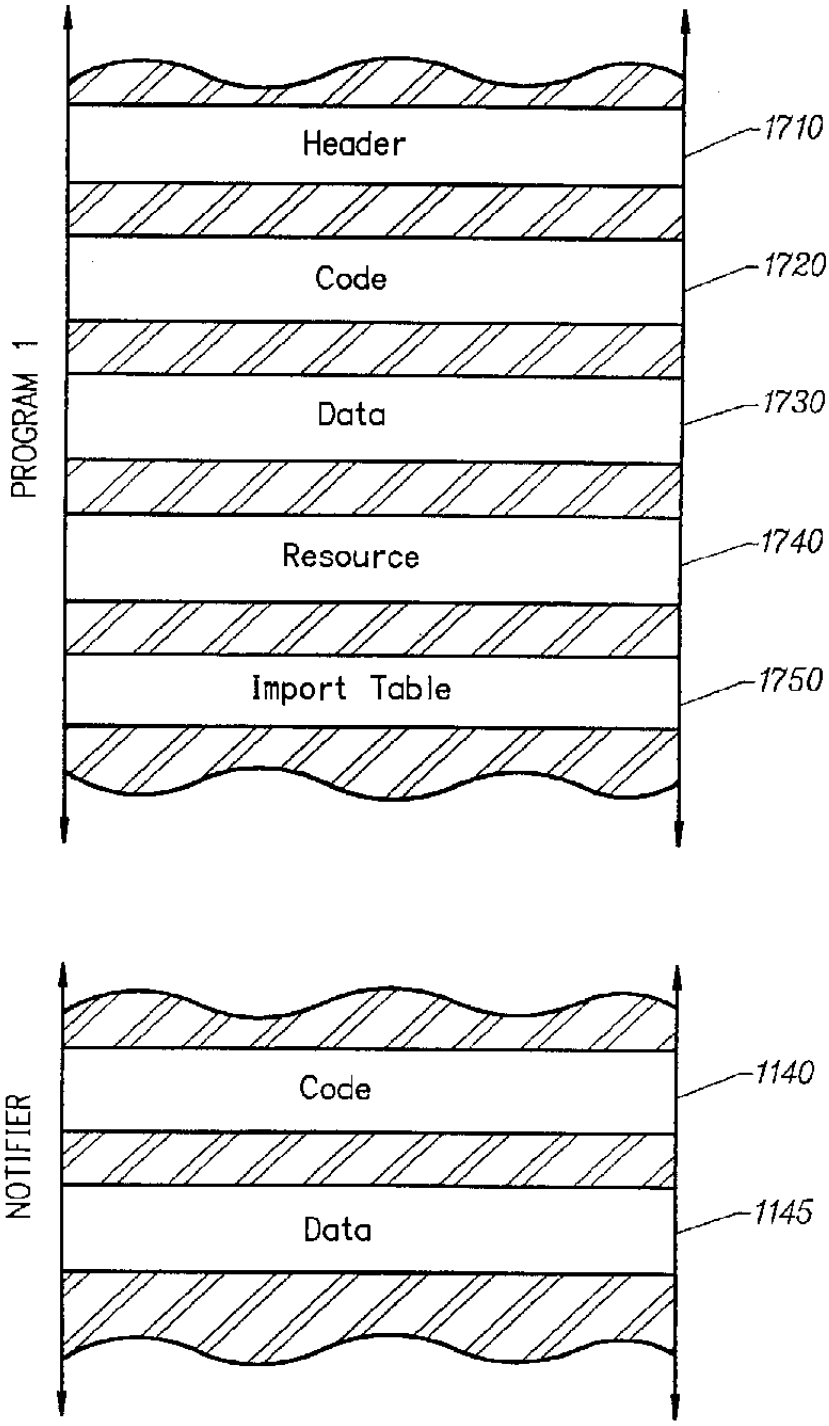

PROGRAM 1
Header
Code
Data
Resource
Import Table
1710
1720
1730
1740
1750
NOTIFIER
Code
Data
1140
1145

FIG. 17

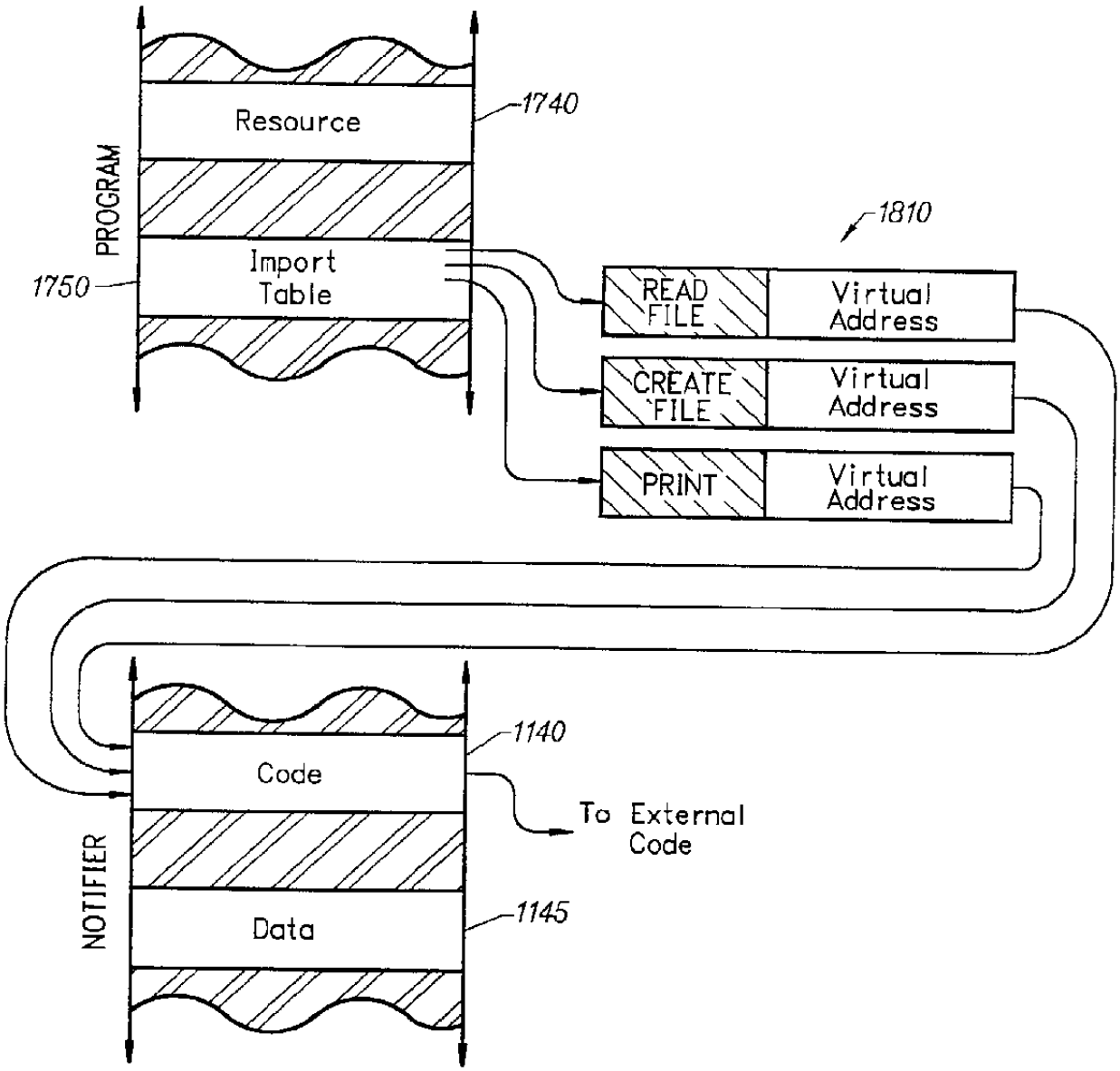

PROGRAM
Resource
1740
Import Table
1750
1810
READ FILE
Virtual Address
CREATE FILE
Virtual Address
PRINT
Virtual Address
NOTIFIER
Code
1140
To External Code
Data
1145

FIG. 18

INTERNATIONAL SEARCH REPORT

Int. tional Application No
PCT/US 00/11545

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 758 068 A (BRAND ET AL.) 26 May 1998 (1998-05-26)	1,2,7,8, 13-15, 17, 22-28, 31-33, 40-45
A	column 2, line 51 -column 8, line 9; figures	46-84
X	MARY TORK ROTH, PETER SCHWARTZ: "A Wrapper architecture for legacy data sources" IBM ALMADEN RESEARCH CENTER, 'Online! 1997, pages 1-21, XP002145099 Retrieved from the Internet: <URL:http://www.almaden.ibm.com/cs/garlic/ vldb97wraprj.ps> 'retrieved on 2000-08-16!	55-65
A	the whole document	1-54

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 August 2000	07/09/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Soler, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. ..tional Application No
PCT/US 00/11545

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5758068 A	26-05-1998	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :		(11) International Publication Number:	WO 92/20022
G06F 1/00	A1	(43) International Publication Date:	12 November 1992 (12.11.92)

(21) International Application Number: PCT/US92/03812

(22) International Filing Date: 6 May 1992 (06.05.92)

(30) Priority data:

697,652	8 May 1991 (08.05.91)	US
723,456	28 June 1991 (28.06.91)	US
722,840	28 June 1991 (28.06.91)	US
723,457	28 June 1991 (28.06.91)	US

(71) Applicant: DIGITAL EQUIPMENT CORPORATION [US/US]; 146 Main Street, Maynard, MA 01754 (US).

(72) Inventor: WYMAN, Robert, Mark; 410 Second Avenue, South No. 108, Kirkland, WA 98033 (US).

(74) Agents: NATH, Ram, B. et al.; c/o Joyce D. Lange, Digital Equipment Corporation. 111 Powdermill Road, Maynard, MA 10754 (US).

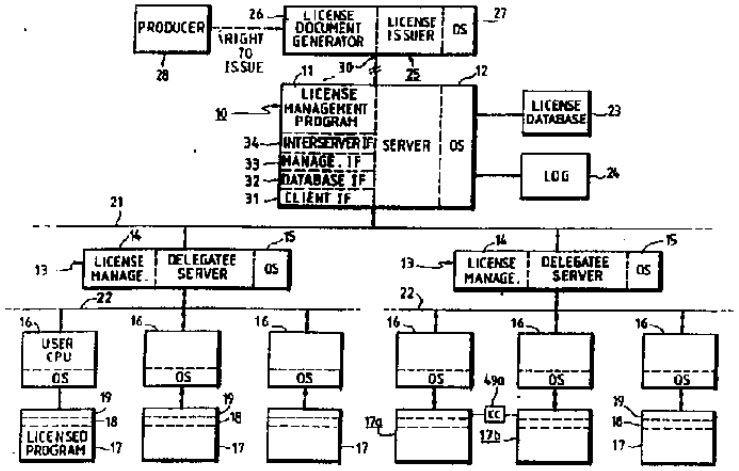
(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BF (OAPI patent), BG, BJ (OAPI patent), BR, CA, CF (OAPI patent), CG (OAPI patent), CH, CH (European patent), CI (OAPI patent), CM (OAPI patent), CS, DE, DE (European patent), DK, DK (European patent), ES, ES (European patent), FI, FR (European patent), GA (OAPI patent), GB, GB (European patent), GN (OAPI patent), GR (European patent), HU, IT (European patent), JP, KP, KR, LK, LU, LU (European patent), MC (European patent), MG, ML (OAPI patent), MR (OAPI patent), MW, NL, NL (European patent), NO, PL, RO, RU, SD, SE, SE (European patent), SN (OAPI patent), TD (OAPI patent), TG (OAPI patent).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: MANAGEMENT INTERFACE AND FORMAT FOR LICENSE MANAGEMENT SYSTEM



(57) Abstract

A distributed computer system employs a license management system to account for software product usage. A management policy having a variety of alternative styles and contexts is provided. Each licensed product upon start-up makes a call to a license server to check on whether usage is permitted, and the license server checks a database of the licenses, called product use authorizations, that it administers. If the particular use requested is permitted, a grant is returned to the requesting user node. The product use authorization is structured to define a license management policy allowing a variety of license alternatives by values called "style", "context", "duration" and "usage requirements determination method". The license administration may be delegated by the license server to a subsection of the organization, by creating another license management facility duplicating the main facility. The license server must receive a license document (a product use authorization) from an issuer of licenses, where a license document generator is provided. A mechanism is provided for one user node to make a call to use a software product located on another user node; this is referred to as a "calling card", by which a user node obtains permission to make a procedure call to use a program on another node. A management interface allows a license manager at a server to modify the license documents in the database maintained by the server, within the restraints imposed by the license, to make delegations, assignments, etc. The license documents are maintained in a standard format referred to as a license document interchange format so the management system is portable and can be used by all adhering software vendors. A feature of the database management is the use of a filter function.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FI	Finland	ML	Mali
AU	Australia	FR	France	MN	Mongolia
BB	Barbados	GA	Gabon	MR	Mauritania
BE	Belgium	GB	United Kingdom	MW	Malawi
BF	Burkina Faso	GN	Guinea	NL	Netherlands
BG	Bulgaria	GR	Greece	NO	Norway
BJ	Benin	HU	Hungary	PL	Poland
BR	Brazil	IE	Ireland	RO	Romania
CA	Canada	IT	Italy	RU	Russian Federation
CF	Central African Republic	JP	Japan	SD	Sudan
CG	Congo	KP	Democratic People's Republic of Korea	SE	Sweden
CH	Switzerland			SN	Senegal
CI	Côte d'Ivoire	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
CS	Czechoslovakia	LK	Sri Lanka	TG	Togo
DE	Germany	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		
ES	Spain	MG	Madagascar		

MANAGEMENT INTERFACE AND FORMAT FOR LICENSE MANAGEMENT SYSTEM

BACKGROUND OF THE INVENTION

This invention relates to methods of operation of computer systems, and more particularly to a method and system for managing the licensing of software executed on computer systems.

In U.S. Patent 4,937,863, issued to Robert, Chase and Schafer and assigned to Digital Equipment Corporation, the assignee of this invention, a Software Licensing Management System is disclosed in which usage of licensed software may be monitored in a computer system to determine if a use is within the scope of a license. The system maintains a database of licenses for software products,

delivering the license document may be in the form of a network, or may be a phone line using modems, or may include physical delivery by disks or CD ROMs, for example. Likewise, the method of delivery of the software products being licensed, i.e., the applications programs 17 to be executed on the CPUs 16, is not material to the license management facility of the invention; the products are delivered by some appropriate means, e.g., the communications link 30 and the networks 21 and 22, by CD ROMs or disks physically distributed, etc.

Although shown in Figure 1 as operating on a distributed system, in the simplest case the license management facility of the invention may be operated on a single CPU. The license management program 11 and the applications program 17 may be executing on the same CPU, in which case the license document would be stored in a database 23 as before, on this CPU, and the calls from the unit 18 to the license server would be local instead of RPCs. As in the distributed system, however, the licensed product would still not have access to the license document, but instead could only make inquires to the server program, even if all are executing on the same CPU.

In operation of the distributed system of Figure 1, the producer 28 gives the issuer 25 authority to grant licenses on its behalf (the producer and issuer can be a single entity or multiple entities). The license document generator program 26, under control of a user (a person), generates a license (usually the result of negotiation between the user of program 26 and a user of the server 10). This license is called a product use authorization, and it is transmitted by the link 30 to the server 10. The license management program in the server 10 stores the product use authorization in the database 23, and, if delegation is an authorized option, may distribute parts of the authorized use to the delegatee servers 13.

where it is likewise stored in a database. Thereafter, administration of the license is only in response to inquiries from user nodes 16. When execution of a program 17 begins, the unit 18 is invoked to check on the availability of a license for this particular node. The unit 18 sends (as by an RPC) a request to the license management program 14 (or 11 if there is no delegatee), where the product use authorization stored in database 23 is checked to see if use is authorized. If so, a return is sent to the user node 16, granting permission to continue. When the program 17 has finished executing, the unit 18 again is invoked to signal to the license management program, again by an RPC, that the authorization is released, so the license management program can take appropriate action, e.g., log the use in log 24, etc.

To implement these operations, the license management program 11 or 14 contains several functions, including a client interface 31, a database interface 32, a management interface 33, and an interserver interface 34 for communicating with the delegatees 13 (if any). The client interface 31, as described below, handles the requests received from the user nodes 16, and returns resulting from these requests. The database interface 32 handles the storing and retrieval of license information in the database 23, and logging license usage activity to log 24, and retrieval of this data. The management interface 33 handles the tasks of receiving the product use authorizations from the issuer 25 and maintaining the database 23 via the database interface 32. The interserver interface 34 handles the task of communicating with the delegatee servers 13, including transmitting the assigned parts of the product use authorizations, or communicating with other license servers that may be separately executing the license management function; for example, calls for validating calling cards may be made to another such server.

If there are no delegatees or no other license servers, then of course the interserver interface 34 has no function, and is idle.

The license document or "product use authorization" forming the basis for the license management activity of the program 11 on the server 10 may be illustrated as a data structure containing the information set forth in Figure 2; in actual practice the product use authorization is preferably a more abstract data arrangement, not in such a rigidly structured format as illustrated. For example, the product use authorization as well as similar documents stored in the database 23, or passed between components of the system of Figure 1, may be of the so-called tag-length-value data format, where the data structure begins with an identifying tag (e.g., PUA or product use authorization) followed by a field giving the length, followed by the value itself (the content). One type of data treatment using this tag-length-value format is an international standard referred to as ASN.1 or Abstract Syntax Notation. In any event, the document 35 illustrated in Figure 2 is merely for discussing the various items of data, rather than representing the way the information is stored. Some of the fields shown here exist at some times and not others, and some are optional; the product use authorization may also include additional fields not shown or discussed here. Also it should be noted that copies of parts of this type of document are made for the delegatees, so this representation of Figure 2 is a composite of several documents used in the system of Figure 1. The document 35 includes fields 36 identifying the software product by product name, producer, version numbers, release date, etc. The issuer 25 is identified in field 37, and the licensee (usually the owner of the license server 10) identified in field 38. The essential terms of the license grant are then defined in fields 40-46. The start date and end date are specified in fields 40; these store the exact time (date, hour, minute, second, etc.) when the license becomes valid and

when it ends, so licenses may be granted to start at some future time and to end at a particular time. Note that the previous practice has been to specify only the ending date, rather than also a start date as employed here. Each of the nodes, including issuer 25, servers 10 and 13, and user nodes 16, maintain a time value by a local clock referenced to a standard, so inherent in the license management facility is the maintaining of a time standard to compare with the start and end date information in the fields 40. The units granted are specified in field 41; the units are an arbitrary quantitative measure of program usage. In a delegatee server 13, the units field 41 will have some subset of the units field in the original product use authorization. As units are granted to users 16 or delegated, the remaining units available for grant are indicated in a subfield 42 in the copy of the document used by the server. The management policy occupies fields 43-46, and includes style, context, duration and LURDM (license use requirements determination method), as will be explained. The style field 43 specifies whether the licensed units are controlled by an "allocative" style or "consumptive" style, or some other "private" algorithm, where styles are ways used to account for the consumption or allocation of the units. The context field 44 specifies the location and environment in which product use or license management occurs, i.e., a CPU or an individual user or a network, etc. Duration field 45 indicates whether the license granted to a user is by assignment, by transaction, or immediate. The LURDM field 46 indicates the license use requirements determination method, in some cases using a license use requirements table (LURT) seen as field 47, as will be described.

Additional fields 48-54 in the product use authorization 35 of Figure 2 define features such as delegation authorization, calling authorization, overdraft

authorization, combination authorization, token, signature, checksum, etc. These will be described in the following paragraphs.

If the delegation field 48 is true, a license server 10 may distribute license units to multiple servers 13. A time limit may be imposed, i.e., units can be delegated to other hardware systems until they time out. Delegation allows an administrator to distribute units to improve response time and increase the resilience of the system. For example, the communication network 21 may include a satellite link to a remote facility where the local server 13 has a number of clients or users 16, in which case the calls to the server 13 would be completed much quicker than would be the case if calls had to be made to the server 10. Also, delegation may be used as a method of allocating licensed units within a budget for administrative purposes. Usually the delegation authorization is a feature that is priced by the issuer, i.e., a license granting 1000 units with delegation authorization is priced higher than without this authorization.

The field 49 contains a calling authorization and/or a caller authorization. If the caller authorization in field 49 is true, the product is permitted to receive calls from other named products requesting use of the product, and if conditions are met (identified caller is authorized) the server can grant a calling card, as described below. If the calling authorization is true, the product can make calls to other products. If neither is true, then the product can neither make or receive calls using the calling card feature. Referring to Figure 1, if product 17a wishes to make a remote procedure call to a feature of product 17b running on a different user node 16, it makes a call to its server 13 including a request for a calling card, and, if permitted, the return to product 17a includes a calling card 49a. The product 17a then makes a call to product 17b in the usual manner of

RPCs, sending along the calling card 49a, which the product 17b then verifies by a call to its server 13 before executing the called procedure and issuing its return to product 17a. The feature of calling cards is important for distributed applications. For example, if a product is able to execute faster in a distributed system by assigning tasks to other CPUs, then the issue is presented of which license policy is needed, i.e., does every node executing a part of the task have to be licensed and consume or receive allocation of a unit, or just the one managing the task? This is resolved for most applications by use of this calling card concept. The product use authorization for such a product has the calling authorization field 49 enabled, so calling cards can be issued. This feature is typically separately priced.

The combination authorization field 50 of Figure 2 determines whether or not license requests from a user node 16 can be satisfied by combining units from multiple product use authorizations. It may be advantageous to purchase licenses with different policy values, and use units from certain product use authorizations only for overflow or the like. Or, for other reasons, it may be advantageous to "borrow" and "lend" units among delegated servers or user nodes. This function is permitted or denied by the content of field 50.

The overdraft field 51 determines whether or not a requested allocation from a user node 16 will be nevertheless granted, even though the units available field 42 is zero or too small to permit the requested use. Overdrafts can be unlimited, or a specific overdraft pool can be set up by a server 10, for a customer's internal administrative purposes. That is, the overdraft value may be unlimited in the original license, but limited or zero for internally distributed copies of the license. Thus, the product use authorization sent by the issuer 25 to

the customer may have overdrafts permitted by the field 51, but the customer may deny overdraft permission for its own budgeting purposes. In any event, if overdraft is permitted, additional fees have to be paid to the issuer at some accounting period, when the logged usage from log 24 indicates the available units have been exceeded. If overdraft is denied, then the units 18 of the user nodes making request allocations are structured to inform the products 17 that a license grant is not available. The intent is not to prevent the application program from running; the license server merely informs the application whether or not the license manager determines that it is authorized to run. The application can itself be structured to shut itself down if not authorized to run, or it can be structured to shut down certain functions (e.g., ability to save files, ability to print, etc.), or it can be structured to continue in a fully functional manner. The purpose of the license management facility is not that of enforcement, nor that of "copy protection", but instead is merely that of license management.

An optional token field 52 is available in the product use authorization 35 of Figure 2. This field can contain comments or other information desired by the issuer or user. For example, a telephone support number may be included in the token field, then when the product 17 shows its "help screen" the number is inserted. This number would be part of the argument, i.e., data transmitted to the user node 16, when the server 10 makes a return following a request allocation message from the user. This field may also be used to store information used in a "private" style, where the information from this field returned to the user node is employed by the application program 17 or the stub 19 to determine if the application can be activated.

The signature field 53 in the product use authorization 35 is a part of a validation mechanism which provides important features. This field contains a digital signature encoded to reflect the data in the license itself, as well as other encoding methods not known to customers, so it cannot be duplicated unless the encoding algorithm is known. In a preferred embodiment, a so-called "public/private key" system of encoding is used for the signature field 53. The encoding algorithm used to generate the signature 53 is known to the issuer 25, using a private key, and anyone knowing the public key can decode the signature to determine if it is valid but cannot determine the encoding algorithm so it cannot produce a forged signature. So, if the server 10 knows the public key which is unique to the issuer 25, it can determine if a license document 35 is genuine, but it cannot itself generate license documents. However, if the server possesses a valid license document that gives it the right to delegate, then it will be assigned its own private key (different from all other issuers or servers) and its delegatees 13 will be able to determine if a valid delegated license is delivered to them as they will be given the public key for the servers 13. The field 53 will thus contain both the original signature from the issuer 25 and the license server's signature when delivered to a delegatee 13. The decoding algorithm using a public key for any signatures is thus used by the license server 10 or delegatee 13 to make sure a product use authorization 35 is authentic before it is stored in the database 23. Related to the digital signature 53 is a checksum field 54, which merely encodes a value related by some known algorithm to the data in the product use authorization 35 itself. This field may be used merely to check for corruption of the data as it is stored, recalled, and transmitted within the system. That is, the checksum is used for data validation rather than security.

Two concepts central to the license management system implemented using the license document or product use authorization 35 of Figure 2 are the "license units", specified in field 41 or 42 and the "context", specified in field 44. License units are an abstract numerical measure of product use allowed by the license. When a product 17 (or a function or feature of a product) makes a license-checking request, the license management program 11 on server 10 computes how many license units are required to authorize this particular use of the product, and this is the license units requirement, in some cases using the LURDM field 46. A "context" is a set of tagged values which define the location and environment in which product use or license management occurs. Context values may be specified in field 44 of the product use authorization 35 of Figure 2 to restrict the environments in which the license may be managed and in which product use may occur. A context template may also be specified in the field 44 to indicate which parts of the complete context of product use (sub-contexts) are significant in differentiating product uses for the purposes of unit allocation; when this is specified, it allows separate product uses to share license units in a controlled way.

The two general types of policies specified in field 43 are allocative and consumptive. An allocative policy grants to the holder a specific number of license units (field 41) and specifies the policy which must be used to account for the allocation of these units. A software product 17 which is being managed by an allocative license will require verification that the appropriate number of license units have been allocated to it prior to performing services to the user. Typically, this allocation of units occurs either at the time of activation of the product 17 or at the time that product use is enabled on a particular platform (user CPU 16). The units typically remain allocated to the product 17 throughout the period that the product is running or is enabled to run. Upon termination of

processing or disabling, the allocated units are deallocated and made available for allocation to other instances of the software product 17 (other users 16 activating the product). In general, as long as any license units remain unallocated in field 42, the holder of the license is contractually authorized to increase his utilization of the licensed product. The usage does not deplete the license, however, as the units are returned to the units-available field 42 after a user is finished, and can be granted again to another user.

A consumptive unit based license, indicated in policy field 43, grants to the holder a specific number of initial license units (from field 42) and specifies the policy used to account for the consumption of those units. A software product 17 which is being managed by a consumptive license will cause an appropriate number of license units to be consumed to reflect the services provided by the product. Once consumed, units cannot be reused. Thus, the number of units available for future use declines upon every use of the licensed software product 17. This may also be referred to as a "metered" policy, being conceptually similar to measured consumption of electricity, water, etc. When the number of available units in field 42 reaches zero, the license may require that further use of the product is prohibited, or, the agreement may permit continued decrementing of the number of available units; the result is the accumulation of a negative number of available units in the field 42. It is anticipated that most consumptive unit based licenses will consider negative units to represent an obligation of the license holder to pay the license issuer 25. The transaction log 24 maintains an audit trail for providing a record of the units used in a consumptive license.

Referring to Figure 3, the major elements of the management policy are set forth in a table, where the possible entries for the fields 43, 44, 45 and 46 are

listed. For the style entry 43, the possibilities are allocative and consumptive as just described, plus a category called "private" which represents a style of management undefined at present but instead to be created especially for a given product, using its own unique algorithm. It is expected that most licenses may be administered using the named alternatives of Figure 3, but to allow for future expansion to include alternatives not presently envisioned, or to permit special circumstances for unique software, the "private" choices are included, which merely mean that the product 17 will generate its own conditions of use. It is important to note that, except for the "private" alternative, the license management is totally in control of the license management program 11 on the license server 10 (or delegatee 13), rather than at the product 17. All the product 17 does, via the unit 18, is to make the request inquiry to the server 10 via the client interface 31, and report when finished.

The context field 44 specifies those components (sub-contexts) of the execution-context name which should be used in determining if unit allocations are required. License data is always used or allocated within, or for the benefit of, some named licensing context, and context can include "platform contexts" and "application contexts". Platform contexts are such things as a specific network, an execution domain, a login domain, a node, a process ID or a process family, a user name, a product name, an operating system, a specific hardware platform, as listed in Figure 3. Applications contexts are information supplied from the application (the product 17), such as may be used in a "private" method of determining license availability. The context name can use several of these, in which case the context name is constructed by concatenating the values of all subcontexts into a single context name, e.g., a VAX 3100 platform using VMS operating system.

The duration field 45 defines the duration of an allocation of license units to a specific context or the duration of the period which defines a valid consumptive use. For durations of type "Assignment," the specification of a reassignment constraint is also provided for, as discussed below. There are three types of duration, these being "transaction," "assignment" and "immediate" as seen in Figure 3.

The transaction duration type, when specified for an allocative policy, indicates that license units should be allocated to the specified context upon receipt of a license request and that those units should be deallocated and returned to the pool of available units upon receipt of a corresponding license release from a user node 16. Abnormal termination of the process or context having made the original license request will be semantically equivalent to a license release. On the other hand, when specified for a consumptive policy, this duration type indicates that license units should be allocated to the specified context upon receipt of a license request and permanently removed from the available units pool (field 42) upon receipt of a license release which reflects successful completion of the transaction. Upon receipt of a license release which carries an error status or upon abnormal termination of the processor context having made the original license request, the allocated units will be deallocated and returned to the pool of available units (field 42).

The assignment duration type in Figure 3 (field 45 of Figure 2) imposes the constraint that the required units must have been previously assigned to a specific context. The sub-contexts which must be specified in the assignment are those given in the context-template. A "reassignment constraint" may be imposed, and this is a limitation on how soon a reassignment can be made. For example, a

reassignment constraint of 30-days would require that units assigned to a specific context could not be reassigned more often than every 30-days; this would prevent skirting the intent of the license by merely reassigning units whenever a user of another context made a request allocation call for the product. Related to this assignment constraint, a "reallocation limit" may also be imposed, to state the minimum duration of an allocation; where there is a context template of process, the intent is to count the number of uses of the software product at a given time, but where software runs in batch rather than interactive mode it may run very quickly on a powerful machine, so a very few concurrent uses may permit almost unlimited usage - by imposing a reallocation constraint of some time period, this manner of skirting the intent of the license may be constrained.

The immediate duration type (field 45 of Figure 2) is used to indicate that the allocation or consumption of an appropriate number of license units from the pool of available units (field 42) should be performed immediately upon receipt of a license request. Receipt of license release or abnormal terminations will then have no impact on the license management system. When specified as the duration for an allocative policy, the effect will be simply to check if an appropriate number of license units are available at the time of a license request. When specified as the duration for a consumptive policy, the effect will be to deduct the appropriate number of license units from the available pool at the time of a license request, and, thereafter, abnormal termination, such as a fault at the user CPU 16 or failure of the network link, will not reinstate the units.

The LURDM or license unit requirement determination method, field 46, has the alternatives seen in Figure 3 and stores information used in calculating the number of units that should be allocated or consumed in response to a license

request. If this field specifies a table lookup kind, this means license unit requirements are to be determined by lookup in the LURT (field 47) which is associated with the current license. If a constant kind is specified, this indicates that the license units requirements are constant for all contexts on which the licensed product or product feature may run. A private LURDM specifies that the license unit requirements are to be determined by the licensed product 17, not by the license management facility 11. The license unit requirements tables (LURTs) provide a means by which issuers of licenses can store information describing the relation between context (or row selector) and unit requirements. The license units requirements determination method (LURDM) must specify "table lookup" for the LURT to be used, and if so a row selector must be specified, where a valid row selector is any subcontext, e.g., platform ID, user name, time of day, etc. An example of an LURT fragment is shown in Figure 4, illustrating the license unit requirements table mechanism. In this example, the row selector is "platform-ID" so the platform-ID value determines which row is used. The issuer of this LURT of Figure 4 has established three unit requirement tiers for use in determining the unit requirements for that issuer's products. The reason for the tiers is not mandated by the license management system, but the issuer 25 (actually the user of the program 26) would probably be establishing three pricing tiers, each reflecting a different perspective on the relative utility of different platforms in supporting the use of various classes of product 17. The first column in Figure 4, Column A, specifies the use requirements for a class of products whose utility is highly sensitive to the characteristics of the specific platform on which they are run. This can be seen by observing that the unit requirements are different for every row in Column A. Products which use the second column (Column B) appear to have a utility which is more related to the class of platform on which they run. This is indicated by the fact that all the PC

platforms share a single value which is different from that assigned to the VAX platform. The final column (Column C) is for use with a class of products which is only supported on the VAX platform. Figure 4 is of course merely an example, and the actual LURT created by the license document generator 26 and stored in the license database 23 (as field 47 of the product use authorization 35) can be of any content of this general format, as desired by the license issuer.

Instead of always selecting the rows in LURT tables according to the platform ID of the execution platform, in order to handle the breadth of business practices that need to be supported by the license management facility, the LURT mechanism is extended by providing a "row selector" attribute in the LURT class structure. No default is provided although it is expected that the normal value for the row selector attribute will be "platform ID."

In the system of patent 4,937,863, a concept similar to that of the LURT of Figure 4 was provided, with rows selected by the platform ID and columns selected by some arbitrary means, typically according to product type. The system of this invention allows flexibility in the selection of both LURT row and column while continuing to provide backwards compatibility for licenses defined within the constraints of patent 4,937,863.

Some examples will illustrate potential uses for the row selector attribute. A customer may only want to pay for the use of a product during one or two months of the year; the product may be FORTRAN and the reason for this request may be that the company has a fairly stable set of FORTRAN subroutines that are given regular "annual maintenance" only during the months of May and June. To handle this customer's needs, the FORTRAN product would generate

an application subcontext which would contain a value representing the month of the year. Then, a LURT table would be defined with twelve rows, one for each month of the year. In some column, probably column A, a negative one (-1) would be placed in each month except for May and June. These two months would contain some positive number. The product use authorization would then have a LURDM field specifying a LURT for use to determine the units requirement, and would name this custom LURT table. The effect would be that the PUA could only be used during the months of May and June since negative one is interpreted by license managers to mean "use not authorized." This mechanism could also be used to do "time of day" charging. Perhaps charging fewer units per use at night than during the day. Also, if a subcontext was used that contained a year value, a type of license would be provided that varied in its unit requirements as time passed. For instance, it might start by costing 10-units per use in 1991 but then cost one unit less every year as time passed, eventually getting to the point where the unit requirement was zero.

Another example is font names. A specific customer may purchase a license giving it the right to concurrent use of 100-units of a large font collection; some of the fonts may cost more to use than others. For instance, Times Roman might cost 10-units per use while New Century Schoolbook costs 20-units per use. The problem is, of course, making sure that charges are properly made. The solution is to build a LURT table with a specified application subcontext as its row selector. A row is then created for each font in the collection and in Column A of the LURT, the number of units required to pay for use of the font would be specified. The print server would then specify the name of a font as the value of the application subcontext whenever it does an *lm_request_allocation()* call. This will allow charges to be varied according to font name.

A further example is memory size. Some products are more or less valuable depending on the size of memory available to support them. A software vendor wishing to determine unit requirements based on memory size will be able to do so by building LURT tables with rows for each reasonable increment of memory (probably 1-megabyte increments). Their applications would then sense memory size (using some mechanism not part of the license management facility) and pass a rounded memory size value to the license manager in a private context.

Other examples are environment and operating system. Some products may be valued differently depending on whether they are being run in an interactive mode or in batch. This can be accomplished by building LURT rows for each of the standard platform subcontexts that specify environment. Regarding operating system, it has been considered desirable by many to have a single product use authorization permit the use of a product on any number of operating systems, this conflicts with some vendors policies who do not want to have to create a single price for a product that applies to all operating systems. Thus, if an operating system independent license were offered for a C compiler, the price would be the same on MS-DOS, VMS, and/or UNIX. Clearly, it can be argued that the value of many products is, in part, dependent on the operating system that supports them. By using a row selector of operating system (one of the standard platform subcontexts), license designers could, in fact, require different numbers of units for each operating system. However, it might be more desirable to base the row selection on a private application subcontext that normally had the same value as the operating system subcontext. The reason for this is that the license designer might want to provide a default value for operating system names that were unknown at the time the LURT rows were defined. If this is the case, the product would contain a list of known operating systems and

pass the subcontext value of "Unknown" when appropriate. The LURT row for "Unknown" would either contain a negative one (-1) to indicate that this operating system was unsupported or it would contain some default unit requirement.

Another example is variable pricing within a group. One of the problems with a "group" license is that there is only one unit requirements field on the PUA for a group. Thus, all members of the group share a single unit requirement. However, in those cases were all members of the group can be appropriately licensed with a constant unit requirement yet it is desired to charge different amounts for the use of each group member, a LURT can be built that has rows defined for each group member. The row selector for such a group would be the standard platform subcontext "product name."

Many different types of license can be created using different combinations of contexts, duration and policy from the table of Figure 3. As examples, the following paragraphs show some traditional licensing styles which can be implemented using the appropriate values of the product use authorization fields 43-46.

A "system license" as it is traditionally designated is a license which allows unlimited use of a product on a single hardware system. The correct number of units must be allocated to the processor in advance and then an unlimited product use is available to users of the system. The product use authorization would have in the context field 44 a context template for a node name, the duration field would be "assignment" and the policy style field 43 would be "allocative".

A "concurrent use" license is one that limits the number of simultaneous uses of a licensed product. Concurrent use license units are only allocated when the product is being used and each simultaneous user of the licensed product requires their own units. In this case the context template, field 44, is a process ID, the duration field is "transaction" and the policy style 43 is "allocative".

A "personal use" license is one that limits the number of named users of a licensed product. This style of licensing guarantees the members of a list of users access to a product. Associated with a personal use type of product use authorization there is a list of registered users. The administrator is able to assign these users as required up to the limit imposed by the product use authorization; the number of units assigned to each user is indicated by the LURDM. It may be a constant or it may vary as specified in a LURT. The context template is "user name", the duration is "assignment", and the policy is "allocative".

A "site license" is one that limits the use of a licensed product to a physical site. Here the product use authorization contains for the context template either "network name" or "domain name", the duration is "assignment" and the policy style field 43 is "allocative".

Generally, a license to use a software product is priced according to how much benefit can be gained from using the product, which is related to the capacity of the machine it will run on. A license for unlimited use on a large platform such as a mainframe, where there could be thousands of potential users at terminals, would be priced at a high level. Here the style would be "allocative", the context template = "node", the duration = "assignment" and the LURDM may be "Column A" - the units, however, would be large, e.g., 1000. At the other end

of the scale would be a license for use on a single personal computer, where the field values would be the same as for the mainframe except the units would be "1". If a customer wanted to make the product available on the mainframe but yet limit the cost, he could perhaps get a license that would allow only five users at any given time to use the product; here the fields in the product use authorization would be: units = 5; style = allocative; context template = process; duration = transaction; LURDM = constant, 1-unit. This would still be priced fairly high since a large number of users may actually use the product if a session of use was short. A lower price would probably be available for a personal use license where only five named persons could use the product, these being identified only in the license server 10, not named by the license issuer 25. Here the fields in the product use authorization are: units = 5; style = allocative; context template = user name; duration = transaction; LURDM = constant, 1-unit.

An additional feature that may be provided for in the product use authorization 35 is license combination. Where there are multiple authorizations for a product, license checking requests sent by user nodes 16 may be satisfied by combining units from multiple authorizations. Individual product use authorizations may prohibit combined use. Thus, a licensee may have a license to use a product 17 on an allocative basis for a certain number of units and on a consumptive basis for another number of units (this may be attractive from pricing standpoint); there might not be enough units available for a particular context from one of these licenses, so some units may be "borrowed" from the other license (product use authorization), in which case a combination is made.

The interface between the program executing on the client or user 16 and the license server 10 or its delegatees 13 includes basically three procedure calls:

a request allocation, a release allocation and a query allocation. Figure 5 illustrates in flow chart form some of the events occurring in this client interface. The request allocation is the basic license checking function, a procedure call invoked when a software product 17 is being instantiated, functioning to request an allocation of license units, with the return being a grant or refusal to grant. Note that a product may use request allocation calls at a number of points in executing a program, rather than only upon start-up; for example, a request allocation may be sent when making use of some particular feature such a special graphics package or the like. The release allocation call is invoked when the user no longer needs the allocation, e.g., the task is finished, and this return is often merely an acknowledge; if the style is consumptive, the caller has the opportunity via the release allocation call to influence the number of units consumed, e.g., decrease the number due to some event. The query allocation call is invoked by the user to obtain information about an existing allocation, or to obtain a calling card, as will be described.

The request allocation, referred to as *lm_request_allocation()*, is a request that license units be allocated to the current context. This function returns a grant or denial status that can be used by the application programmer to decide whether to permit use of the product or product feature. The status is based on the existence of an appropriate product use authorization and any license management policies which may be associated with that product use authorization. License units will be allocated or consumed, if available, according to the policy statement found on the appropriate product use authorization. The product would normally call this function before use of a licensed product or product feature. The function will not cause the product's execution to be terminated should the request fail. The decision of what to do in case of failure to obtain allocation of license

units is up to the programmer. The arguments in a request allocation call are the product name, producer name, version, release date, and request extension. The product name, producer name, version and release date are the name of the software product, name of producer, version number and release date for specifically identifying the product which the user is requesting an allocation be made. The request extension argument is an object describing extended attributes of the request, such as units required, LURT column, private context, and comment. The results sent back to the calling node are a return code, indicating whether the function succeeded and, if not, why not, and a grant handle, returned if the function completes successfully, giving an identifying handle for this grant so it can be referred to in a subsequent release allocation call or query allocation call, for example.

The release allocation, referred to as *lm_release_allocation()*, is an indication from a user to the license manager to release or consume units previously allocated. This function releases an allocation grant made in response to a prior call to request allocation. Upon release, the license management style 38 determines whether the units should be returned to the pool of available units or consumed. If the caller had specified a request extension on the earlier call to request allocation which contained a units-required-attribute, and the number of units requested at that time are not the number of units that should be consumed for the completed operation, the caller should state with the units-consumed argument how many units should be consumed. The arguments of the release allocation are: grant handle, units consumed, and comment. The grant handle identifies the allocation grant created by a previous call to request allocation. The units-consumed argument identifies the number of units which should be consumed if the license policy is consumptive; this argument should only be used

in combination with an earlier call to request allocation which specified a units requirement in a request extension. Omission of this argument indicates that the number of units to be consumed is the same as the number allocated previously. The comment argument is a comment which will be written to the log file 24 if release units are from a consumptive style license or if logging is enabled. The result is a return code indicating if the function succeeded, and, if not, why not.

The query allocation, or *lm_query_allocation()*, is used by licensed products which have received allocations by a previous request allocation call. The query is to obtain information from the server 10 or delegatee server 13 about the nature of the grant that has been made to the user and the license data used in making the grant, or to obtain a calling card (i.e., a request that a calling card be issued). Typically, the item read by this query function is the token field 52 which contains arbitrary information encoded by the license issuer and which may be interpreted as required by the stub 19 for the licensed product software 17, usually when a "private" allocation style or context is being employed. The arguments in this procedure call are the grant handle, and the subject. The grant handle identifies the allocation grant created by a previous call to request allocation. The subject argument is either "product use authorization" or "calling card request"; if the former then the result will contain a public copy of the product use authorization. If this argument is a calling card request and a calling card which matches the previous constraints specified in that request can be made available, the result will contain a calling card. If the subject argument is omitted, the result will contain an instance of the allocation. The results of the query allocation call are (1) a return code, indicating whether the function succeeded, and, if not, why not, and (2) a result, which is either an allocation, a product use authorization or a calling card, depending on type and presence of the subject argument.

Referring to Figure 5, the flow chart shows the actions at the client in its interface with the server. When the software product 17 is to be invoked, the unit 18 is first executed as indicated by the block 60, and the first action is to make a request allocation call via the stub 19, indicated by the block 61. The client waits for a return, indicated by the loop 62, and when a return is received it is checked to see if it is a grant, at decision block 63. If not, the error code in the return is checked at block 64, and if a return code indicates a retry is possible, block 65, control passes back to the beginning, but if no retry is to be made then execution is terminated. If the policy is to allow use of the product 17 without a license grant, this function is separately accounted for. If the decision point 63 indicates a grant was made, the grant handle is stored, block 66, for later reference. The program 17 is then entered for the main activities intended by the user. During this execution of product 17, or before or after, a query allocation call can be made, block 67, though this is optional and in most cases not needed. When execution of the program 17 is completed, the grant handle is retrieved, block 68, and a release allocation call is made, block 69. A loop 70 indicates waiting for the return from the server, and when the return received it is checked for an error code as before, and a retry may be appropriate. If the release is successfully acknowledged, the program exits.

Referring to Figure 6, the actions of the server 10 or delegatee server 13 in executing the license management program 11 or 14, for the client interface, are illustrated in flow diagram form. A loop is shown where the server program is checking for receipt of a request, release or query call from its clients. The call would be a remote procedure call as discussed above, and would be a message communicated by a network, for example. This loop shows the decision blocks 71, 72 and 73. If a release allocation call is received, a list of products for which

authorizations are stored is scanned, block 74, and compared to the product identity given in the argument of the received call, block 75. If there is no match, an error code is returned to the client, block 76, and control goes back to the initial loop. If the product is found, the authorization is retrieved from the database 23, block 77 (there may be more than one authorization for a given product, in which case all would be retrieved, but only one will be referred to here) and all of the information is matched and the calculations made depending upon the management policy of Figures 3 and 4, indicated by the decision block 78. If a grant can be made, it is returned as indicated at block 79, or if not an error code is returned, block 80. If a release allocation call is received, indicated by a positive at the decision block 72, the grant handle in the argument is checked for validity at block 81. If no match is found, an error code is returned, block 82, and control passes back to the initial loop. If the handle is valid, the authorization for this product is retrieved from the database 23 at block 83, and updated as indicated by the block 84. For example, if the license management style is allocative, the units are returned to the available pool. Or, in some cases, no update is needed. The authorization is stored again in the database, block 85, and a return made to the client, block 86, before control passes back to the initial loop. If the decision block 73 indicates that a query allocation call is received, again the grant handle is checked at block 87, and an error code returned at block 88 if not valid. If the grant handle matches, the authorization is retrieved from the database 23, at block 89, and a return is made to the client giving the requested information in the argument, block 90.

The basic allocation algorithm used in the embodiment of the license management system herein described, and implemented in the method of Figures 5 and 6, is very simple and can handle a very large proportion of known license

unit allocation problems. However, it should be recognized that a more elaborate and expanded algorithm could be incorporated. Additions could be made in efforts to extend the allocation algorithm so that it would have specific support for optimizing unit allocation in a wider variety of situations. Particularly, sources of non-optimal allocations occurring when using the basic allocation algorithm are those that arise from combination and reservation handling.

The first step is formation of full context. The client stub 19 is responsible for collecting all specified platform and application subcontexts from the execution environment of the product 17 and forwarding these collected subcontexts to the license management server 13 or 10. The collection of subcontexts is referred to as the "full context" for a particular license unit allocation request.

The next step is retrieval of the context template. When the license manager receives an *lm_request_allocation()*, it will look in its list of available product use authorizations (PUA) to determine if any of them conform to the product identifier provided in the *lm_request_allocation()* call. The product identifier is composed of: product name, producer, version, release date. If any match is found, the license manager will extract from the matching PUA the context template. This template is composed of a list of subcontexts that are relevant to the process of determining unit requirements. Thus, a context template may indicate that the node-ID subcontext of a specific full context is of interest for the purposes of unit allocation. The context template would not specify any specific value for the node-ID; rather, it simply says that node-ID should be used in making the allocation computation.

The next step is masking the full context. Having retrieved the context template, the license manager will then construct an "allocation context" by filtering the full context to remove all subcontexts which are not listed in the context template. This allocation context is the context to be used in determining allocation requirements.

Then follows the step of determining if the request is new. The license manager maintains for each product use authorization a dynamic table which includes the allocation contexts of all outstanding allocations for that PUA (i.e., allocations that have been granted but have not yet been released). Associated with each entry in this table is some bookkeeping information which records the number of units allocated, the full context, etc. To determine if a recent *lm_request_allocation()* requires an allocation of units to be made, the license manager compares the new allocation context with all those allocation contexts in the table of outstanding allocations and determines if an allocation has already been made to the allocation context. If the new allocation context does not already exist in the table, an attempt will be made to allocate the appropriate number of units depending on the values contained in the LURDM structure of the PUA and any LURTs that might be required. If an allocation context similar to that specified in the new allocation request does exist in the table, the license manager will verify that the number of units previously allocated are equal to or greater than the number of units which would need to be allocated to satisfy the new allocation request. If so, the license manager will return a grant handle to the application which indicates that the allocation has been made (i.e., it is a "shared allocation" - the allocated units are shared between two requests.) If not, the license manager will attempt to allocate a number of units equal to the

difference between the number previously allocated and the number of units required.

The step of releasing allocations (Fig. 6, blocks 84-85) occurs when the license manager receives an *lm_release_allocation()* call; it will remove the record in its dynamic allocation table that corresponds to the allocation to be released. Having done this, the license manager will then determine if the allocation to be removed is being shared by any other allocation context. If so, the units associated with the allocation being released will not be released. They will remain allocated to the remaining allocation contexts. Some of the units might be released if the license manager determines that the number of allocated units exceeds the number needed to satisfy the outstanding allocation contexts. If this is the case, the license manager will "trim" the number of allocated units to an appropriate level.

In summary, the two things that make this algorithm work are (1) the basic rule that no more than one allocation will be made to any single allocation context, and (2) the use of the context template to make otherwise dissimilar full contexts appear to be similar for the purposes of allocation.

The license designer's task, when defining basic policy, is then to determine which contexts should appear to be the same to the license manager. If the license designer decides that all contexts on a single node should look the same (context template = node-ID), then any requests that come from that node will all share allocations. On the other hand, a decision that all contexts should be unique (i.e., context template = process-ID) will mean that allocations are never shared.

and stores a unit value indicating the number of licensing units for each product. When a user wishes to use a licensed product, a message is sent to the central license management facility requesting a license grant. In response to this message, the facility accesses the database to see if a license exists for this product, and, if so, whether units may be allocated to the user, depending upon the user's characteristics, such as the configuration of the platform (CPU) which will execute the software product. If the license management facility determines that a license can be granted, it sends a message to the user giving permission to proceed with activation of the product. If not, the message denies permission.

While the concepts disclosed in the patent 4,937,863 are widely applicable, and indeed are employed in the present invention, there are additional functions and alternatives that are needed in some applications. For example, the license management system should allow for simultaneous use of a wide variety of different licensing alternatives, instead of being rigidly structured to permit only one or only a few. When negotiating licenses with users, vendors should have available a wide variety of terms and conditions, even though a given vendor may decide to narrow the selection down to a small number. For example, a software product may be licensed to a single individual for use on a single CPU, or to an organization for use by anyone on a network, or for use by any users at terminals in a cluster, or only for calls from another specific licensed product, or any of a large number of other alternatives. A vendor may have a large number of products, some sold under one type of license and some under others, or a product may be a composite of a number of features from one or more vendors having different license policies and prices; it would be preferable to use the same license management system for all such products.

Distributed computing systems present additional licensing issues. A distributed system includes a number of processor nodes tied together in a network of servers and clients. Each node is a processor which may execute programs locally, and may also execute programs or features (subparts of programs) via the network. A program executing on one node may make remote procedure calls to procedures or programs on other nodes. In this case, some provision need be made for defining a license permitting a program to be executed in a distributed manner rather than separately on a single CPU, short of granting a license for execution on all nodes of a network.

In a large organization such as a company or government agency having various departments and divisions, geographically dispersed, a software license policy is difficult to administer and enforce, and also likely to be more costly, if individual licenses are negotiated, granted and administered by the units of the organization. A preferred arrangement would be to obtain a single license from the software producer, and then split this license into locally-administered parts by delegation. The delays caused by network communication can thus be minimized, as well as budgetary constraints imposed on the divisions or departments. Aside from this issue of delegation, the license management facility may best be operated on a network, where the licensing of products run on all nodes of the network may be centrally administered. A network is not necessary for use of the features of the invention however, since the license management can be implemented on a single platform.

Software products are increasingly fragmented into specific functions, and separate distribution of the functions can be unduly expensive. For example, a spreadsheet program may have separate modules for advanced color graphics, for

accessing a database, for printing or displaying an expanded list of fonts, etc. Customers of the basic spreadsheet product may want some, none or all of these added features. Yet, it would be advantageous to distribute the entire combination as one package, then allow the customer to license the features separately, in various combinations, or under differing terms. The customer may have an entire department of the company needing to use the spreadsheet every day, but only a few people who need to use the graphics a few days a month. It is advantageous, therefore, to provide alternatives for varied licensing of parts or features of software packages, rather than a fixed policy for the whole package.

Another example of distribution of products in their entirety, but licensing in parts, would be that of delivering CD ROMs to a customer containing all of the software that is available for a system, then licensing only those parts the customer needs or wishes to pay fees for rights to use. Of course, the product need not be merely applications programs, operating systems, or traditional executable code, but instead could also include static objects such as printer fonts, for example, or graphics images, or even music or other sound effects.

As will be explained below, calling and caller authorizations are provided in the system according to one feature of the invention, in order to provide technological support for a number of business practices and solve technical problems which require the use of what is called "transitive licensing." By "transitive licensing" is meant that the right to use one product or feature implies a right to use one or more other products or features. Transitive licenses are similar to group licenses in that both types of license consist of a single instrument providing rights of use for a plurality of products. However, transitive licenses differ from group licenses in that they restrict the granted rights by specifying that

the licensed products can only be used together and by further specifying one or more permitted inter-product calling/caller relationships. Some examples may help to clarify the use and nature of a transitive license: the examples to be explained are (1) two products sold together, (2) a give-away that results from narrow choices of licensing alternatives, (3) a client licensing method in a client/server environment, (4) impact of modular design, and (5) the impact of distributed design.

A software vendor might have two products for sale: the first a mail system, and the second a LEXIS™-like content-based text retrieval system. Each of these products might be valued at $500 if purchased separately. Some customers would be satisfied by purchasing the rights to use only one of these products. others might find that they can justify use of both. In order to increase the likelihood that customers will, in fact, purchase both products, it would not be surprising if the software vendor offered his potential customers a volume discount, offering the two products for a combined price of $800. The customers who took advantage of this combined offer would find that they had received two products, each of which could be exploited to its fullest capabilities independently from the other. Thus, these customers would be able to use the content based retrieval system to store and retrieve non-mail documents. However, from time to time, the vendor may discover that particularly heavy users of mail wish to be able to use the content based retrieval system only to augment the filing capabilities provided by the standard mail offering. It is likely that many of these potential customers would feel that $800 is simply too much to pay for an extended mail capability. The vendor might then consider offering these customers a license that grants mail users the right to use the content-based retrieval system only when they are using mail and prohibits the use of content

based retrieval with any other application that might be available on the customers system. This type of license is referred to below a "transitive license," and it might sell for $600.

Another example is a relational database product (such as that referred to as Rdb™) designed for use on a particular operating system, e.g., VMS. This relational database product has two components: (1) A user interface used in developing new databases, and (2) a "run-time" system which supports the use of previously developed databases. The developers of the database product might spend quite a bit of effort trying to get other products made by the vendor of the database product to use it as a database instead of having those other products build their own product-specific databases. Unfortunately, the other product designers may complain that the cost of a run-time license for the database product, when added to the cost of licenses for their products, would inevitably make their products uncompetitive. Thus, some mechanism would be needed that would allow one or another of the vendor's products to use the run-time system for the relational database product in a "private" manner while not giving unlicensed access to products of other vendors. No such mechanism existed, prior to this invention; thus, the vendor might be forced to sell the right to use its run-time system for the database product with its proprietary operating system license. Clearly, this combined license would make it possible for the vendor's products to use its database product without increasing their prices; however, it also would make it possible for any customers and third-parties to use the database product without paying additional license fees. However, had the system of the invention been available, the vendor could have granted transitive licenses for the run-time component of its database product to all the vendor's products. Essentially, these licenses would have said that the database run-time could be used without an

additional license fee if and only if it was used in conjunction with some other of the vendor's products. Any customer wishing to build a new relational database application or use a third-party application that relied on the vendor's database product would have had to pay the vendor for its database run-time license.

A proposed client/server licensing method provides yet another example of a problem which could be solved by transitive licensing. Typically, a client is only used by one user at a time, while a server can support an arbitrary number of clients depending on the level of client activity and the capacity of the machine which is supporting the server. While traditionally, server/client applications have been licensed according to the number of clients that a server could potentially support, this may not be the most appropriate method for licensing when the alternatives afforded by the invention are considered. The business model for the proposed client/server method requires that each client be individually licensed and no explicit licensing of servers is required to support properly licensed clients. Such a licensing scheme makes it possible to charge customers only for the specific number of clients they purchase. Additionally, it means that a single client can make use of more than one server without increasing the total cost of the system. The solution to this transitive licensing problem would be to provide a mechanism that would allow the clients to obtain license unit allocations and then pass a "proof" of that allocation to any servers they may wish to use. Servers would then support any clients whose proofs could be verified to be valid. On the other hand, if a client that had not received a proof of allocation attempted to use a server, the server would obtain a license allocation for that client session prior to performing any services. Such a solution has not been heretofore available.

As the complexity and size of the software systems provided to customers increases, it is found that the actual solution provided to customers is no longer a single product. Rather, customers are more often now offered solutions which are built up by integrating an increasing number of components or products, each of which can often stand alone or can be part of a large number of other solutions. In fact, a product strategy may rely almost exclusively on the vendor's engineering and selling a broad range of specialized components that can only be fully exploited when combined together with other components into a larger system. Such components include the relational database runtime system mentioned above, mail transport mechanisms, hyperinformation databases, document format conversion services, time services, etc. Because these components are not sold on their own merits, but rather on their ability to contribute to some larger system, it is unlikely that any one customer will be receiving the full abstract economic value of any one of the components once integrated into a system. Similarly, it can be observed that the value of any component once integrated into a larger system varies greatly from system to system. Thus, it may be found that a mail transport mechanism contributes a large part of a system whose primary focus is mail, however, it will contribute proportionally less of the value of a system that provides a broader office automation capability. As a result of these observations, the job of the business analyst who is attempting to find the "correct" market price for each component standing on its own, is more complex. In reality, the price or value of the component can only be determined when considering the contribution of that component to the full system or solution in which it is integrated. Attempting to sell the components at prices based on their abstract, independent values will simply result in overpriced systems.

Transitive license styles are particularly suited to dealing with pricing of modular components, since component prices can be clearly defined in relation to the other components or systems which they support. Thus, a vendor can charge a price of $100 for the right to use a mail transport system in conjunction with one product, yet charge $200 for the use of the same mail transport system when used by another product.

In addition to the "business" reasons for wanting to support transitive licensing, there is also a very good technical reason that arises from the growing tendency of developers to build "distributed products" as well as the drive toward application designs that exploit either tightly or loosely coupled multiprocessor systems; the availability and growing use of remote procedure calls has contributed to this tendency. This technical problem can be seen to arise when considering a product which has a number of components, each of which may run in a different process space and potentially on a different computer system. Thus, there might be a mail system whose user interface runs on one machine, its "file cabinet" is supported by a second machine and its mail transport system runs on yet a third machine. The simple question which arises is: "Which of the three components should check for licenses?" Clearly it must be ensured that no single component can be used if a valid license is not present. Thus, the answer to the question will probably be that all three components should check for licenses. However, the question is then presented: "Where are the licenses to be located?". This can become more complex.

Increasingly, the distributed systems being built are being designed so that it is difficult to predict on which precise machine any particular component will run. Ideally, networks are supposed to optimize the placement of functions

automatically so that the machine with the most available resource is always the one that services any particular request. This dynamic method of configuring the distribution of function servers on the network makes it very difficult for a system or network manager to predict which machines will run any particular function and thus very difficult for him to decide on which machines software licenses should be loaded.

Even if a system manager could predict which machines would be running the various application components and thus where the license units should be loaded, the situation would still be less than ideal. The problem arises from the fact that each of the components of the application would be independently making requests for license unit allocations. This behavior will result in a difficult problem for anyone trying to decide how many license units are required to support any one product. Given the mail example, the problem wouldn't exist if it were assumed that all three components (i.e., user interface, file cabinet, and transport system) were required by the design of the mail system to be in use simultaneously. If this were the case, it could be simply assumed that supporting a single activation of the mail system would require three units. However, in a real mail system, it will be inevitably discovered that many users will only be using just the user-interface and file-cabinet components of the system at one time. Thus, there will be some unused units available which could be used to authorize additional users. This situation might not be what is desired by the software vendor.

The problem of providing license support to multi-component products which are dynamically configured could be solved by viewing each of the product components as a distinct licensable product and by treating the problem as one

of transitive licensing, but a mechanism for accomplishing this has not been available. Essentially, a single license document would be created that stated that if any one of the components had successfully obtained a license to run, it could use this grant to give it the right to exploit the other components. Thus, in the example above, the user might start the mail system by invoking its user interface. This user interface code would then query the license management facility for a license allocation and once it has received that allocation, it would pass a proof of allocation to the other mail components that it uses. Each of the other components would request that the license management system validate that the "proof" is valid prior to performing any service; however, none of the other components would actually require specific allocations to be made to them. In this way, the complexity of licensing and managing networks of distributed applications can be significantly reduced.

SUMMARY OF THE INVENTION

In accordance with one embodiment of the invention, a license management system is used to account for software product usage in a computer system. The system employs a license management method which establishes a management policy having a variety of simultaneously-available alternative styles and contexts. A license server administers the license, and each licensed product upon start-up makes a call to the license server to check on whether usage is permitted, in a manner similar to that of patent 4,937,863. The license server maintains a store of the licenses, called product use authorizations, that it administers. Upon receiving a call from a user, the license server checks the product use authorization to determine if the particular use requested is

permitted, and, if so, returns a grant to the requesting user node. The license server maintains a database of product use authorizations for the licensed products, and accesses this database for updating and when a request is received from a user. While this license management system is perhaps of most utility on a distributed computer system using a local area network, it is also operable in a stand-alone or cluster type of system. In a distributed system, a license server executes on a server node and the products for which licenses are administered are on client nodes. However, the license management functions and the licensed products may be executing on the same processor in some embodiments.

The product use authorization is structured to define a license management policy allowing a variety of license alternatives by components called "style", "context", "duration" and "usage requirements determination method". The style may be allocative or consumptive. An allocative style means the units of the license may be allocated temporarily to a user when a request is received, then returned to the pool when the user is finished, so the units may be reused when another user makes a request. A consumptive style means the units are deducted from an available pool when a user node makes a valid request, and "consumed", not to be returned for reuse. The context value defines the context in which the use is to be allowed, such as on a particular network, by a particular type of CPU, by a particular user name, by a particular process, etc. The duration value (used in conjunction with the style component) concerns the time when the license units are to be deducted from the available pool of units, whether at the time of request, after a use is completed, etc. A usage requirements determination method may be specified to define or provide information concerning the number of license units charged in response to a license request from a user node; for example, some CPU platforms may be charged a larger number of license units

than others. A table may be maintained of usage requirements, and the determination method may specify how to access the table, for example. The important point is that the user node (thus the software product) can only make a request, identifying itself by user, platform, process, etc., and the license management facility calculates whether or not the license can be granted (that is, units are available for allocation), without the user node having access to any of the license data or calculation. There is a central facility, the license server, storing the license documents, and, upon request, telling the licensed products whether they can operate under the license terms.

An important feature of one embodiment is that the license administration may be delegated to a subsection of the organization, by creating another license management facility duplicating the main facility. For example, some of the units granted in the product use authorization may be delegated to another server, where the user nodes serviced by this server make requests and receive grants.

The license management facility cannot create a license itself, but instead must receive a license document (a product use authorization) from an issuer of licenses. As part of the overall license management system of the invention, a license document generator is provided which creates the product use authorizations under authority of the owner of the software, as negotiated with customers. Thus, there are three distinct rights in the overall license management facility of the invention: (1) the right to issue licenses, (2) the right to manage licenses, and (3) the right to use the licensed products. Each one of these uses the license document only in prescribed ways. The license issuer can generate a license document. The license manager (or license server as referred to herein) can grant products the right to use under the license, and can delegate parts of the

licensed units for management by another server, as defined by the license document; the way of granting rights to products is by responding to certain defined calls from the products. And, the licensed products can make certain calls to the license server to obtain grants of rights based upon the license document, inquire, or report, but ordinarily cannot access the document itself.

As explained above, transitive licensing is an important feature of one embodiment. This is the provision of a mechanism for one user node to get permission to use another software product located on another user node; this is referred to as a calling authorization and a caller authorization, using a "calling card," and these are examples of the optional features which must be specifically permitted by the product use authorization. A user node must obtain permission to make a procedure call to use a program on another node; this permission is obtained by a request to the license server as before, and the permission takes the form of a calling card. When a calling card is received by a second node (i.e., when the procedure call is made), a request is made by the second node to the license server to verify (via the product use authorization) that the calling card is valid, and a grant sent to the user node if allowed. In this manner, all nodes may have use of a program by remote calls, but only one consumes license units.

Another important feature of one embodiment is a management interface which allows a license manager to modify the license policy components of a license document maintained by at a license server in its database. Usually the license manager can only make modifications that restrict the license policy components to be more restrictive than originally granted. Of course, the management interface is used to make delegations and assignments, if these are authorized.

The license document interchange format is an important feature, in that it allows the license management system to be used with a wide variety of software products from different vendors, so long as all follow the defined format. The format uses data structures that are defined by international standards.

An important function is the filter function, used in the management interface and also in the client interface to select among elements in the data structures.

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as other features and advantages thereof, will be best understood by reference to the detailed description of specific embodiments which follows, when read in conjunction with the accompanying drawings, wherein:

Figure 1 is a diagram in block form of a distributed computer system which may be used to implement the license management operations according to one embodiment of the invention;

Figure 2 is a diagram of the content of a license document or "product use authorization" generated by the license document generator and stored by the license server in the system of Figure 1;

Figure 3 is a diagram of the alternatives for license style, context and duration making up the license management policy implemented in the system of Figure 1, according to one embodiment of the invention;

Figure 4 is a diagram of an example of a fragment of a license use requirements table (LURT) used in the system of Figure 1, according to one embodiment of the invention;

Figure 5 is a logic flow chart of a program executed by a user node (client), in the system of Figure 1, according to one embodiment of the invention;

Figure 6 is a logic flow chart of a program executed by a license server, in the system of Figure 1, according to one embodiment of the invention; and

Figure 7 is a diagram of the calls and returns made in an example of use of calling cards in the system of Figure 1.

Figure 8 is a diagram of an LDIF document identifier, according to an standard format;

Figure 9 is a syntax diagram of an LDIF document;

Figure 10 is a diagram of an LDIF document structure;

Figures 11, 13, 15, 17, 18, 19, 21-28 and 31-43 are syntax diagrams for elements of various ones of the LDIF data structures;

Figure 16 is a diagram of a license data structure;

Figures 12, 14 and 20 are examples of descriptions of data elements using a standard notation;

Figures 29 and 30 are examples of context templates used in the license management system;

Figures 44 and 45 are tables of attributes specific to filter and filter item type; and

Figure 46 is notation in a standard format for an example of a filter.

DETAILED DESCRIPTION OF SPECIFIC EMBODIMENTS

Referring to Figure 1, a license management facility according to one example embodiment of the invention is centered around a license server 10, which typically includes a CPU located in the customer's main office and executing a license management program 11 as will be described, under an operating system 12. The license server 10 communicates with a number of delegatees 13 which likewise include CPUs in departments or divisions of the company or organization, each also executing a license management program 14 under an operating system 15. The license management program 14 is the same as the program 11 executing on the main server 10; the only difference in the functions of server 10 and servers 13 is that the latter have a delegated subset of the license units granted to the server 10, as will be described. The CPUs 13 are in turn servers for a number of

users 16, which are CPU nodes where the licensed programs 17 are actually executed. The programs 17 executing on the user CPUs 16 are applications programs (or operating systems, etc.) which have added to them units 18 and 19, according to the invention, allowing them to make inquiry to the their server 13 (or 10) before executing and to report back after executing, using a client stub 19 in the manner of remote procedure calls, in one embodiment. A user node 16 may have many different programs 17 that may be executed, and the various user nodes 16 would usually each have a set of programs 17 different from the other user nodes, all of which would be administered by the license management program 14 or 11. The terms "program" and "licensed product" are used in reference to the element 17, but it is understood that the products being administered may be segments of programs, or functions or features called by another program, or even merely data (such as printer fonts), as well as complete stand-alone applications programs. The license server 10 communicates with the delegatee servers 13 by a network 21, as is usual in large organizations, and the delegatee servers 13 each communicate with their user nodes 16 by networks 22; these networks may be of the Ethernet, token ring, FDDI types or the like, or alternatively, the user nodes 16 may be merely a cluster of terminals on a multiuser system with the delegatee being a host CPU. The particular hardware construction of the user nodes, server nodes, communication networks, etc., and the operating systems 12 or 15, are of no concern regarding the utility of the features of the invention, the only important point being that the user CPUs 16 of the software products 17 in question can communicate readily and quickly with their respective server nodes 13 or 10. In one embodiment, remote procedure calls (RPCs) are used as the communication medium for the interfaces between components of the system, handling the inquiries and grants as will be described.

A remote procedure call is similar to a local procedure call but is made to a procedure located on a remote node, by way of a communications network.

The function of the unit 19 is that of a client stub, in a remote procedure call sense. The calls to the license server 10 are made through this stub 19, and returns are received by the stub 19 and passed on to the program 17. The stub 19 is responsible for obtaining the network addresses of other nodes on the network, such as the server 10. Also, the stub 19 is responsible for determining the context (as defined below) for passing on to the server 10. The unit 18 functions to execute a "private" type of license availability determination if this is used, rather than this task being done by the application program 17, but if the ordinary method of determination is employed (using the license server) as is usually the case, the unit 18 is merely code that starts the execution and passes calls and returns back and forth between the program 17 and the unit 19.

The license server 10, using the license management program 11, maintains a license data file 23 comprising a number of license documents or licenses (product use authorizations), and also maintains a log 24 which is a record of the usage activity of all of the user CPUs 16 of each of the licensed programs. The delegatee servers 13 would maintain similar license databases and logs. The license server 10 has no authority to originate a license, but instead must receive a license from a license issuer 25. The issuer 25 is again a CPU executing a license document generator program 26 under an operating system 27. The license issuer 25 may be under control of the producer 28 of the programs or software products being licensed, or may be controlled by a distributor who has received the authority to grant licenses from the producer or owner 28. The communications link 30 between the license issuer 25 and the license server 10 for

This mechanism permits the system of the invention to dispose of the cumbersome, explicit support of license types having different scope such as the cluster licenses, node licenses, and process licenses found in prior license management systems including that of patent 4,937,863. Instead of defining a limited set of scopes (cluster, node, etc.), the system of this invention provides a general mechanism which allows an effectively unlimited range of allocation scopes to be defined.

Transitive licensing, as referred to above, is supported by the system of the invention by (1) calling authorizations, which are statements made in field 49 of the product use authorization 35 for one product (the "caller") to permit that product to call another product (the "callee"), and, (2) caller authorizations, which are statements made in field 49 of the product use authorization for one product (the "callee") to permit it to be called by another product (the "caller").

If calling or caller authorizations are to be exploited by products, then whenever one product calls another product, it must pass the callee a calling card 49a. This calling card 49a is an encoding of an identification of the caller as well as a statement by the license management system that a license unit allocation has been made to the caller which is passing the calling card. This calling card is then passed by the callee to the license management system for validation and, if the either the product use authorization of the caller carries an appropriate calling authorization or the product use authorization of the callee carries an appropriate caller authorization, the use of the callee by the caller will be authorized without requiring any additional license unit allocations.

Referring to Figure 7, the intercomponent interactions that occur when either calling or caller authorizations are being used are illustrated. This figure shows a license management server 10, a caller product 17a named "Product-1" and a callee product 17b named "Product-2". When Product-1 starts to run, it will make an *lm_request_allocation()* call to the license management server 10 to obtain a grant handle for an allocation of some number of units of the Product-1 license. Either immediately, or at some later time, but always prior to making a call to Product-2, Product-1 will call *lm_query_allocation()*, passing the grant handle received earlier and specifying that it wants a calling card for the product named "Product-2." If the field 49 of the product use authorization 35 used to satisfy the grant represented by the grant handle carries a calling authorization in field 49 naming "Product-2," the license manager will create a calling card 49a which includes the statement that a calling authorization exists and pass this calling card back to Product-1. If the calling authorization does not exist, the calling card passed to Product-1 will contain a statement to that effect.

Once Product-1 has successfully obtained a calling card 49a from the license manager, it will then make a call to Product-2, passing the calling card along with any other initialization parameters that would normally be used when starting Product-2. Product-2 will then pass that calling card to the license manager as part of its *lm_request_allocation()* call and the license manager will determine if the calling card is valid. Note that calling cards become invalid once the process which received the calling card makes an *lm_release_allocation()* call or terminates abnormally. If the calling card is valid, and it indicates that a calling authorization is present, the license manager will verify this statement and if found to be true, will return a grant handle to Product-2. If, on the other hand, the calling card carries an indication that no calling authorization is present, the

license manager will attempt to find a product use authorization for Product-2 that contains a caller authorization naming Product-1 as an authorized caller. If the caller authorization is found, a grant handle will be passed back to Product-2. If not, the license manager will ignore the calling card and proceed with the normal *lm_request_allocation()* logic.

The requirement to be passing calling cards between products requires that both the caller and the callee be "aware" of the fact that calling and caller authorizations may be used. This is one of the few examples of a requirement for a product 17 to become actively involved in the licensing problem when using the licensing management system of the invention. However, since the use of calling/caller authorizations if a fairly "sophisticated" and powerful feature, it is considered acceptable to impose this burden on application coders.

MANAGEMENT INTERFACE

Referring to Figure 1, the license management program 11 executing on a server 10 includes a license management interface 33 which functions to allow a user at a console for the server 10 CPU or at a remote terminal to implement certain necessary operations. The management interface 33 is essentially the tools or mechanisms available to the license manager at the licensee's site to (a) load the various licenses received from issuers 25 into the database 23 and make them available for request allocation calls from the users, (b) remove the licenses from the machine when expired, (c) to make delegations if permitted, (d) to make assignments, (e) to make reservations, etc. Whatever the license manager is allowed to do to modify the license for his special circumstances (within the

original grant, of course), he does it by the mechanism of the management interface 33. Some licenses are not modified at all, but merely loaded. In a multiple machine environment, as on a network, there is considerable modification, as it is necessary to make sure the correct number of units are distributed onto the correct machines, the right people have access, other people don't have access, etc. Thus, in a network environment, there is extensive use of the management interface 33.

In reference to the terminology used in describing the management interface, as well as the license management system in general, it is helpful to note that the documentation conventions, data declarations, macro declarations, etc., for the object management used in one embodiment of the invention are according to the standards set forth in *OSI Object Management API Specification, Version 2.0*, X.400 API Association and X/Open Company Limited, 24 August 1990, a published document.

The specific operations available to the management interface 33 are to allow a manager to open and close a management session, register (load) objects in the license database 23, obtain a list of objects in the license database 23, and control a cursor (a cursor is a movable pointer to a member of a list of items). Once an object in the license database 23 is identified with the cursor, certain changes may be made in the object by a write function. For example, certain fields of a license document of Figure 2 or an LURT of Figure 4 may be changed in only specified ways as will be explained.

The operation of opening a session goes by the name of *lm_open_session()* and is used to establish a license management service session between a

management client and the service. Opening a session also creates a workspace to contain objects returned as a result of functions invoked within the session. Object management Objects can be created and manipulated within this workspace. Objects created within this workspace, and only such objects, may be used as Object arguments to the other license management service management functions used during the session established by a call to this function. More than one session may exist simultaneously.

The arguments that go with a *lm_open_session()* call are (a) the binding handle, which is binding information that defines one possible binding (a client-server relationship), and (b) a comment which will be inserted in the log file 24 if logging is enabled. The results from a *lm_open_session()* call are (a) a return code indicating whether the function succeeded, and, if not, why not, (b) a session, which is an established license management session between the management client and the license management service, and (c) a workspace that will contain all objects returned as a result of functions invoked in the session.

The close session call is referred to by *lm_close_session()* and functions to terminate the lm session. This function terminates the license service management session and makes the argument unavailable for use with other interface functions. The arguments that go with a *lm_close_session()* call are (a) the session which identifies the established lm session between the management client and the license management service, and (b) a comment which will be inserted in the log file if logging is enabled. The result of the call is a return code indicating whether the function succeeded, and, if not, why not.

The list function returns a set of selected objects in the license database 23, and uses the name *lm_list_licenses()*. This function is used to search the license database 23 and return a cursor which represents the first of one or more objects which match the specified filter. The specified filter will be applied to each object in the license database 23; all objects for which the filter evaluates true will be included in the object list accessible by the set_cursor function. The arguments that go with *lm_list_licenses()* are (a) session which identifies an established session between the management client and the license management service, and (b) a filter which is an object used to select license database 23 objects; license database objects will only be included in the object list headed by the cursor if they satisfy the filter - the constant no-filter may be used as the value of this argument if all license data objects are to be included in the object list. The results of the *lm_list_licenses()* call are (a) a return code indicating whether the function succeeded, and, if not, why not, and (b) a license list upon successful completion of this call containing a cursor which represents the first of one or more objects in the current license database 23 for which the specified filter evaluates true.

The register function is to register objects in the license database 23, and uses the name *lm_register()*. This function is used to register (i.e., load or create) new objects, or modify existing objects, in the license database 23; the objects which may be registered include only those which are subclasses of the license data class or history objects. The arguments are (a) session, which identifies an established session between the management client and the license management service, (b) license data object which is to be registered; if this argument is omitted, the comment argument is a required argument and a history object containing the comment will be registered in the license database 23, and (c)

comment, which will be inserted in the log file if logging is enabled. The result is a return code indicating whether the function succeeded, and, if not, why not. The errors possible when it does not succeed include data-expired, duplicate-object, no-such-session, memory-insufficient, network-error, etc., indicated by this return code.

The set cursor function establishes a new cursor, and is called by *lm_set_cursor()*. The arguments are (a) session, which identifies an established session between the management client and the license management service, (b) forward, which is a boolean value indicating if the direction in which the cursor is to be moved is forward or reverse, (c) filter which is used to eliminate cursors from the search for the next cursor that are not wanted; a new cursor will only be set if it satisfies the filter - the constant no-filter may be used as the value of this argument if any cursor is to be considered as the target cursor, and (d) the cursor which is to be used as the starting point in searching for the new cursor. The results are (a) a return code indicating whether the function succeeded, and, if not, why not, and (b) next-cursor, which is the requested cursor. The error codes in the return code may be end-of-list, not-a-cursor, etc.

After a session is opened, and an object such as a product use authorization or a LURT has been identified by the cursor, using the functions explained above, the management interface 33 is able to execute certain object management interface functions such as write or copy. By this mechanism, the management interface can modify certain limited attributes. None of these attributes can be modified in such a way that they reduce constraints established by corresponding attributes in the license data objects. The more important attributes which can be modified by the management interface 33 using this mechanism are:

(a) assignment: an assignment of some or all of the units granted on the associated product use authorization;

(b) reservation: a reservation of some or all of the units granted on the associated product use authorization;

(c) delegation: a delegation of the right to manage some or all of the units granted on the associated product use authorization, or if the associated license data is not a product use authorization, the delegation is of the right to use that license data;

(d) backup delegation: a statement of the right to manage some or all or the units granted on the associated product use authorization; this right is only active at times when the delegating server is not available;

(e) allocation: an allocation of units to a specific context;

(f) allocation period: the minimum duration of a single allocation - all allocated units cannot be allocated to a new context until a time period equal to the allocation period has passed since the units were last allocated;

(g) termination date: a date which is to override the value specified as the end date of the product use authorization 40 - this date must be earlier than specified;

(h) delegation permitted: an override of the delegation permitted flag of the associated license data;

(i) overdraft: the current overdraft level;

(j) overdraft logging: an override of the overdraft logging attribute of the associated product use authorization;

(k) comment: a comment created by the licensee;

(l) extended info: information not defined by the architecture which may be of use in managing the license data.

It will be noted that an assignment and a reservation are identical, the only difference being that a reservation is something optional, while an assignment is something that is required. If the duration is Assignment in the policy declaration of Figure 3, the license manager must assign some or all of the units before units can be allocated. Reservations, on the other hand, are made by the license manager using the management interface, regardless of the policy.

Thus, there are certain attributes that can be changed by a license administrator using the management interface at the server 10, but none of these can result in obtaining more extensive rights to use than granted by the product use authorization. In each case, the license administrator can limit the rights which will be allocated to users in some way that may be appropriate for the administrator for control purposes.

LICENSE DOCUMENT INTERCHANGE FORMAT

The major structural components of an ASN.1 encoded document which conforms to the specifications for the license management system discussed above will be described. The object identifier that is assigned to this data syntax, according to one embodiment, is that specified in ASN.1 as seen in Figure 8. The International Standards Organization or ISO, as it is referred to, defines how bit patterns are chosen to uniquely identify an object type, so the bit pattern set forth in Figure 8 would preceed each document used in the license management system so the document could be identified as being a document conforming to the prescribed License Document Interchange Format.

A document encoded according to this format is represented by a value of a complex data type called "license document interchange format document" of LDIFDocument, in this embodiment. A value of this data type represents a single document. This self-describing data structure is of the syntax defined in the international standard ASN.1 referred to above. The X/Open standard referred to above defines the conventions that must be used in employing this syntax, while the syntax itself is described in an OSI (Open Systems Interconnect, a standard administered by ISO) document identified as X.409 (referenced in the X/Open document identified herein).

The LDIFDocument data type consists of an ordered sequence of three elements: the document descriptor, the document header, and the document itself. Each of these elements are in turn composed of other elements. The overall structure of the LDIFDocument data type will be described, and the nature of the document descriptor and document header types. Then, the document content elements will be described in detail, as well as the various component data types used in the definition of the descriptor, the header and the content.

The LDIFDocument represents a single license document, with the syntax being shown in Figure 9 and the high-level structure of an LDIF document in graphical form being seen in Figure 10. The DocumentDescriptor of Figure 9 is a description of the document encoding, the DocumentHeader contains parameters and processing instructions that apply to the document as a whole, and the DocumentContent is the content of the document, all as explained below.

Referring to Figure 9, what this says is that an LDIFDocument is composed of (::= means "is composed of") a number of elements, the first thing in an

LDIFDocument is a bit pattern (tag) according to an international standard, indicating a certain type of document follows, which is indicated here to be "private" or vendor selected, the number 16373 in this case. Following the bit pattern which functions as a "starting delimiter" it is "implicit" that a "sequence" of elements must follow, where a sequence is distinguished from a set. A sequence is one or more of the elements to follow, whereas a set is exactly one of the elements to be listed. Implicit means that any file identified as LDIFDocument must have a sequence data type, rather than some other type. In the case of Figure 9, the sequence is document-descriptor, document header and document content; the document-content is mandatory, whereas the first two are optional. If an element in the sequence begins with a "0" it is a document-descriptor, "1" means a document-header, and "2" means it is a document-content. Again, it is implicit that the data following is of the format DocumentDescriptor, etc., in each case, and these are defined in Figure 11, Figure 13 and Figure 15.

Each file is in the tag-length-value format mentioned above, and also each element of a file containing multiple elements is of the tag-length-value format. The data stream could be examined beginning at any point, and its content determined by first looking for a tag, which will tell what data structure this is, then a length field will say how long it is, then the content will appear. These structures are nested within one another; a document containing several product-use-authorizations would be an LDIFDocument of the format of Figure 9, with a number of DocumentContent elements of Figure 15 following, with the length given for the LDIFDocument spanning the several PUAs, and the length given for each PUA being for the one PUA.

In Figure 11, the elements major-version and minor-version are seen to be "implicit integer". This means that because the element is of the type major-version, etc., it must be an integer. Various other implicit types are given in other syntax diagrams, such as character-string, boolean, etc.

In Figure 15, the license body is identified as being of the type "choice" meaning it can be one of PUA, LURT, GroupDefinition, KeyRegistration, etc. Thus, knowing this is a license-body does not mean the data type of the object is known; it is a bit further where the kind of a license-body becomes known. The defintion of a license body is not implicit, but instead is a chioce type.

The contents of the various data elements will now be described in detail with reference to Figures 11-43. Using these detailed descriptions, the exact format of each of the elements used in the LDIF can be interpreted.

The license document descriptor or DocumentDescriptor consists of an ordered sequence of four elements which specify the version level of the LDIF encoding and identify the software that encoded the document, with the syntax being shown in Figure 11. An example of the way a product called PAKGEN V1.0 is expressed in the DocumentDescriptor encoding is shown in Figure 12. The fields in the DocumentDescriptor syntax are major-version, minor-version, encoder-identifier and encoder-name. The major-version field is the primary indicator of compatibility between LDIF processors and the encoding of the present document; this major-version field is updated if changes are made to the system encoding that are not backward compatible. The minor-version field is the revision number of the system encoding. The encoder-identifier field is a registered facility mnemonic representing the software that encoded the document;

the encoder-identifier can be an acronym or abbreviation for the encoder name - this identifier is constant across versions of the encoder. The encoder-identifier should be used as a prefix to Named Value Tags in Named Value Lists to identify the encoder of the named value. The encoder-name field is the name of the product that encoded the document; the encoder-name string must contain the version number of the product.

The document header or DocumentHeader contains data that pertains to the document as a whole, describing the document to processors that receive it; the syntax is shown in Figure 13. An example of a document header is shown in Figure 14, using the hypothetical product PAKGEN V1.0 of Figure 12. The private-header-data contains the global information about the document that is not currently standardized; all interpretations of this information are subject only to private agreements between parties concerned, so a processor which does not understand private header data may ignore that data. The Title field is the user-visible name of the document. The Author field is the name of the person or persons responsible for the information content of the document. The Version field is the character string used to distinguish this version of the document from all other versions. The Date filed is the date associated with this document. Note that the nature and significance of the Title, Author, Version, and Date fields can vary between processing systems.

The content of an LDIF document is represented by a value of a complex data type called DocumentContent. An element of this type contains one or more LicenseData content element using a syntax as shown in Figure 15. There are no restrictions on the number, ordering or context of LicenseData elements. The structure of a LicenseData element is represented in Figure 16. No restrictions

are made on the number, ordering, or context of LicenseData elements. The license-data-header field of Figure 16 specifies that data, common to all types of license data, which describes the parties to the licensing agreement, the term of the agreement, and any constraints that may have been placed on the management of the license data encoded in the license body. The license-body is an element that contains one content element, including: product use authorizations, license unit requirements tables, group definitions, key registrations, and various forms of delegations. The Management-Info is an element that contains information concerning the current state of the license data; this element is not encoded by Issuers.

The license data header, called LicenseDataHeader, is represented as a syntax diagram in Figure 17. The license-id field provides a potentially unique identification of the encoded license data, so issuers of license data can generate unique license-ids to distinguish each issuance of license data; however, the architecture does not require this to be the case, since the only architectural restriction is that no two objects in any single license management domain may have the same value for license-id. The licensee field identifies the party who has received the rights reflected in the license data; there are at least two parties involved in all transfers of license data, first, the issuer of the license data, and second, the licensee or recipient of that data - it is anticipated that individual licensees will specify to those issuing them licenses what the licensee fields on their license data should contain. the term field identifies the term during which the license data may be used; the validity of license data can be limited by issuers to specific time ranges with given starting and ending dates, which are carried in the term element - attempts to use license data or products described by that data either before the start date or after the end date will result in conforming license

managers denying access to the license. Management-constraints identifies constraints placed on the right to manage the associated license data; these constraints can include (a) limiting the set of contexts permitted to manage the data, (b) limiting the set of platforms which may benefit from that management, and (c) limiting the right to backup and delegate the managed data. The signature provides the digital signature used by the issuer to sign the license data and identifies the algorithm used in encoding the signature. Issuer-comment is a comment provided by the issuer and associated with the license data.

The IssuerComment is of an informational nature and does not impact the process of authorizing product or feature use. This field is not included in the fields used to generate the signature for a license, thus, even if specified by an issuer, the IssuerComment can be omitted from a license without invalidating the license. If specified, the IssuerComment should be stored in the appropriate license data base with the associated license data. The IssuerComment can be retrieved by products which use the system and may be of particular utility to products in the "Software Asset Management" domain which are intended to extend or augment the administrative or accounting facilities or basic system components. Some examples of potential uses for this field are order information, additional terms and conditions, and support information. For order information, some issuers may wish to include with their loadable license data some indication of the purchase order or orders which caused the license data to be issued; licensees may find it useful to include this data in their license databases to assist in the license management process. For additional terms and conditions, the system will never provide automatic means for the management of all possible license terms and conditions, and so some issuers may wish to include summaries of non-system managed terms and conditions in the comment as a reminder. For

support information, the IssuerComment could be used to record the phone numbers or addresses of the responsible individuals within the issuing organization who should be contacted if there are problems with the data as issued.

A product use authorization as previously discussed in reference to Figure 2 is used to express the issuance of a right to use some product, product feature, or members of some product group. As such, it records the identity of the product for which use is authorized and specifies the means that will be used by the license manager to ensure that the licensee's actual use conforms to the terms and conditions of the license. Figure 18 illustrates a syntax diagram for a ProductUseAuthorization. Product-id identifies the name of the producer of the product or product feature of which usage rights are being granted as well as the name of that product; in addition, issuers of product use authorizations may specify a range of versions and/or releases whose use is controlled by the specific product use authorization. Units-granted - Contains the number of units of product use which are granted by the license. Management-policy defines the policy which is to be used in managing the granted software usage rights; this definition specifies the Style, Context-Template, Duration, and License Unit Requirements Determination Method which must be used. The calling-authorizations and caller-authorizations are as explained above in reference to calling cards. The execution-constraints field identifies constraints placed on the characteristics of execution contexts which may be authorized to benefit from the units granted by this Product Use Authorization. The product-token filed contains product specific data not interpreted in any way by any processors conformant with the architecture; software product producers 28 use this array to augment the capabilities of conformant license managers.

Some anticipated uses of the token field include language support, detailed feature authorizations, and product support number. For language support, a token could be constructed which contains a list of local language interface versions whose use is authorized; thus, if a product were available in English, German, French and Spanish, a token could be constructed listing only English and German as the authorized languages. For detailed feature authorizations, some license issuers will wish to have very fine control over the use of features in a complex product; however, they may not wish to issue a large number of individual Product Use Authorizations to "turn on" each feature, so these vendors could construct tokens which contain lists of the features authorized or whose use is denied. For product support number, some issuers may wish to include on the product use authorization, and thus make available to the running product, some information concerning the support procedures for the product; for example, an issuer might include the telephone number of the support center or a support contract number, and the product could be designed to retrieve this data from the license manager and display it as part of Help dialogues.

The LURT's or license use requirements tables of Figure 4 provide a means by which issuers of licenses, whose LURDM is dependent on the type of platform on which the product is run, can store information describing the relationship between the platform type and unit requirements. A syntax diagram for a LURT is shown in Figure 19. In Figure 20, an example of how the LURT of Figure 4 might be encoded is illustrated. Lurt-name specifies the name by which the LURT is to be known to conforming license managers. The rows field models a list of multicolumn lurt rows. Platform-id identifies the platform for which this LurtRow provides license unit requirements. The lurt-columns field provides a list of one or more lurt column values; the first value provided is

assigned to column-1 of the lurt-row, the second value provided is assigned to column-, etc. A lurt column value of -1 indicates that use of the product or feature is not authorized, while a lurt column value of 0 or greater indicates the number of units that must be allocated in order to authorize product use on the platform described by this lurt-row. All unspecified columns (e.g., columns whose number is greater than the number of column values provided in the lurt columns element) will be considered to contain the value -1.

In reference to Figure 19, to use the row-selector feature mentioned above, the platform-ID element would be replaced with *row-selector* which would be implicit of Context. Also, in Figure 34 described below, in the lurdm-kind element, *row-selector* would be included if the row-select feature is to be used.

As discussed above, Figure 4 provides an example of a hypothetical LURT, illustrating the LURT mechanism, where the issuer of this LURT table has established three unit requirement tiers for use in determining the unit requirements for that issuer's products. Figure 20 provides an example of how the LURT presented in Figure 4 might be encoded.

A group definition is used to define and name a license group. Once so defined, the name of this group can be used on product use authorizations in the same manner as a product name. Since a single product use authorization specifies the management policy for all members of the group, the members of that group must be compatible in their licensing styles, i.e., a personal use type product can not be mixed with a concurrent use product in the same group. Figure 21 shows a group definition syntax diagram. Group-name is the name which must appear on Product Use Authorizations for this group. Group-version

specifies the current version of this group; the requirements for matching between the version information on a product use authorization and that on a specified group definition are the same as those rules which require matching between produce use authorizations and the Release Date data provided by products. Group-members lists those products or features which are components of the named group.

A key registration is used by a producer 28 or issuer 25 who have been registered as authorized license issuers and provided with an appropriate public and private key pair. The key registration identifies the public key which is to be used by conforming license managers 10 in evaluating signatures 53 created by the named issuer 25 or producer 28. A key registration syntax diagram is shown in Figure 22. Key-owner-name provides the name which must be used in either of, or both, of the Producer and Issuer fields of license data generated by the issuer; the key-owner-name must be identical to that specified in the Issuer field of the header record. Key-algorithm identifies the registered algorithm that is to be used when producing digital signatures with this key. Key-value identifies the public key.

An issuer delegation is typically issued by a producer 28 and authorizes the named issuer 25 to issue licenses for products produced by the producer. An issuer delegation syntax diagram is shown in Figure 23. Delegated-issuer-name identifies the name which must appear in the Issuer field of any Product Use Authorization generated using the License Issuer Delegation. Delegated-product-id identifies the products whose licenses the named issuer is authorized to issue. Delegated-units-granted, if specified, indicates that the use of this IssuerDelegation is to be managed in the style of a consumptive license; the value of this attribute

gives the number of units for which license documents may be generated (i.e., if granted 1000 units by a Producer, an Issuer can only issue 1000 units.) Template-authorization provides a "template" Product Use Authorization whose attribute values must be included on any Product Use Authorization generated using this IssuerDelegation; in the case of attributes which have a scalar value (i.e., Version, Release Date, etc.), the Issuer may issue licenses with more restrictive values than those specified on the Template Authorization. Sub-license-permitted indicates whether the Issuer identified on this IssuerDelegation may issue an IssuerDelegation for the delegated-product-id.

A license delegation, as shown in a syntax diagram of Figure 24, is used to delegate the right to manage license data. Such delegations are created by the licensee (by the license manager), if authorized by the issuer 28. A backup delegation, also shown in Figure 24, is used by one license management facility to authorize another to manage the delegated rights in the case that the delegating license manager is not running. The delegated-units field specifies the number of units whose management is being delegated; this may only be specified when a product use authorization is being delegated. Delegation-distribution-control defines the mechanisms by which the distribution and refreshing of the delegation will be accomplished. Delegatee-execution-constraints identifies any constraints which are placed on the execution-context of the Delegatee; these constraints are applied in addition to those which are a part of the delegated License Data. Assignment-list identifies any assignments of the delegated units that must be respected by the delegatee. Delegated-data stores a copy of the LicenseData received from the issuer that is the subject of the delegation; the delegated data is not provided when the LicenseDelegation element is included in a DelegationList.

The management information or ManagementInfo element records information concerning the current state of the LicenseData with which it is associated. A syntax diagram of the ManagementInfo element is shown in Figure 25. The assignments field identifies a list of one or more assignments which may be outstanding for the units on the associated product use authorization. Reservations identifies a list of one or more reservations which may be outstanding for the units on the associated product use authorization. Delegations identifies a list of all outstanding delegations. Backup-delegations identifies all outstanding backup delegations. the allocations field provides detailed information about outstanding allocations which involve units from the associated product use authorization. Registration-date is the date on which the LicenseData was registered in the license database. Registrar is the context which caused the LicenseData to be registered. Local-comment is a comment field. Termination-date means a license defined date after which the license data may not be used; this date must be earlier than the end-date specified in the license data's term record. The extended-info field allows additional information concerning the state of the LicenseData and its handling by the license manager that is not standardized.

The defined types of elements will now be described. These defined type are:

Allocation	ManagementPolicy
Assignment	Member
Context	NamedValue
DistributionControl	NamedValueList
ExecutionConstraints	ProductID
IntervalTime	Signature

LicenseID	Term
LUDRM	Version
ManagementConstraints	

The allocation element records the information concerning a single unit allocation, and is shown in a syntax diagram in Figure 26. Allocation-context specifies the context to which the allocation was made. The allocation-lur field specifies the license unit requirement which applies to the allocation-context; this license unit requirement is calculated without consideration of any allocation sharing which may be possible. The allocation-group-id field identifies the "allocation-group" for the current allocation, in which an unshared allocation will always have an allocation group id of 0; allocations which utilize shared units will have an allocation group id which is shared by all other allocations sharing the same units.

The assignment element is shown in syntax diagram in Figure 27. Assigned-units identifies the number of units which are assigned. Assignment-term identifies the start and end of the assignment period. Assignee identifies the context to which the assignment is made.

The context element is shown in syntax diagram in Figure 28. The SubContext-type field identifies the type of subcontext, and this type can be either standard or private; if standard, the type value will be taken from the standard-subcontext-type enumeration: (a) network-subcontext means the subcontext value identifies a network; (b) execution-domain-subcontext means the subcontext value is the name of the management domain within which the caller is executing; (d) login-domain-subcontext means the subcontext value is the name of the

management domain within which the user of the caller was originally authenticated or "logged in"; (d) node-subcontext means the subcontext value is the name of a node; (e) process-family-subcontext means the subcontext value is an implementation specific identifier for a group of related processes; (f) process-ID-subcontext means the subcontext value is an implementation specific process identifier; (g) user-name-subcontext means the subcontext value is a user name; (h) product-name-subcontext means the subcontext value is the same as the product name found on the Product Use Authorization; (i) operating-system-subcontext means the subcontext value is a character string representation of the name of the operating system; (j) platform-ID-subcontext means the subcontext value is an identifier that describes the hardware platform supporting the context. The subcontext-value field is the value of the subcontext.

As discussed above, license data is always used or allocated within, or for the benefit of, some named licensing context. This context name is constructed by concatenating the values of all subcontexts into a single context name. A Context Template specifies those components of the context name which should be used in calculating license unit requirements. The management system determines the need to perform a unit allocation each time license units are requested. The full context on whose behalf the allocation should be made is obtained for each requested authorization. The system will mask the full context to exclude all sub-contexts not specified in the context template and then determine if the resulting context already has units allocated to it. If not, units will be allocated according to the specification of the LURDM, otherwise, the units previously allocated will be shared by the new context. Thus, if a given product authorization contains a context specification of NODE + USER_NAME, each context which requests license unit allocations and which has a unique pair

of NODE + USER_NAME subcontext values will require an explicit grant of license units to be made. On the other hand, any contexts which share the same pair of NODE and USER_NAME subcontext values will be able to "share" a single allocation of license units. The requirement for specific allocations of units and the ability to share units is exhibited in Figure 29 which attempts to provide a "snapshot" of the units allocated for the product FOOBAR V4.1 at a particular instance. It is seen from the figure that although presented with five unique full contexts, only four of them are unique when looking only at those portions of each context which are described by the Context Template (ie: NODE + USER_NAME). A unit allocation must be made for each of the four instances of unique contexts, when masked by the Context Template. The fifth context can share allocated units with another context. Thus, the total requirement to support product use as described in this example would be 40-units (ie: four allocations of ten units each). Significant changes in the unit requirements can be achieved by making small modifications to the Context Template. Figure 30 shows the same contexts as in Figure 29 but a Context_Template of NODE. The total unit requirement for this example would be three units (three allocations of ten units each) rather than the forty units required in the previous example.

The distribution control element defines the mechanism that will be used for distributing the subject delegation and records some status information concerning the distribution of that delegation. A syntax diagram of the distribution control element is shown in Figure 31. Distribution-method identifies the means by which the delegation will be distributed, and the alternatives are refresh-distribution, initial=distribution-only, and manual-distribution. Refresh-distribution means the license manager shall be responsible for the initial distribution of the delegation and for ensuring that refresh delegations are

properly distributed. Initial-distribution-only means the license manager shall be responsible for the initial distribution of the delegation, however, distribution of refresh delegations will be made by some other means. Manual-distribution means the distribution of the delegation will be under the control of some other mechanism (perhaps a license asset manager). Current-start-date is the time that the last successful initial or refresh delegation distribution was performed. Current-end-date identifies the last date on which the most recent delegation distribution was performed. Refresh-interval identifies the period of time between attempts to refresh the delegation; the refresh-interval may not be longer than the maximum-delegation-period and should normally be less than that in order to ensure that refresh delegations are distributed prior to the expiration of the previous delegations that they are replacing. Retry-interval identifies the amount of time to wait for an unsuccessful distribution attempt to try again. Maximum-retry-count identifies the maximum number of times that an unsuccessful distribution attempt may be retried. Retries-attempted records the number of unsuccessful retry attempts which have been made since the last successful initial or refresh delegation distribution was performed.

The execution constraints elements place limits on the environments and contexts which may receive allocations. A syntax diagram of the execution constraints element is shown in Figure 32. Operating-system contains a list of zero or more operating systems on which the use of the subject license is authorized; if no operating systems are specified, it is assumed that license use is authorized on all operating systems. Execution-context specifies a list of zero or more full or partial context names which identify the contexts within which products described by the license data may be executed; if no context names are specified, the licensed products may be executed in any context controlled by the licensee.

Environment-list identifies those environments within which the licensed product may be used.

The interval time element is defined by the syntax IntervalTime ::= UTCTime.

The license ID element uniquely identifies the license data it is associated with, and is described by the syntax diagram of Figure 33. Here issuer uniquely identifies the issuer of the license data as well as the name space within which the LicenseID Number is maintained. While the issuer name will typically be the same as the name of the issuer's company or personal name, this is not a requirement. For instance: The issuer name for Digital Equipment Corporation is "DEC," an abbreviation of the corporate name. Valid contents of the Issuer field are maintained in the an Issuer Registry. The serial-number provides a unique identification or serial number for the license data. The amendment field is an integer which is incremented each time license data is amended by its issuer, with the first version of any license data carries the amendment number 0; an amendment can only be applied to license data if that license data has identical Issuer and Number values and an amendment number less than the number of the amendment to be applied.

The license units requirements determination method or LURDM element is shown in syntax diagram in Figure 34. The combination-permitted field indicates whether conforming license managers are permitted to combine together into a common pool the units from different product use authorizations if those produce use authorizations have the same product record value; for example, if combination is permitted and a single license manager discovers in its database

two 500-unit authorizations for the use of DEC Cobol, the license manager would be permitted to combine these two authorizations into a logical grant of 1000 units. The overdraft-limit modifies the behavior of a conforming license management facility in those cases where it is found that there are zero or fewer license units available for use at the time of a request for the allocation or consumption of additional license units. Operation of overdraft is different depending upon whether allocative, or consumptive style is being used. In using with allocative style, an allocation is granted even though the remaining units are zero or less, up to the overdraft-limit. In using with consumptive style, the license is authorized to accumulate a negative balance of license units, up to the overdraft-limit. Overdraft-logging-required indicates whether all license grants which are the result of overdraft use must cause a log record to be generated. When the allocation-size field is non-zero, then all unit allocations and delegations must be made in sizes which are whole number multiples of the allocation-size value. Lurdm-kind identifies the method by which license unit requirements will be calculated once the requirement for an allocation has been discovered, the permitted alternatives being (a) LURT which specifies that license unit requirements are to be determined by lookup in the LURT which is associated with the current license, (b) Constant which specifies that license unit requirements are constant for all platforms on which the licensed product or product feature may run, and (c) Private-LURDM which specifies that license unit requirements are to be determined by the licensed product, not by the license management facility. The named-lurt-id specifies the name of the LURT table to be used in determining license unit requirements if the LURDM-kind is specified as LURT; if the LURDM-kind is specified as LURT and no table is explicitly named, the name of the table to be used is constructed from the issuer name on the product use authorization. Lurdm-value specifies the LURT column to be

used when LURDM-kind = LURT; however, when LURDM-kind = Constant, the Lurdm-value field contains the precise number of units to be allocated or consumed. Default-unit-requirement specifies the unit requirement value to be used when the appropriate LURT does not have a row corresponding to the appropriate platform ID; when specified on a product use authorization with Style = Allocative, the context template will change to Process + Product_Specific and the Duration will change to Transaction in cases of unrecognized Platform ID's.

The management constraints element is shown in a syntax diagram in Figure 35. The management-context field specifies a list of zero or more partial context names which identify the specific contexts within which the license data may be managed. If no management contexts are specified, the license data may be managed within any context controlled by the licensee. The contexts used in specifying Management Context Constraints may only contain the Network, Domain, and Node subcontexts. Specifying a list of management contexts does not effect whether or not the license data can be used within other contexts. For example, unless otherwise restricted, license data with a specified management context can be remotely accessed from or delegated to other nodes in a network. The management-scope field defines the maximum permitted size of the license management domain within which the license data may be managed or distributed, these being single-platform, management-domain, or entire-network. Single-platform constrains the license management domain for the subject license data to be no larger than a single platform. Management-domain constrains the license management domain for the subject license data to be no larger than a single management domain. Entire-network constrains the license management domain for the subject license data to be no larger than a single wide area network; that

network which contains the platform on which the license units were initially loaded. Although technology may not exist to detect the interorganizational boundaries of a wide area network (i.e., what is on the Internet as opposed to being on a company's own network), the assumption is that interorganization and internetwork sharing of licenses will normally be considered a violation of license terms and conditions. The backup-permitted field indicates if the Issuer has authorized the use of backup delegations for this data. Delegation-permitted indicates if the Issuer has authorized the licensee to delegate this data. Maximum-delegation-period identifies the longest interval during which a delegation may be valid; by default, delegations have a life of 72-hours.

The major elements of the management policy specification are shown in Figure 3, as previously discussed. A syntax diagram for the management policy element is shown in Figure 36. For the Style field, three fundamental styles of license management policy are supported, allocative, consumptive, and private-style, as explained above. Only one of these styles may be assigned to any single product use authorization. The Context-template specifies those components (sub-contexts) of the execution-context name which should be used in determining if unit allocations are required. The Duration defines the duration of an allocation of license units to a specific context or the duration of the period which defines a valid consumptive use. For durations of type "Assignment," the specification of a Reassignment Constraint is also provided for. Three types of Duration_Kind are supported, these being Transaction, Assignment and Immediate, as explained above. The lur-determination-method stores information used in calculating the number of units that should be allocated or consumed in response to a license request. The allocation-sharing-limit identifies the largest number of execution contexts that may share an allocation made under this management policy; an

allocation-sharing-limit of 0 indicates that the number of execution contexts that may share an allocation is unlimited. The reassignment-constraint specifies a minimum duration of assignment; although there is normally no constraint placed on how frequently granted units may be reassigned, an issuer may constrain reassignment by specifying this minimum duration of an assignment, in which case reassignment of assigned units will not be supported until the amount of time specified in the Reassignment Constraint has passed. If an assignment of some particular set of units has been delegated and the delegation period for that delegation has not terminated, cancellation of the delegation must be performed prior to reassignment.

The member element identifies a specific licensed product which may be part of a calling authorization or group definition, and is shown in syntax diagram in Figure 37. Member-product identifies the product which is a member. Member-signature is constructed from the product and token fields of the called member structure as well as the product and issuer fields of the calling product. Member-token provides the data which should be used as the product token for this member.

Named values are data elements with a character string tag that identifies the data element, and have a syntax as shown in Figure 38, which also shows the syntax for ValueData and named value list. A named value list models a list of named values, with an example being shown in Figure 39. In Figure 38, Value-Name uniquely identifies the value; no standard value names are defined, and the period character can be used as a part of the value name to form a hierarchical tag registry at the discretion of the issuer. Value-data is the data that has been named; data types are selected from the possible Value Data types, seen in the

Figure. Value-boolean means the named data is a boolean value. Value-integer means the named data is an integer value. Value-text means the named data is a StringList value. Value-general means the named data is a stream of bytes in any format. Value-list means the named data is a list of named data values.

The product ID explicitly identifies the product which is the subject of the license data with which it is associated, with the syntax for ProductID being shown in Figure 40. The version and release date fields provide a mechanism for defining which specific instances of the licensed product are described in the associated license data. The Producer field is a registered name which identifies the producer of the licensed feature; in the case of Group Names, the Producer is always also the Issuer of the group. The Product-name identifies a licensed software feature. The First-version identifies the earliest version of the product whose use is authorized. The Last-version identifies the latest version of the product whose use is authorized. The First-release-date identifies the earliest release of the product whose use is authorized. The Last-release-date identifies the latest release of the product whose use is authorized. Conforming license managers are required to interpret the contents of these fields in the most restrictive way possible. Thus, if a license is issued with Last-version = 3.0 and a Last-release-Date of 1-Jan-1991, then the use of version 2.0 of the licensed product would be unauthorized if it had a release date of 2-Jan-1991. If either a First-version or First-release-date is specified without a matching Last-version or Last-release-date, use of the produce is authorized for all versions or release dates following that specified. Similarly, if either a last-version or Last-release-date is specified without a matching First-version or First-release-date, use of the produce is assumed to be authorized for all versions or release dates prior to that specified. Issuers should typically only specify one of either First-version or First-release-

date. This is the case since it is anticipated that these fields will typically refer to events which occurred prior to the moment of license data issuance. Thus, it should normally be possible for the issuer to state unambiguously with only one of these two fields which is the oldest implementation of the product that is to be authorized. The architecture does permit, however, both fields to be used in a single product authorization.

The signature element is used to establish the integrity and authorship of the license data with which it is associated. A syntax diagram for the signature element is shown in Figure 41. The Signature-algorithm field identifies the registered algorithm that was used to produce the digital signature. Signature-parameters are the values of the algorithm's parameters that are to be used; the need for and syntax of parameters is determined by each individual algorithm. Signature-value is an enciphered summary of the information to which the signature is appended; the summary is produced by means of a one-way hash function, while the enciphering is carried out using the secret key of the signer (Issuer).

The term element defines an interval during which the license data is valid, and is shown in syntax diagram form in Figure 42. The fields are start-date and end-date. Start-date identifies the first date of the term; if not specified, the license data is considered valid on any date prior to the end-date. End-date identifies the last date of the term; if not specified, the license data is considered valid on any date after the Start-date. While the Start-date is always either omitted or specified as an absolute date, the End-date can be either absolute or relative. If the End-date is specified as a relative or "interval" date and the Start-date has been omitted, the date of license registration will be used as the effective

start date in computing the valid term of the license data. It should be noted that the system does not specify the mechanism by which system dates are maintained by platforms supporting system components. Instead, the system always accepts that system time returned to it as correct. Thus, the reliability of the management of license data which specifies terms is dependent on the time management function of the underlying platform.

The version element identifies a four-part version of the licensed software product or feature. A syntax diagram of the version element is shown in Figure 43. The schematics of each of the four parts is not detailed, but it is required that producers who wish to permit version ranges to be specified on product use authorizations ensure that the collating significance of the four parts is maintained. When comparing versions, Part-1 is considered first, then Part-2, then Part-3, and finally, Part-4. Part-1 identifies a major modification to the versioned object. Part-2 identifies a modification to the versioned object which is less significant than a modification which would cause a change in the Part-1 value. Part-3 identifies a modification to the versioned object which is less significant than a modification which would cause a change in the Part-2 value. Part-4 identifies a modification to the versioned object which is less significant than a modification which would cause a change in the Part-3 value.

FILTERS

An important feature is the use of filters in the license management program 11, including the client interface 31 and the management interface 33. A filter is used is select items in the license database 23, for example. Various

selection mechanisms are used in picking out or doing lookups in database technology; filters are one of them. The filter engine used in the license management system 11 of Figure 1 is generally of a known construction, with the exception of the select filter item type as will be described, which allows a complex rather than a flat data format to be selected from. The feature that is of importance to this embodiment is the way of specifying items as an input to the filter function , rather than the filter function itself. Thus, there is described below a template for specifying input to the filter engine. This is as if a form were used as the input, with blanks on the form; by filing in certain blanks these would be the items selected on, the blanks not filled in would be "don't care".

An instance of the class *filter* is a basis for selecting or rejecting an object on the basis of information in that object. At any point in time, a filter has a value relative to every object - this value is false, true or undefined. The object is selected if and only if the filter's value is true. This concrete class has the attributes of its superclass - *Object* - and the specific attributes listed in the table of Figure 44.

A filter is a collection of simpler filters and elementary filter-items together with a Boolean operation. The filter value is undefined if and only if all the component filters and filter-items are undefined. Otherwise, the filter has a Boolean value with respect to any object, which can be determined by evaluating each of the nested components and combining their values using Boolean operation (components whose value is undefined or ignored). The attributes specific to *filter* as shown in Figure 44 are (a) *filter items* which are a collection of assertions, each relating to just one attribute of an object, (b) *filters* which are a

collection of simple filters, and (c) *filter type* which is the filter's type, of one of the following values: And, Or, Not.

An instance of the class *filter item* is a component of a *filter*. It is an assertion about the existence or values of a single attribute of a license data object or one or its subobjects. This concrete class has the attributes of its superclass - *object* - and the specific attributes listed in the table of Figure 45.

The value of a filter item is undefined if: (a) the Attribute Types are unknown, or (b) the syntax of the Match Value does not conform to the attribute syntax defined for the attribute type, or (c) a required Attribute is not provided. The attributes specific to *filter item* as shown in Figure 45 are (a) *filter item type* which identifies the type of filter item and thereby the nature of the filter, and its value must be one of

equality	less
inequality	present
greater or equal	select
less or equal	request candidates
greater	simulate request

(b) *attribute type* which identifies the type of that attribute whose value or presence is to be tested; the value of All Attributes may be specified, (c) *match value* which is the value which is to be matched against the value of the attribute, (d) *filter* which identifies the filter to be used in evaluating a selected subobject of the current object; the filter is ignored if the *filter item type* is not *select* or if the specified attribute type is not present in the object, and upon evaluation of the *filter* the value of *filter item* will be set to that of the *filter*, (e) *initial substring*, if present, this is the substring to compare against the initial portion of the value of

the specified attribute type, (f) *substring*, if present, this is the substring(s) to compare against all substrings of the value of the specified attribute type, (g) *final substring*, if present, this is the substring to compare against the final portion of the value of the specified attribute type, and (h) *license request*, if present, this is license request against which the appropriate license data objects should be evaluated; this attribute may only be specified if the value of the filter item type is either Request Candidates or Simulate Request.

An instance of enumeration syntax *Filter Type* identifies the type of a filter. Its value is chosen from one of the following: (a) *And* means the filter is the logical conjunction of its components; the filter is true unless any of the nested filters or filter items is false, or if there are no nested components, the filter is true; (b) *Or* means the filter is the logical disjunction of its components; the filter is false unless any of the nested filters or filter items is true, or, if there are no nested components, the filter is false; (c) *Not* means the result of the filter is reversed; there must be exactly one nested filter or filter item, and the filter is true if the enclosed filter or filter item is false, and is false if the enclosed filter or filter item is true.

An instance of enumeration syntax *Filter Item Type* identifies the type of a filter item. Its value is chosen from one of the following: (a) *Equality* which means the filter item is true if the object contains at least one attribute of the specified type whose value is equal to that specified by Match Value (according to the equality matching rule in force), and false otherwise; (b) *Inequality* which means the filter item is true if the object contains at least one attribute of the specified type whose value is not equal to that specified by Match Value (according to the equality matching rule in force), and false otherwise; (c) *Greater*

or Equal which means the filter item is true if the object contains at least one attribute of the specified type whose value is equal to or greater than the value specified by Match Value (according to the matching rule in force), and false otherwise; (d) *Less or Equal* which means the filter item is true if the object contains at least one attribute of the specified type whose value is equal or less than the value specified by Match Value (according to the matching rule in force), and false otherwise; (e) *Greater* which means the filter item is true if the object contains at least one attribute of the specified type whose value is greater than the value specified by Match Value (according to the matching rule in force), and false otherwise; (f) *Less* which means the filter is true if the object contains at least one attribute of the specified type, whose value is less than the value specified by Match Value (according to the matching rule in force),and false otherwise; (g) *Present* which means the filter item is true if the object contains at least one attribute of the specified type, and false otherwise; (h) *Select* which means the filter item is true if the object contains at least one attribute of the specified type which has an object syntax and when the Filter is evaluated against the attributes of that object the Filter is true, and false otherwise; (i) *Request Candidates* which means the filter item is true if the object against which it is evaluated is one which could be used to provide some or all of the units requested by the specified License Request; the evaluation is made independently of any outstanding allocations or preallocations; and (j) *Simulate Request* which means the filter item is true if the object against which it is evaluated is one which would be used to provide some or all of the units requested by the specified License Request.

The Request Candidates and Simulate Request filter item types are of special use in testing and prototyping of systems by a license manager at a

licensee's site. For example, the license manager can simulate the effect of potential assignments, the effect of a population of certain types on a network, etc.

As an example, Figure 46 shows how a filter may be constructed to identify "All Product Use Authorizations issued by Digital for the Product 'Amazing Graphics System' which contains a calling authorization for Digital's 'Amazing Database' Product". This example is in the international standard format referred to as X.409 as mentioned above.

Filters can also used in a request allocation, being specified in a request extension as explained above. That is, a filter is one of the optional items in a request extension. For example, if a user wanted to use a version of WordPerfect with French language extension, and there were version with and without on the network, his request allocation would have a request extension that specified a filter for "French" in the token field. In this manner, a product can describe itself more richly. The filter in the request extension can be a Required filter or a Preferred filter, meaning the feature such as "French" is either absolutely necessary, or merely the preferred.

While this invention has been described with reference to specific embodiments, this description is not meant to be construed in a limiting sense. Various modifications of the disclosed embodiments, as well as other embodiments of the invention, will be apparent to persons skilled in the art upon reference to this description. It is therefore contemplated that the appended claims will cover any such modifications or embodiments as fall within the true scope of the invention.

WHAT IS CLAIMED IS:

1. A method of managing use of licensed software items, said software items separately executable on a computer system or accessible by said computer system, the computer system including a processor and one or more nodes, comprising the steps of:

maintaining by said processor a store of license authorizations for said software items; each license authorization including an indication of license management policy for a software item, said indication having a plurality of sets of policy components, said sets of policy components granting alternatives of specified restrictive rights to selectively access and execute said software items in said system; said indication of license management policy being in the format of an encoded document of a data type consisting of an ordered sequence of elements;

accessing said store by said processor to modify in said store one or more of said specified restrictive rights of said policy components of an identified license authorization;

accessing said store by said processor using a filter to obtain information from said license authorization for a selected software item, in response to a request from a node, and

comparing an identification of said node and said software item with said information, to produce and send to said node a grant or refusal of said request.

2. A method according to claim 1 including the step of receiving said license authorizations , for storing in said store,

from a license grantor external to said processor, and wherein said step of accessing said store to modify in said store one or more of said specified restrictive rights employs management functions executable on said processor but not on said nodes or said license grantor to identify a license authorization in said store.

3. A method according to claim 1 wherein said indication is in the format of an encoded document of a data type consisting of an ordered sequence of three elements, the three elements including a document descriptor, a document header and the document content.

4. A method according to claim 1 wherein said filter specifies one or more of said attributes and a Boolean operator for each selected attribute.

5. A method according to claim 2 wherein said step of accessing said store to modify one or more of said policy components is to allow grant of rights to use which are more restrictive than said specified restrictive rights.

6. A method according to claim 2 including the steps of:

sending a request by a user of one of said software items to obtain permission to use said software item; said request identifying the user and said software item;

accessing said store to obtain information from said license authorization for said software item, in response to said request, and comparing said identification of said user and said software item with said information, to produce a grant or refusal of said request for sending to said user.

7. A method according to claim 6 wherein said store is maintained by a license server, and said request is sent to said server and wherein said request is in the form of a remote procedure call, and said grant or refusal sent to said user is a return of said procedure call.

8. A method according to claim 7 wherein said license authorization is a data arrangement specified as a product use authorization, and said product use authorization is received by said server from an issuer, and wherein said server and said users are nodes on a computer network.

9. A method according to claim 2 wherein said policy components include a termination date, and said management functions can modify said termination date to an earlier termination date and wherein said policy components include a right of delegation of a right to grant said requests to another server, and said management functions can modify said right of delegation to remove said right of delegation.

10. A method according to claim 2 including storing in association with said license authorization a number of management attributes, and said management functions being able to modify said management attributes.

11. A method according to claim 10 wherein said management attributes include a reservation of units of license use granted by said license authorization so that said units will not be granted to a user in response to said request, and wherein said management attributes include an allocation of units of license use to a specific context.

12. A method according to claim 10 wherein said management attributes include an allocation period which is the minimum duration of an allocation of units, and wherein said management attributes include permission to enable a backup delegation of the right to grant said requests.

13. A system for managing use of licensed software products, comprising: means for maintaining a store of license documents, one for each said product; each license document including an indication of license policy having plurality of sets of policy components granting specified restrictive rights to use said software products, said policy components in each set providing alternatives;

a management interface for accessing said store to modify

selected ones of said components of an identified license authorization.

14. A system according to claim 13 including:

means for sending a request from a user of one of said products to obtain permission to use said product; said request identifying the user and said product;

means for accessing said store to obtain information from said license document for said product, in response to said request, and for comparing said identification of said user and said product with said information, and with constraints imposed by said policy components, to produce a grant or refusal of said request and send said grant or refusal to said user.

15. A system according to claim 13 wherein said management interface can modify said selected ones of said components to allow grant of rights to use which are more restrictive than said specified restrictive rights and wherein said means for maintaining, and said means for accessing and sending to said user are all located at a server on a distributed network, and said means for sending a request is located at a user node on said network.

16. A system according to claim 14 wherein said request is in the form of a remote procedure call, and said grant or refusal sent to said user is a return of said procedure call, and wherein said

license document is a data arrangement specified as a product use authorization, and said product use authorization is received by said server from a license issuer.

17. A system according to claim 13 wherein said policy components include a termination date, and said management functions can modify said termination date to an earlier termination date, and wherein said policy components include a right of delegation of a right to grant said requests to another server, and said management functions can modify said right of delegation to remove said right of delegation.

18. A system according to claim 15 including means for storing in association with said license authorization a number of management attributes, wherein said management functions are able to modify said management attributes and wherein said management attributes include a reservation of units of license use granted by said license authorization so that said units will not be granted to a user in response to said request.

19. A system according to claim 18 wherein said management attributes include an allocation of units of license use to a specific context.

20. A system according to claim 18 wherein said management attributes include an allocation period which is the minimum

duration of an allocation of units, and include permission to enable a backup delegation of the right to grant said requests.

21. A method according to claim 3 wherein said document descriptor includes an encoding method version number, and encoder-identifier and an encoder-name, and wherein said document-header includes a title, an author, a version and a date for the software item.

22. A method according to claim 3 wherein said document content includes at least one of the following:

a product-use-authorization;

a license-use-requirements-table;

a group-definition;

a key-registration;

a delegation.

23. A method according to claim 3 wherein said document-content includes a license-data-header, and said license-data-header describes the parties to the license document, the term of the agreement and constraints that may have been placed on management of the license data.

24. A method according to claim 3 wherein said document-content includes management-info, where the management-info may include at least one of the following:

an assignment;

a reservation;

a delegation;

a backup delegation;

an allocation;

a registration date;

a registrar;

a comment;

a termination-date.

25. A method according to claim 3 wherein:

said document descriptor includes an encoding method version and a date for the software item;

said document content may include at least one of the following: a product-use-authorization, a license-use-requirements-table, a group-defination, a key-registration, and a delegation;

said document-content selectively includes a license-data-header, and said license-data-header describes the parties to the license document, the term of the agreement and constraints that may have been placed on management of the license data;

said document-content may have been placed on management of the license data;

said document-content selectively includes management-info, where the management-info may include at least one of the following: an assignment, a reservation, a delegation, a backup delegation, an allocation, a registration date, a registrar, and a

comment.

26. A method according to claim 3 wherein said store is maintained by a license server, and said request is sent to said server, and wherein said server and said users are nodes on a computer network.

27. A method according to claim 3 wherein said request is in the form of a remote procedure call, and said grant or refusal sent to said user is a return of said procedure call, and wherein said license authorization is received by said server from an issuer.

28. A method according to claim 3 including the steps of: sending a request by a user of one of said software items to obtain permission to use said software item; said request identifying the user and said software item;

sending said grant or refusal to said user.

29. Apparatus for managing use of licensed software items, comprising:

means for maintaining a store of license authorizations for said software items; each license authorization including an indication of license management policy for a software item, said indication being in the format of an encoded document of a data type consisting of an ordered sequence of three elements, the three elements including a document descriptor, a document header and the

document content;

means for sending a request by a user of one of said software items to obtain permission to use said software item; said request identifying the user and said software item;

means for accessing said store to obtain information from said license authorization for said software item, in response to said request, and comparing said identification of said user and said software item with said information, to produce a grant or refusal of said request;

means for sending said grant or refusal to said user.

30. Apparatus according to claim 29 wherein said document descriptor includes an encoding method version number, and an encoder-identifier and an encoder-name, and wherein said document-header includes a title, an author, a version and a date for the software item.

31. Apparatus according to claim 29 wherein said document content includes at least one of the following:

a product-use-authorization;

a license-use-requirements-table;

a group-definition;

a key-registration;

a delegation.

32. Apparatus according to claim 29 wherein said document-content includes a license-data-header, and said license-data-header describes the parties to the license document, the term of the agreement and constraints that may have been placed on management of the license data.

33. Apparatus according to claim 29 wherein said document-content includes management-info, where the management-info may include at least one of the following:

an assignment;

a reservation;

a delegation;

a backup delegation;

an allocation;

a registration date;

a registrar;

a comment;

a termination-date.

34. Apparatus according to claim 29 wherein:

said document descriptor includes an encoding method version number, and encoder-identifier and an encoder-name;

said document-header includes a title, an author, a version and a date for the software item;

said document content may include at least one of the following: a product-use-authorization, a license-use-requirements-

table, a group-definition, a key-registration, and a delegation;

said document-content may include a license-data-header, and said license-data-header describes the parties to the license document, the term of the agreement and constraints that may have been placed on management of the license data;

said document-content may include management-info, where the management-info may include at least one of the following: an assignment, a reservation, a delegation, a backup delegation, an allocation, a registration date, a registrar, and a comment.

35. Apparatus according to claim 29 wherein said store is maintained by a license server, and said request is sent to said server, and wherein said request is in the form of a remote procedure call, and said grant or refusal sent to said user is a return of said procedure call.

36. Apparatus according to claim 29 wherein said license authorization is received by said server from an issuer, and wherein said server and said users are nodes on a computer network.

37. A method of storing license documents by a server for a license management system, comprising the steps of:

maintaining a store of license documents for software items; each license document including an indication of license management policy for a software item, said indication being in the format of an encoded document of a data type consisting of an ordered

sequence of three elements, the three elements including a document descriptor, a document header and the document content;

accessing said store to obtain information from a selected one of said license documents for a software item, in response to a request, and referencing said indication of license management policy, to produce a grant or refusal of said request.

38. A method according to claim 37 wherein said document descriptor includes an encoding method version number, an encoder-identifier and an encoder-name, and wherein said document-header includes a title, an author, a version and a date for the software item.

39. A method according to claim 37 wherein said document content includes at least one of the following:

a product-use-authorization;

a license-use-requirements-table;

a group-definition;

a key-registration;

a delegation.

40. A method according to claim 4 wherein said step of selecting by a filter may select on one or more of the attributes: issuer, producer, product name, product use authorization, calling authorization, and wherein said store is maintained by a license server, and said request is sent to said server.

41. A method according to claim 4 wherein said request is in the form of a remote procedure call, and said grant or refusal sent

to said user is a return of said procedure call.

42. A method according to claim 40 wherein said license authorization is a data arrangement specified as a product use authorization, and said product use authorization is received by said server from an issuer, and wherein said server and said users are nodes on a computer network.

43. Apparatus for managing use of licensed software items, comprising:

means for maintaining a store of license authorizations for said software items; each license authorization including an indication of license management policy for a software item, said indication being an encoded document containing a number of attributes defining said license policy;

filter means for selecting from said store, said filter means specifying one or more of said attributes and a Boolean operator for each selected attribute;

means for sending a request by a user of one of said software items to obtain permission to use said software item; said request identifying the user and said software item;

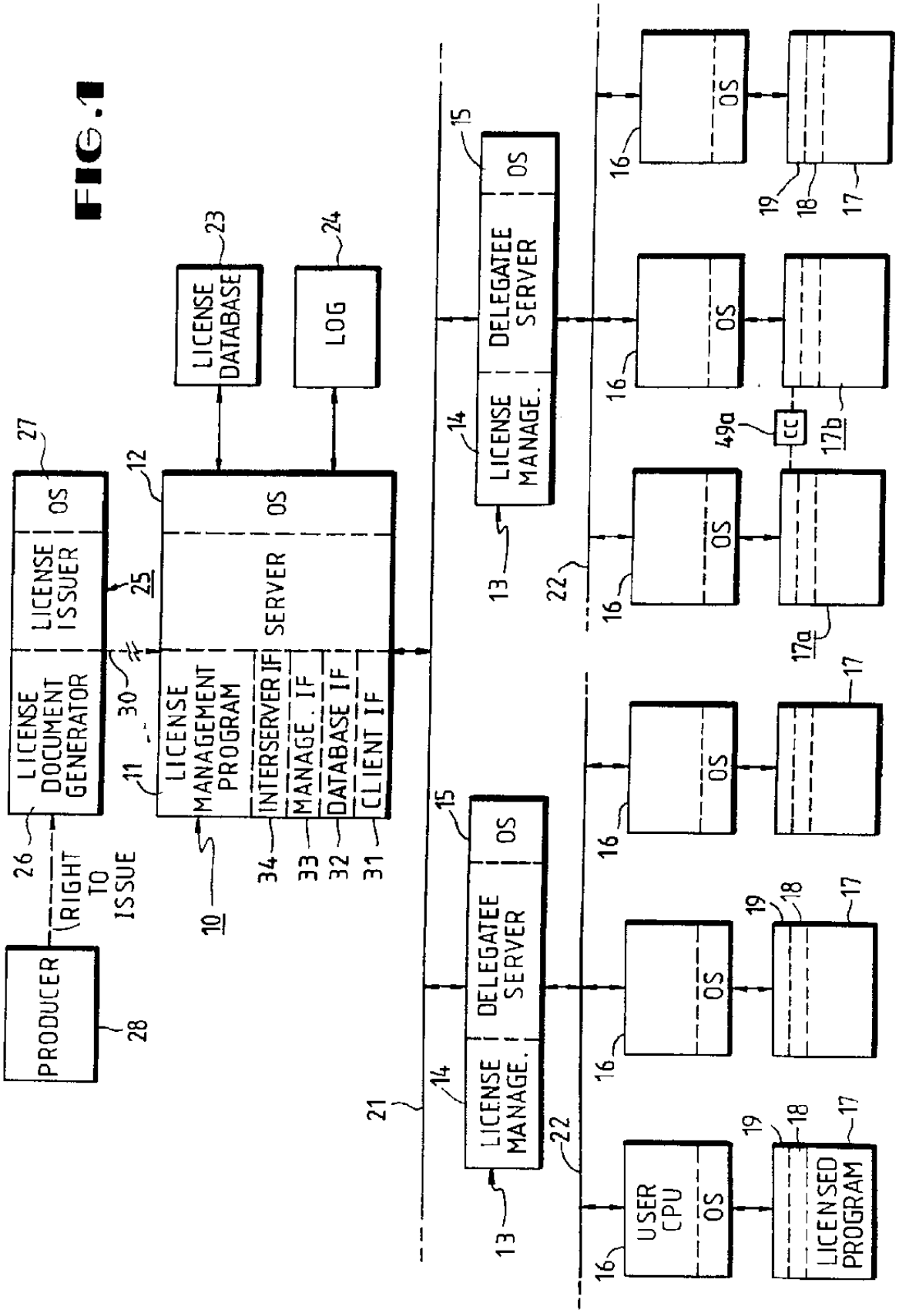
means for accessing said store to obtain information from said license authorization for said software item, in response to said request, and comparing said identification of said user and said software item with said information, to produce a grant or refusal of said request; and

means for sending said grant or refusal to said user.

44. Apparatus according to claim 43 wherein said filter means may select on one or more of the attributes: issuer, producer, product name, product use authorization, calling authorization, and wherein said store is maintained by a license server, and said request is sent to said server, and wherein said request is in the form of a remote procedure call, and said grant or refusal sent to said user is a return of said procedure call.

45. Apparatus according to claim 43 wherein said license authorization is a data arrangement specified as a product use authorization, and said product use authorization is received by said server from an issuer, wherein said server and said users are nodes on a computer network.

FIG.1



SUBSTITUTE SHEET

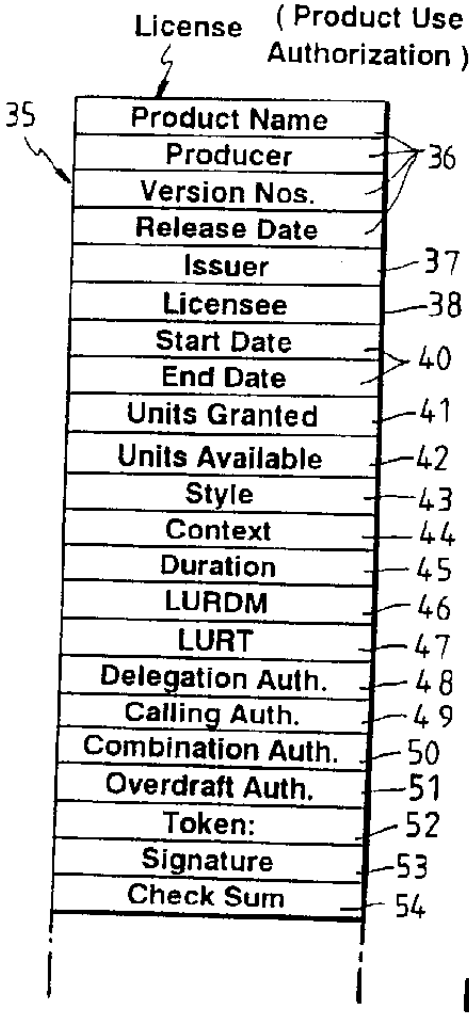


FIG.2

License Unit Requirements Table			
Row Selector	Columns		
Platform ID	A	B	C
PC-0	10	230	-1
PC-1	12	230	-1
VAX 6210	158	300	150

FIG.4

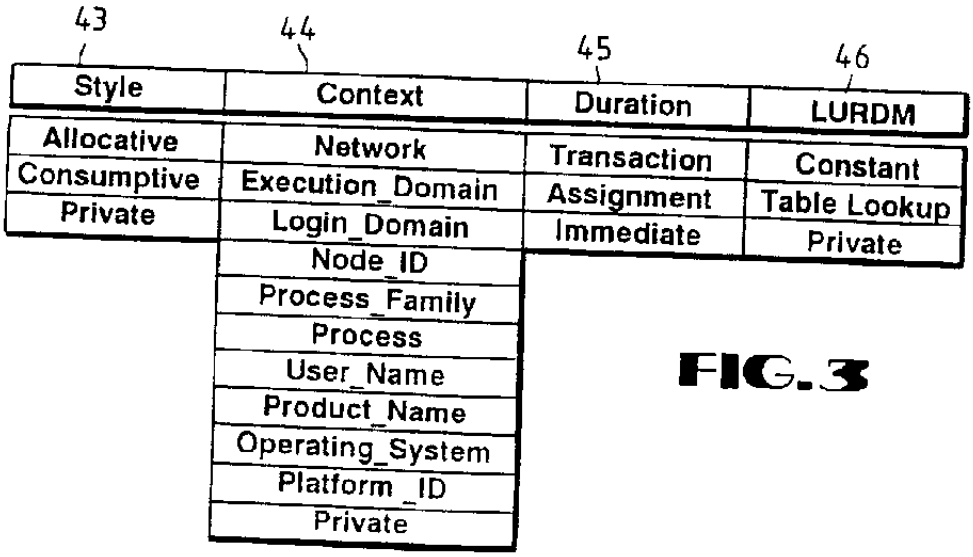


FIG.3

APPLE EXHIBIT 1102, Page 357 of 1048

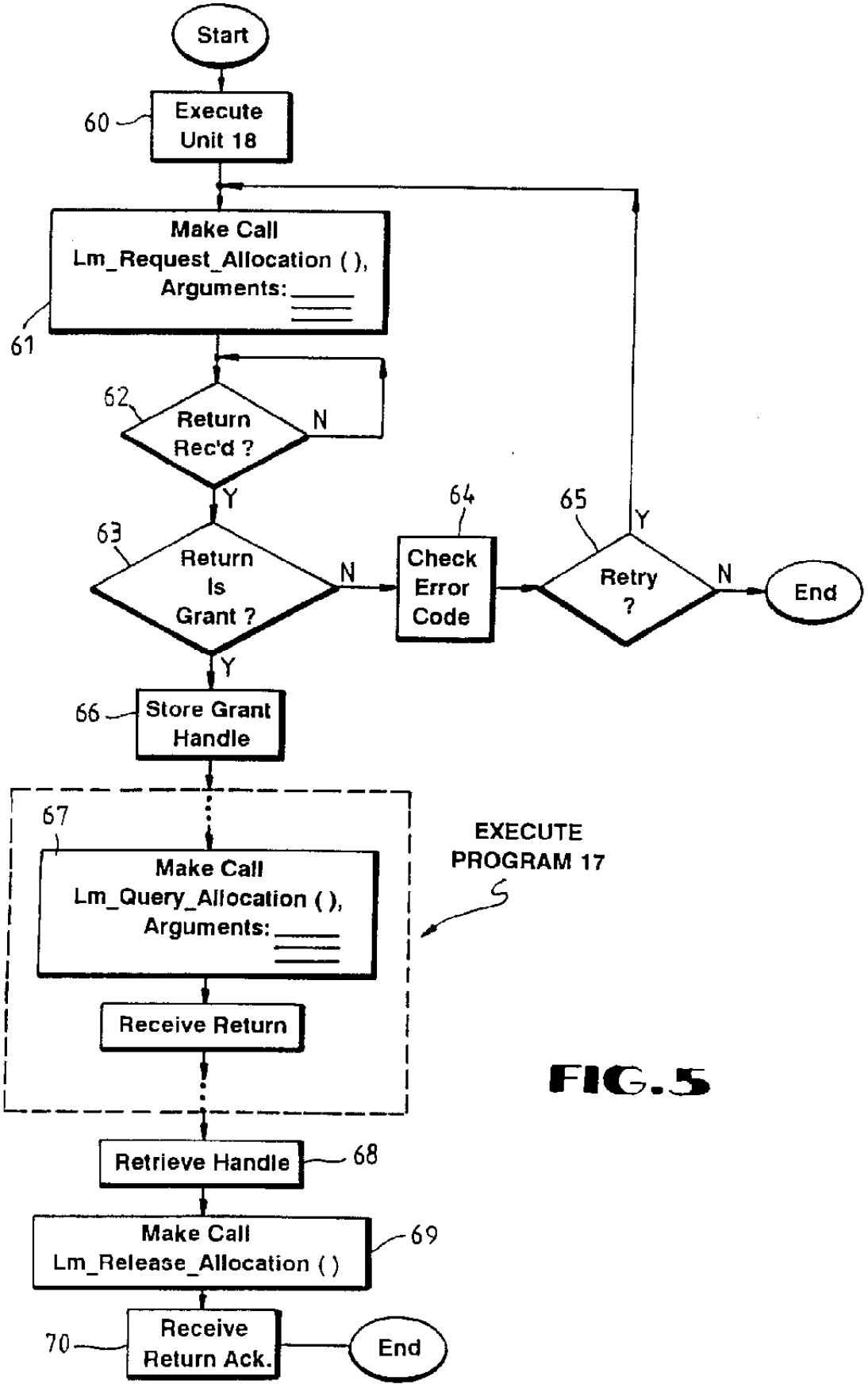

Start
60
Execute Unit 18
Make Call Lm_Request_Allocation ( ), Arguments:
61
62
Return Rec'd ?
N
Y
63
Return Is Grant ?
N
64
Check Error Code
65
Retry ?
Y
N
End
Y
66
Store Grant Handle
67
Make Call Lm_Query_Allocation ( ), Arguments:
Receive Return
EXECUTE PROGRAM 17
5
Retrieve Handle
68
Make Call Lm_Release_Allocation ( )
69
70
Receive Return Ack.
End

FIG.5

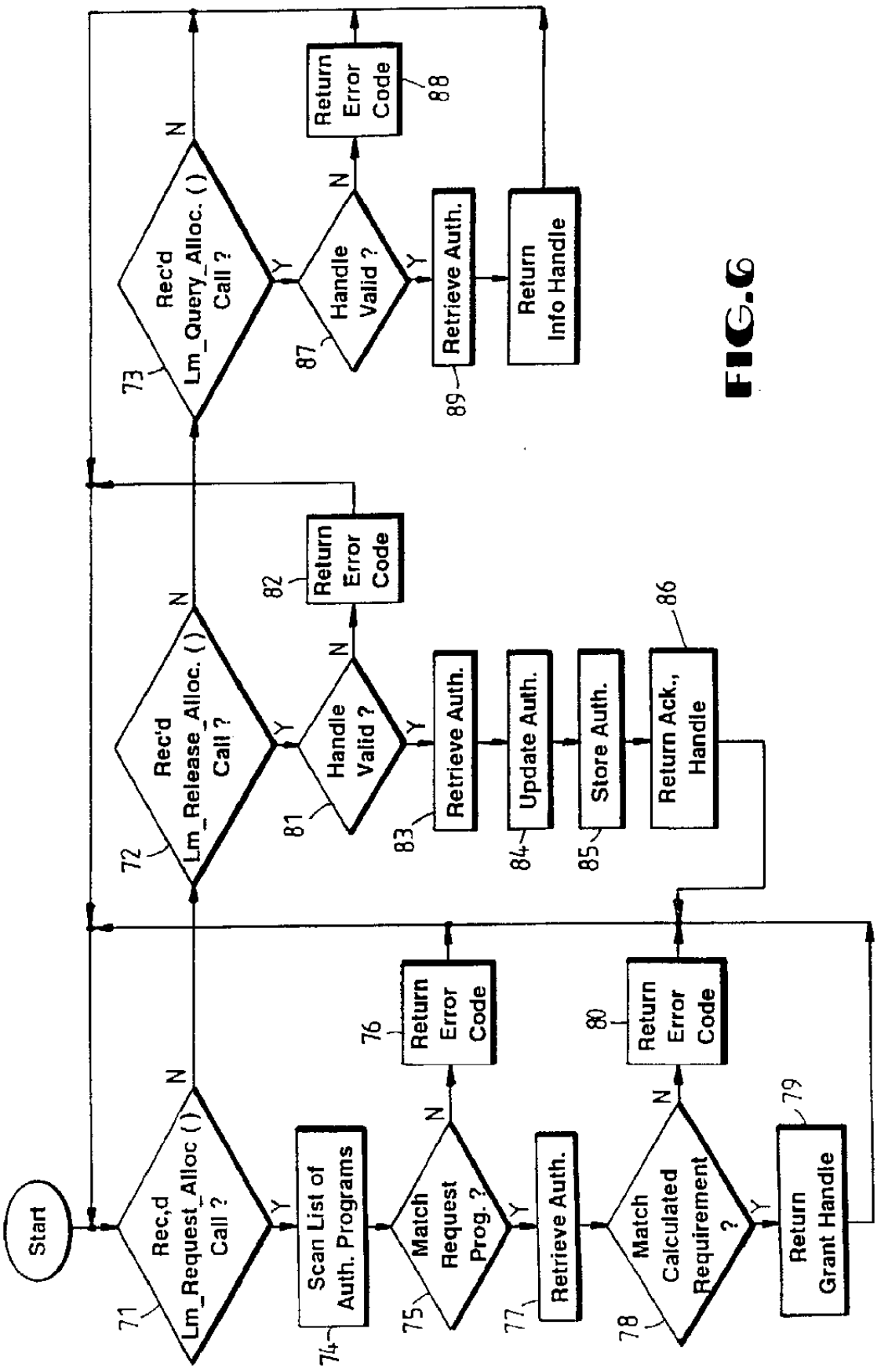

Start
Rec,d Lm_Request_Alloc ( ) Call ?
71
N
Y
Scan List of Auth. Programs
74
Match Request Prog. ?
75
N
Y
Return Error Code
76
Retrieve Auth.
77
Match Calculated Requirement ?
78
N
Y
Return Error Code
80
Return Grant Handle
79
Rec'd Lm_Release_Alloc. ( ) Call ?
72
N
Y
Handle Valid ?
81
N
Y
Return Error Code
82
Retrieve Auth.
83
Update Auth.
84
Store Auth.
85
Return Ack., Handle
86
Rec'd Lm_Query_Alloc. ( ) Call ?
73
N
Y
Handle Valid ?
87
N
Y
Return Error Code
88
Retrieve Auth.
89
Return Info Handle

FIG.6

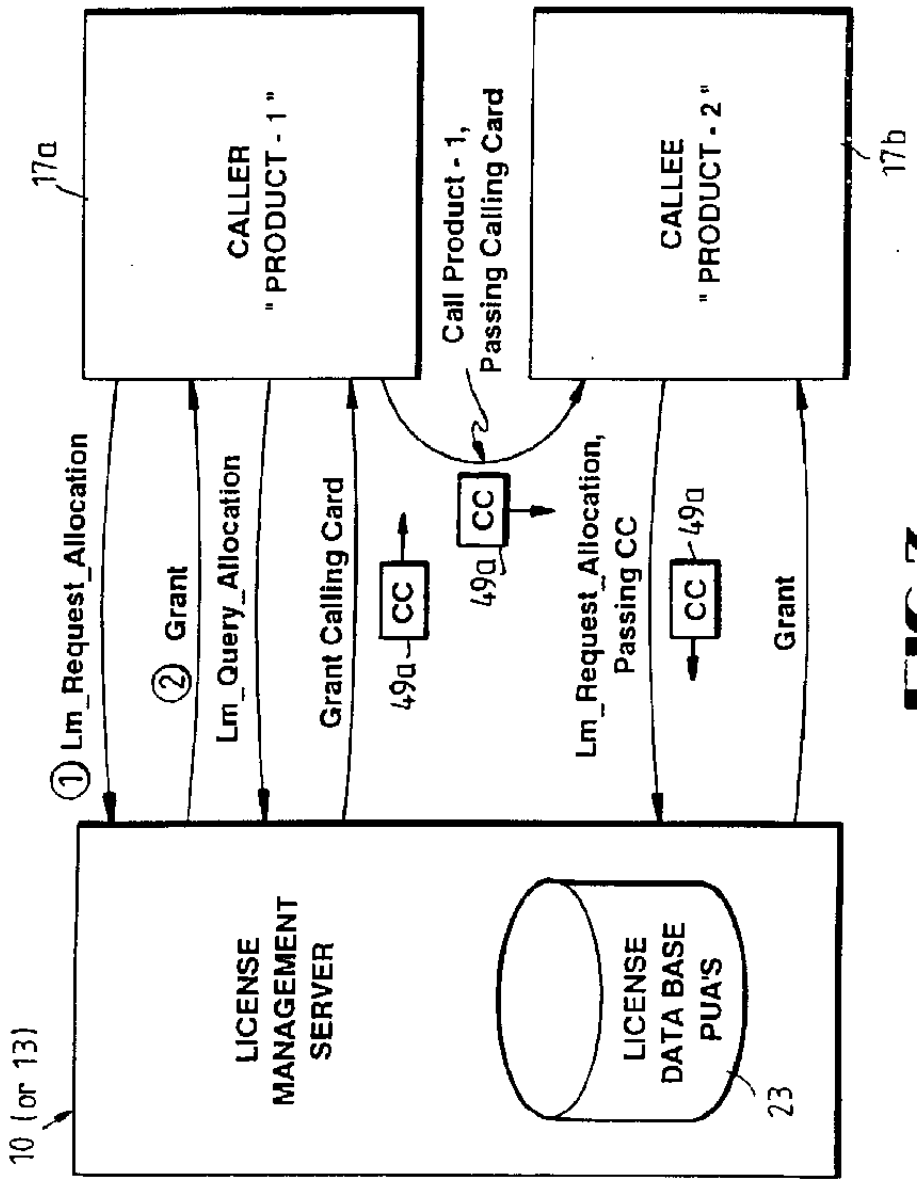

17a
CALLER
" PRODUCT - 1 "
17b
CALLEE
" PRODUCT - 2 "
Call Product - 1,
Passing Calling Card
① Lm_Request_Allocation
② Grant
Lm_Query_Allocation
Grant Calling Card
CC
49a
CC
49a
Lm_Request_Allocation,
Passing CC
CC
49a
Grant
10 (or 13)
LICENSE
MANAGEMENT
SERVER
LICENSE
DATA BASE
PUA'S
23

FIG.7

```
Object Identifier Value : : =    {
                                     iso(1)
                                     identified-organization(3)
                                     icd-ecma(12)
                                     member-company(2)
                                     dec(1011)
                                     data-syntaxes(1)
                                     cda(3)
                                     ldif(17)
                                 }
Object Identifier Encoding : : = {
                                     0x6, 0x8, 0x2B, 0xC, 0x2,
                                     0x87, 0x73, 0x1, 0x3, 0x11
                                 }
```

FIG. 8 LDIF Object Identifier

APPLE EXHIBIT 1102, Page 361 of 1048

```
LDIFDocument        ::=    [PRIVATE 16373] IMPLICIT SEQUENCE {
  document-descriptor          [0] IMPLICIT DocumentDescriptor OPTIONAL,
  document-header              [1] IMPLICIT DocumentHeader OPTIONAL,
  document-content             [2] IMPLICIT DocumentContent
                               }
```

FIG. 9 LDIF Document Syntax Diagram

FIG. 10 LDIF Document Structure

```
DocumentDescriptor      ::=     SEQUENCE {
    major-version                   [0] IMPLICIT INTEGER OPTIONAL,
    minor-version                   [1] IMPLICIT INTEGER OPTIONAL,
    encoder-identifier              [2] IMPLICIT Character-String OPTIONAL,
    encoder-name                    [3] IMPLICIT Character-String OPTIONAL
                            }
```

FIG. 11 Document Descriptor Syntax Diagram

```
Pakgen DocumentDescriptor  ::= {
    major-version 1,
    minor-version 0,
    encoder-identifier "PAKGEN",
    encoder-name {Character-String "PAK Generator V1.0"}
    }
```

FIG. 12 Document Descriptor Example

```
DocumentHeader      ::=     SEQUENCE {
  private-header-data           [0] IMPLICIT NamedValueList OPTIONAL,
  title                         [1] IMPLICIT Character-String OPTIONAL,
  author                        [2] IMPLICIT Character-String OPTIONAL,
  version                       [3] IMPLICIT Character-String OPTIONAL,
  date                          [4] IMPLICIT UTCTime OPTIONAL
                        }
```

FIG. 13 Document Header Syntax Diagram

```
example-header document-header ::= {
    title {Character-String "PAKGEN Licenses with Associated LURT data"}
    author {Character-String "Tom Jones, Foobar, Inc. License Department"}
    version {Character-String "V0.1"}
    date "19880102110O-0500"
    }
```

FIG. 14 Document Header Example

```
Document Content        ::= SEQUENCE OF LicenseData

LicenseData             ::= SEQUENCE {
  license-data-header           [0] IMPLICIT LicenseDataHeader,
  license-body                  [1] CHOICE {
    product-use-authorization       [0] IMPLICIT ProductUseAuthorization,
    license-units-requirements-table [1] IMPLICIT LURT,
    group-definition                [2] IMPLICIT GroupDefinition,
    key-registration                [3] IMPLICIT KeyRegistration,
    issuer-delegation               [4] IMPLICIT IssuerDelegation,
    license-delegation              [5] IMPLICIT LicenseDelegation,
    backup-delegation               [6] IMPLICIT BackupDelegation
                                },
  management-info               [2] IMPLICIT ManagementInfo OPTIONAL
                            }
```

FIG. 15 Document Content Syntax Diagram

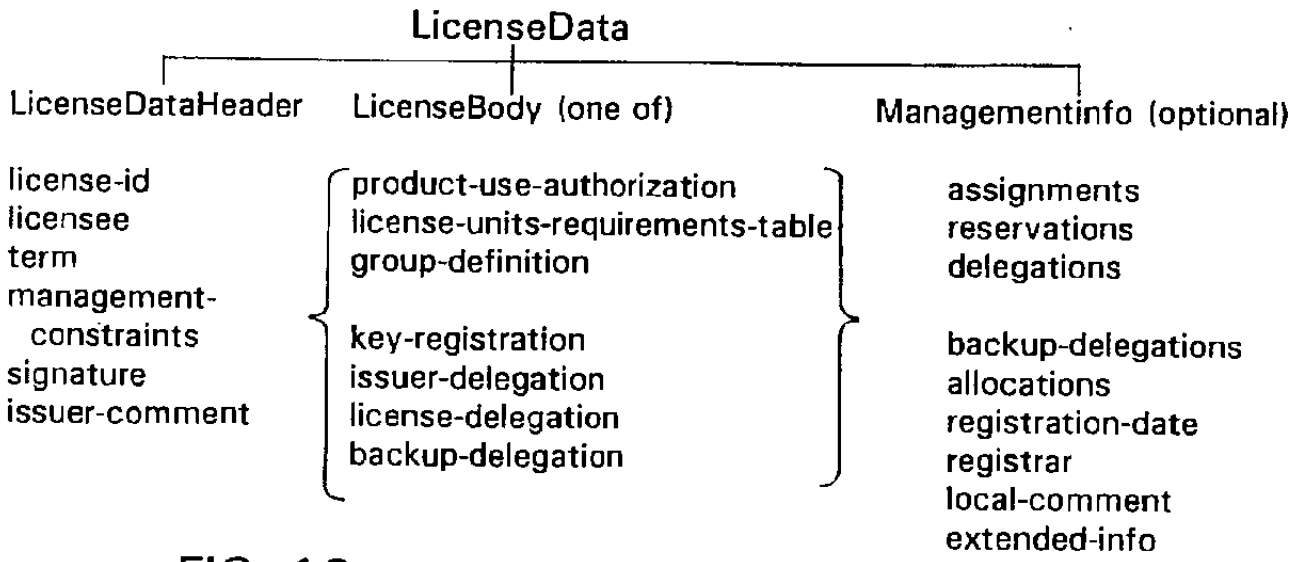


FIG. 16 License Data Structure

```
LicenseDataHeader          ::= SEQUENCE {
  license-id                     [0] IMPLICIT LicenseID,
  licensee                       [1] IMPLICIT Character-String OPTIONAL,
  term                           [2] IMPLICIT Term OPTIONAL,
  management-constraints         [4] IMPLICIT ManagementConstraints OPTIONAL,
  signature                      [5] IMPLICIT Signature,
  issuer-comment                 [6] IMPLICIT NamedValueList OPTIONAL
```

FIG. 17 License Data Header Syntax Diagram

SUBSTITUTE SHEET

APPLE EXHIBIT 1102, Page 366 of 1048

```
ProductUseAuthorization    ::= SEQUENCE {
    product-id                    [0] IMPLICIT ProductID,
    units-granted                 [1] IMPLICIT INTEGER,
    management-policy             [2] IMPLICIT ManagementPolicy,
    calling-authorizations        [3] IMPLICIT SEQUENCE OF Member OPTIONAL,
    caller-authorizations         [4] IMPLICIT SEQUENCE OF Member OPTIONAL,
    execution-constraints         [5] IMPLICIT ExecutionConstraints OPTIONAL,
    product-token                 [6] IMPLICIT NamedValueList OPTIONAL
                               }
```

FIG. 18 Product Use Authorization Syntax Diagram


```
LURT                       ::= SEQUENCE {
    lurt-name                     [0] IMPLICIT Character-String,
    rows                          [1] IMPLICIT RowList
                               }
RowList                    ::= SEQUENCE OF LurtRow

LurtRow                    ::= SEQUENCE {
    platform-id                   [0] IMPLICIT Character-String,
    lurt-columns                  [1] IMPLICIT SEQUENCE OF INTEGER
                               }
```

FIG. 19 License Unit Requirement Table Syntax Diagram

SUBSTITUTE SHEET

```
Example LURT ::= {
    lurt-name { Character-String "Example LURT"}
    rows {
        LurtRow {
            {Character-String "PC-0"}
            {{10} {230} {-1}}
        LurtRow {
            {Character-String "PC-1"}
            {{12} {230} {-1}}
        LurtRow {
            {Character-String "VAX 6210"}
            {{158} {300} {150}}
    }
}
```

FIG. 20 Example Encoding of LURT

```
Group Definition    ::= SEQUENCE {
    group-name              [0] IMPLICIT Character-String,
    group-version           [1] IMPLICIT Version,
    group-release-date      [2] IMPLICIT UTCTime,
    group-members           [3] IMPLICIT SEQUENCE OF Member
                        }
```

FIG. 21 Group Definition Syntax Diagram

```
KeyRegistration     ::= SEQUENCE {
    key-owner-name          [0] IMPLICIT Character-String,
    key-algorithm           [1] IMPLICIT Character-String,
    key-value               [2] IMPLICIT OCTET STRING
                        }
```

FIG. 22 Key Registration Syntax Diagram

```
IssuerDelegation        ::= SEQUENCE {
  delegated-issuer-name       [0] IMPLICIT Character-String,
  delegated-product-id        [1] IMPLICIT SEQUENCE OF Member,
  delegated-units-granted     [2] IMPLICIT INTEGER OPTIONAL,
  template-authorization      [3] IMPLICIT ProductUseAuthorization OPTIONAL,
  sub-license-permitted       [4] IMPLICIT BOOLEAN DEFAULT FALSE
                          }
```

FIG. 23 Issuer Delegation Syntax Diagram

```
LicenseDelegation       ::= SEQUENCE {
  delegated-units             [0] IMPLICIT INTEGER OPTIONAL
  delegated-distribution-control  [1] IMPLICIT DistributionControl,
  delegatee-execution-constraints [2] IMPLICIT ExecutionConstraints OPTIONAL,
  assignment-list             [3] IMPLICIT AssignmentList OPTIONAL,
  delegated-data              [4] IMPLICIT LicenseData OPTIONAL
                          }
```

FIG. 24 License Delegation & Backup Delegation Syntax Diagrams

```
ManagementInfo      ::= SEQUENCE {
    assignments             [0] IMPLICIT AssignmentList OPTIONAL,
    reservations            [1] IMPLICIT AssignmentList OPTIONAL,
    delegations             [2] IMPLICIT DelegationList OPTIONAL,
    backup-delegations      [3] IMPLICIT DelegationList OPTIONAL,
    allocations             [4] IMPLICIT AllocationList OPTIONAL,
    registration-date       [5] IMPLICIT UTCTime,
    registrar               [6] IMPLICIT Context,
    local-comment           [7] IMPLICIT NamedValueList OPTIONAL,
    termination-date        [8] IMPLICIT UTCTime OPTIONAL,
    extended-info           [9] IMPLICIT NamedValueList OPTIONAL
                        }
```

FIG. 25 ManagementInfo Syntax Diagram

```
AllocationList      ::= SEQUENCE OF Allocation

Allocation          ::= SEQUENCE {
    allocation-context      [0] IMPLICIT Context,
    allocation-lur          [1] IMPLICIT INTEGER,
    allocation-group-id     [2] IMPLICIT INTEGER OPTIONAL
                        }
```

FIG. 26 Allocation Syntax Diagram

```
AssignmentList      ::= SEQUENCE OF Assignment

Assignment          ::= SEQUENCE {
    assigned-units          [0] IMPLICIT INTEGER,
    assignment-term         [1] IMPLICIT Term,
    assignee                [2] IMPLICIT Context
                        }
```

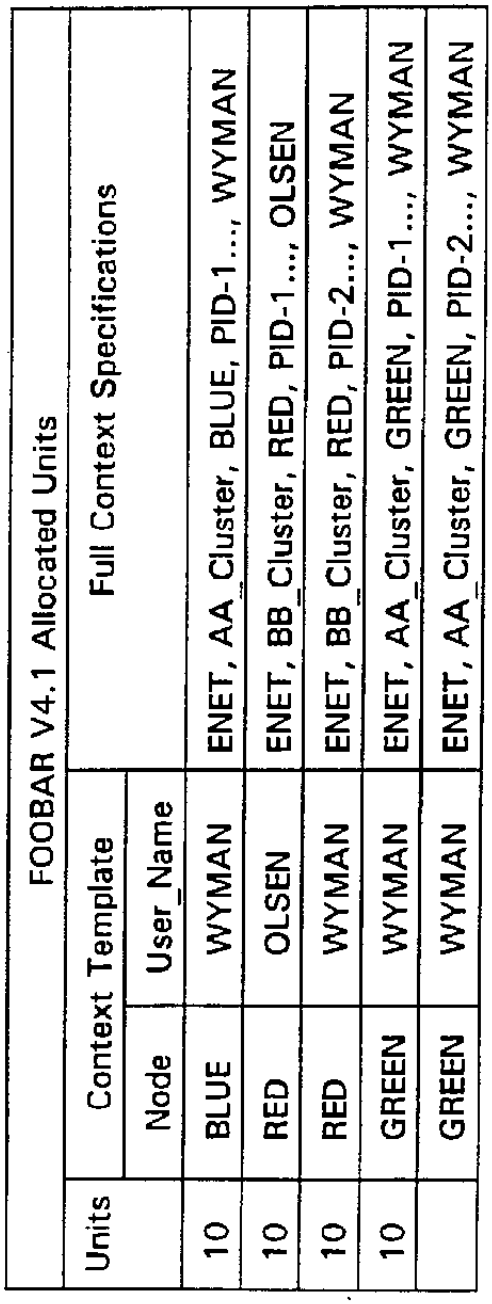
FIG. 27 Assignment Syntax Diagram

```
ContextList          ::= SEQUENCE OF Context

Context              ::= SEQUENCE OF Subcontext

SubContext           ::= SEQUENCE {
    sub-context-type            [0] SubContextType,
    subcontext-value            [1] ValueData
                        }
SubContextType       ::= CHOICE {
    standard-subcontext-type    [0] IMPLICIT INTEGER {
        network-subcontext(1),
        execution-domain-subcontext(2),
        login-domain-subcontext(3),
        node-subcontext(4),
        process-family-subcontext(5),
        process-id-subcontext(6),
        user-name-subcontext(7),
        product-name-subcontext(8),
        operating-system-subcontext(9),
        platform-id-subcontext(10)
                        }
    private-subcontext          [1] IMPLICIT INTEGER (first(0),last(255))
                        }
```

FIG. 28 Context Syntax Diagram



FOOBAR V4.1 Allocated Units			
Units	Context Template		Full Context Specifications
	Node	User_Name	
10	BLUE	WYMAN	ENET, AA_Cluster, BLUE, PID-1..., WYMAN
10	RED	OLSEN	ENET, BB_Cluster, RED, PID-1..., OLSEN
10	RED	WYMAN	ENET, BB_Cluster, RED, PID-2..., WYMAN
10	GREEN	WYMAN	ENET, AA_Cluster, GREEN, PID-1..., WYMAN
	GREEN	WYMAN	ENET, AA_Cluster, GREEN, PID-2..., WYMAN

FIG. 29 Only unique contexts require explicit unit allocations.

FOOBAR V4.1 Allocated Units		
Units	Context Template Node	Full Context Specifications
10	BLUE	ENET, AA_Cluster, BLUE, PID-1..., WYMAN
10	RED	ENET, BB_Cluster, RED, PID-1..., OLSEN
	RED	ENET, BB_Cluster, RED, PID-2..., WYMAN
10	GREEN	ENET, AA_Cluster, GREEN, PID-1..., WYMAN
	GREEN	ENET, AA_Cluster, GREEN, PID-2..., WYMAN

FIG. 30 Modification of Context_Template impacts units requirements.

```
DistributionControl          ::= SEQUENCE {
    distribution-method              [0] IMPLICIT INTEGER {
        refresh-distribution(1),
        initial-distribution-only(2),
        manual-distribution(3)
                                     },
    current-start-date               [1] IMPLICIT UTCTime OPTIONAL
    current-end-date                 [2] IMPLICIT UTCTime OPTIONAL,
    refresh-interval                 [3] IMPLICIT IntervalTime OPTIONAL,
    retry-interval                   [4] IMPLICIT IntervalTime OPTIONAL,
    maximum-retry-count              [5] IMPLICIT INTEGER OPTIONAL,
    retries-attempted                [6] IMPLICIT INTEGER OPTIONAL
                                 }
```

FIG. 31 Distribution Control Syntax Diagram

APPLE EXHIBIT 1102, Page 376 of 1048

```
ExecutionConstraints ::= SEQUENCE {
    operating-system     [0] IMPLICIT SEQUENCE OF Character-String OPTIONAL,
    execution-context    [1] IMPLICIT ContextList OPTIONAL,
    environment-list     [2] IMPLICIT SEQUENCE OF EnvironmentKind OPTIONAL
                         }
EnvironmentKind      ::= INTEGER {
        batch(1),
        interactive(2),
        local(3),
        network(4),
        remote(5)
                         }
```

FIG. 32 Execution Constraints Syntax Diagram

```
LicenseID        ::= SEQUENCE {
    issuer                [0] IMPLICIT Character-String,
    serial-number         [1] IMPLICIT Character-String,
    amendment             [2] IMPLICIT INTEGER DEFAULT 0
                     }
```

FIG. 33 License ID Syntax Diagram

```
LURDM                ::= SEQUENCE {
    combination-permitted          [0] IMPLICIT BOOLEAN DEFAULT TRUE,
    overdraft-limit                [1] IMPLICIT INTEGER DEFAULT 0,
    overdraft-logging-required     [2] IMPLICIT BOOLEAN DEFAULT FALSE,
    allocation-size                [3] IMPLICIT INTEGER OPTIONAL,
    lurdm-kind                     [4] IMPLICIT INTEGER {
        lurt(1),
        constant(2),
        private-lurdm(3)
                                       },
    named-lurt-id                  [5] IMPLICIT Character-String OPTIONAL,
    lurdm-value                    [6] IMPLICIT INTEGER OPTIONAL,
    default-unit-requirement       [7] IMPLICIT INTEGER OPTIONAL
                                   }
```

FIG. 34 License Unit Requirements Determination Method Syntax Diagram

```
ManagementConstraints    ::= SEQUENCE {
    management-context          [0] IMPLICIT ContextList OPTIONAL,
    management-scope            [1] IMPLICIT INTEGER {
        single-platform(1),
        management-domain(2),
        entire-network(3)
                                } OPTIONAL,
    backup-permitted            [2] IMPLICIT BOOLEAN DEFAULT TRUE,
    delegation-permitted        [3] IMPLICIT BOOLEAN DEFAULT TRUE,
    maximum-delegation-period   [4] IMPLICIT IntervalTime OPTIONAL
                                }
```

FIG. 35 Management Constraints Syntax Diagram

```
ManagementPolicy    ::= SEQUENCE {
  style                    [0] IMPLICIT INTEGER {
      allocative(1),
      consumptive(2),
      private-style(3)
                           },
  context-template         [1] IMPLICIT SEQUENCE OF SubcontextType
                               OPTIONAL,
  duration                 [2] IMPLICIT INTEGER {
      transaction(1),
      assignment(2),
      immediate(3)
                           } OPTIONAL,
  lur-determination-method [3] IMPLICIT LURDM OPTIONAL,
  allocation-sharing-limit [4] IMPLICIT INTEGER OPTIONAL,
  reassignment-constraint  [5] IMPLICIT IntervalTime OPTIONAL
                           }
```

FIG. 36 Management Policy Syntax Diagram

```
Member                  ::= SEQUENCE {
    member-product              [0] IMPLICIT ProductID,
    member-signature            [1] IMPLICIT Signature,
    member-token                [2] IMPLICIT NamedValueList OPTIONAL
                        }
```

FIG. 37 Member Syntax Diagram

```
NamedValue              ::= SEQUENCE {
    value-name              Character-String,
    value-data              ValueData
                        }

ValueData               ::= CHOICE {
    value-boolean               [0] IMPLICIT BOOLEAN,
    value-integer               [1] IMPLICIT INTEGER,
    value-text                  [2] IMPLICIT SEQUENCE OF Character-String
    value-general               [3] IMPLICIT OCTET STRING,
    value-list                  [4] IMPLICIT SEQUENCE OF ValueData
                        }

NamedValueList          ::= SEQUENCE OF NamedValue
```

FIG. 38 Named Value, Value Data & Named Value List Syntax Diagrams

SUBSTITUTE SHEET APPLE EXHIBIT 1102, Page 382 of 1048

```
ExampleList NamedValueList ::= {
    NamedValue {
        value-name {Character-String "Purchase Order"}
        value-data {INTEGER 154493}
    }
    NamedValue {
        value-name {Character-String "Telephone Support #"}
        value-data {Character-String {+1 (999) 555-1234}
    }
}
```

FIG. 39 Named Value List Example

```
ProductID           ::= SEQUENCE {
    producer                [0] IMPLICIT Character-String,
    product-name            [1] IMPLICIT Character-String,
    first-version           [2] IMPLICIT Version OPTIONAL,
    last-version            [3] IMPLICIT Version OPTIONAL,
    first-release-date      [4] IMPLICIT UTCTime OPTIONAL,
    last-release-date       [5] IMPLICIT UTCTime OPTIONAL
                        }
```

FIG. 40 Product ID Syntax Diagram

SUBSTITUTE SHEET
APPLE EXHIBIT 1102, Page 383 of 1048

```
Signature        ::= SEQUENCE {
    signature-algorithm      [0] IMPLICIT Character-String,
    signature-parameters     [1] IMPLICIT NamedValueList OPTIONAL,
    signature-value          [2] IMPLICIT OCTET STRING
                             }
```

FIG. 41 Signature Syntax Diagram

```
Term             ::= SEQUENCE {
    start-date               [0] IMPLICIT UTCTime OPTIONAL,
    end-date                 [1] IMPLICIT UTCTime OPTIONAL,
                             }
```

FIG. 42 Term Syntax Diagram

```
Version          ::= SEQUENCE {
    part-1                [0] IMPLICIT INTEGER,
    part-2                [1] IMPLICIT INTEGER DEFAULT 0,
    part-3                [2] IMPLICIT INTEGER DEFAULT 0,
    part-4                [3] IMPLICIT INTEGER DEFAULT 0
                      }
```

FIG. 43

Attributes Specific to Filter				
Attribute	Value Syntax	Value Length	Value Number	Value Initially
Filter Items	Object(Filter Item)	-	0 or more	-
Filters	Object(Filter)	-	0 or more	-
Filter Type	Enum(Filter Type)	-	1	-

FIG. 44

Attributes Specific to Filter				
Attribute	Value Syntax	Value Length	Value Number	Value Initially
Filter Item Type	Enum(Filter Item Type)	-	1	-
Attribute Type	Type	-	1	-
Match Value	any	-	0-1	-
Filters	Object(Filter)	-	0-1	-
Initial Substring	String(*)	1 or more	0-1	-
Substring	String(*)	1 or more	0 or more	-
Final Substring	String(*)	1 or more	0-1 or more	-
License Request	Object(License Request)	-	0-1	-

FIG. 45

APPLE EXHIBIT 1102, Page 386 of 1048

```
Filter {
    Filter-Type AND
    Filter-Item {
        Filter-Item-Type SELECT
        Attribute-Type Product-Use-Authorization
        Filter {
            Filter-Type AND
            Filter-Item{
                Filter-Item-Type SELECT
                Attribute-Type Calling-Authorization
                Filter{
                    Filter-Type AND
                    Filter-Item {
                        Filter-Item-Type EQUALITY
                        Atribute-Type Producer
                        Match-Value "Digital"
                    }
                    Filter-Item {
                        Filter-Item-Type EQUALITY
                        Attribute-Type Producer
                        Match-Value "Amazing Database"
                    }
                }
            }
            Filter-Item {
                Filter-Item-Type EQUALITY
                Attribute-Type Producer
                Match-Value "Digital"
            }
            Filter-Item{
                Filter-Item-Type EQUALITY
                Attribute-Type Issuer
                Match-Value "Digital"
            }
            Filter-Item {
                Filter-Item-Type EQUALITY
                Attribute-Type Product-Name
                Match-Value "Amazing Graphics System"
            }
        }
    }
}
```

FIG. 46 Example Filter Value Notation

INTERNATIONAL SEARCH REPORT

PCT/US 92/03812

International Application No

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F1/00

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category[o]	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
Y	EP,A,0 332 304 (DIGITAL EQUIPMENT CORPORATION) 13 September 1989 cited in the application	1-3, 6-19,22, 24, 26-29, 31-33, 35-37,39 43-45
Y	see figure 1 cited in the application	
A	see column 3, line 31 - column 7, line 55	5,15,21, 25,30

-/--

[o] Special categories of cited documents :[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
09 SEPTEMBER 1992	17. 09. 92
International Searching Authority	**Signature of Authorized Officer**
EUROPEAN PATENT OFFICE	WEISS P.

Form PCT/ISA/210 (second sheet) (January 1985)

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)

Category °	Citation of Document, with indication, where appropriate, of the relevant passages	Relevant to Claim No.
Y	IBM TECHNICAL DISCLOSURE BULLETIN. vol. 31, no. 8, 1 January 1989, NEW YORK US pages 195 - 198; 'METHOD FOR MANAGING CLIENT/SERVER RELATIONSHIP IN THE AIX OPERATING SYSTEM'	1-3, 6-19,22, 24, 26-29, 31-33, 35-37,39 43-45
Y A	see the whole document	21

ANNEX TO THE INTERNATIONAL SEARCH REPORT ON INTERNATIONAL PATENT APPLICATION NO. US 9203812 SA 60557

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report. The members are as contained in the European Patent Office EDP file on 09/09/92
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP-A-0332304	13-09-89	US-A- 4937863	26-06-90
		JP-A- 2014321	18-01-90

EPO FORM P0479

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :		(11) International Publication Number:	WO 93/01550
G06F 11/34, H04L 9/00	A1	(43) International Publication Date:	21 January 1993 (21.01.93)

(21) International Application Number: PCT/US92/05387

(22) International Filing Date: 30 June 1992 (30.06.92)

(30) Priority data:

724,180	1 July 1991 (01.07.91)	US
907,934	29 June 1992 (29.06.92)	US

(71) Applicant: INFOLOGIC SOFTWARE, INC. [US/US]; 1223 Peoples Avenue, Suite 5405, Troy, NY 12180 (US).

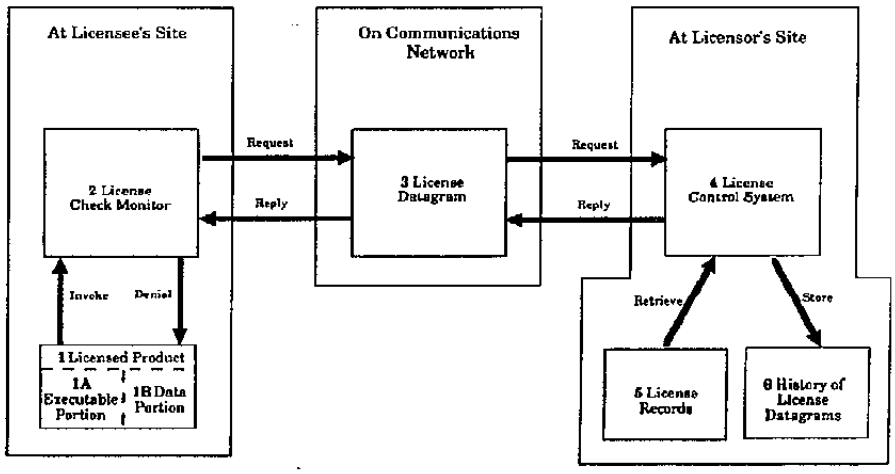
(72) Inventor: GRISWOLD, Gary, N. ; 1937 Regent Street, Schenectady, NY 12309 (US).

(74) Agents: LAZAR, Dale, S. et al.; Cushman, Darby & Cushman, Ninth Floor, 1100 New York Avenue, N.W., Washington, DC 20005-3918 (US).

(81) Designated States: AT, AU, BB, BG, BR, CA, CH, CS, DE, DK, ES, FI, GB, HU, JP, KP, KR, LK, LU, MG, MN, MW, NL, NO, PL, RO, RU, SD, SE, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IT, LU, MC, NL, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, SN, TD, TG).

Published
With international search report.

(54) Title: LICENSE MANAGEMENT SYSTEM AND METHOD



(57) Abstract

A license management system and method for recording (6) the use of licensed product (1), and for controlling (4) its use. A licensed product invokes a license check monitor (2) at regular time intervals. The monitor generates request datagrams (3) which identify the licensee and the product and sends the request datagrams over a communications facility to a license control system (4). The license control system maintains a record (6) of the received datagrams, and compares the received datagrams to data stored in its licensee database (5). Consequently, the license control system (4) transmits reply datagrams with either a denial or an approval message. The monitor (2) generates its own denial message if its request datagrams are unanswered after a predetermined interval of time. The datagrams are counted at the control system to provide billing information.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	FI	Finland	ML	Mali
AU	Australia	FR	France	MN	Mongolia
BB	Barbados	GA	Gabon	MR	Mauritania
BE	Belgium	GB	United Kingdom	MW	Malawi
BF	Burkina Faso	GN	Guinea	NL	Netherlands
BG	Bulgaria	GR	Greece	NO	Norway
BJ	Benin	HU	Hungary	PL	Poland
BR	Brazil	IE	Ireland	RO	Romania
CA	Canada	IT	Italy	RU	Russian Federation
CF	Central African Republic	JP	Japan	SD	Sudan
CG	Congo	KP	Democratic People's Republic of Korea	SE	Sweden
CH	Switzerland			SN	Senegal
CI	Côte d'Ivoire	KR	Republic of Korea	SU	Soviet Union
CM	Cameroon	LI	Liechtenstein	TD	Chad
CS	Czechoslovakia	LK	Sri Lanka	TG	Togo
DE	Germany	LU	Luxembourg	US	United States of America
DK	Denmark	MC	Monaco		
ES	Spain	MG	Madagascar		

LICENSE MANAGEMENT SYSTEM AND METHOD

BACKGROUND

Field of the Invention

The present invention generally relates to systems for managing licenses of products such as computer software, video games, CD-ROM information, movies and other video products, music and other audio products, multimedia products, and other systems for up-to-date recording of actual usage of such a licensed product to enable efficient billing therefor.

Description of Related Art

Licenses for information products such as computer software, music, video products and the like usually provide licensees with limited rights. The licenses may restrict sites of use, duration of use, or number of concurrent uses of the products. The licenses also may limit the use of the products depending on currentness of licensee's payments. However, enforcing the conditions of the licenses is difficult, because, in general, the licensed products may be easily copied or "pirated" and used without the licensor's knowledge.

Compliance with limited license rights has been encouraged with copy protection. Known methods of computer software copy protection include putting a

physical hole or mark on the diskette containing a product, or placing data on the diskette in a location where no data is expected. A disk with an illegally copied software product usually would not contain the marks. At the beginning of its operation, a copy-protected, but illegally copied software product would search its own diskette for the marks. Upon failing to detect the marks, the software would abort from its normal procedures.

Most software products sold today do not have such copy protection, partly because copy protection renders legitimate duplication of copy protected software difficult, but not impossible. Copy protection frustrates the making of legitimate copies, while not eliminating unauthorized copying. Many software publishers have experienced higher sales by eliminating copy protection schemes.

Another method for enforcing limited licensing rights of computer software is described in U.S. patent No. 4,932,054 to Chou. Chou describes a "coded filter" hardware device which is plugged into a port of a computer. The "coded filter" outputs an authorization control code when a predetermined control code is sent to it. The licensed software functions properly only if the "coded filter" transmits the correct authorization control code to the software.

While devices such as described by Chou have existed for several years, they have not been well accepted by the market. Since the device is attached to the outside of a computer, it can easily be lost or stolen, preventing the use of licensed software. In addition, if a licensee purchased a number of software

products, each of which used Chou's protection scheme, the licensee would collect a stack of "coded filters."

Hershey, in U.S. patent No. 4,924,378, describes a method for limiting the number of concurrent uses of a licensed software product. Each workstation of a network has a license storage area in its local memory. License Management System (LMS) daemons are provided in the network in a number corresponding to the permissible number of concurrent uses of the software product. To use the software, a work station stores a daemon in its license storage area. If all daemons are in use, no further work stations may use the software.

Robert et al., in U.S. patent No. 4,937,863, describe a similar invention. This invention includes a license management facility which accesses a database of license information related to licensed computer software programs. When a user attempts to use a licensed program, the license management facility first checks the database. Access to the licensed product is prevented if licensing conditions related to the product are not satisfied (e.g., expiration of licensing dates, etc).

While the Robert et al. and Hershey patents show effective techniques for controlling licensed computer software, each also reveals components that cannot be easily managed by an average user. A system manager, or someone with special access privileges to the internals of a machine, must install the licensed software. This hinders the distribution of the software.

Licensable products other than computer software have not generally been copy-protected. For example,

video tapes can be easily copied by anyone with two VCR machines, and audio tapes and music CDs can be easily copied to tape. Computer CD-ROMs can be copied to magnetic disk; however, their large information storage capacity relative to that of magnetic disks makes this a very expensive proposition. The introduction of digital audio tape is being delayed, because some view its ability to easily produce very high quality copies as a threat to music royalties.

Hellman, in U.S. patent No. 4,658,093, describes means to bill by usage. This is accomplished via communication of an encrypted authorization code from a licensor to a base unit at the licensee's site. The encrypted authorization code contains information related to an identification of the base unit, a number of uses requested, and a random or non-repeating number; however, implementation of Hellman's scheme requires a "base unit", such as a computer, video game unit, record player, video recorder, or video disk player, with a unique identification number. The requirement is difficult to satisfy, because, at the present, only a fraction of such systems on the market have an internally readable serial number for identification. In addition, vendors of these systems provide no guarantees for the uniqueness of any given device's serial number. Furthermore, an internal serial number can change when hardware maintenance is performed on the device. Also, Hellman's approach requires that an identical copy of each software product be stored at the authorization site. These copies are used in the generation of unique keys. The unstated assumption that all copies of a specific version of a software

product are identical is unrealistic. Minor bug fixes to software are often made without generating a new version of the product. Also, some software products, such as those which run on Macintosh computers, are self-modifying.

While Hellman's invention counts each use of the software, it does not monitor the duration of use. Thus, Hellman's system would not be able to bill for extensive use of licensed software if the software were continuously operated. Finally, while Hellman suggests the inclusion of an automated communication system as part of his invention, he does not disclose how this communication system could be implemented. Instead, he mentions non-automated use of telephone and mail. In summary, Hellman's patent is an interesting discussion of cryptographic techniques, but it does not provide a practical, real-world implementation of those techniques.

Shear, in U.S. Patent No. 4,977,594, describes a system and method to meter usage of distributed databases such as CD-ROM systems. The method describes a hardware module which must be part of the computer used to access the distributed database. This module retains records of the information viewed. Once the module storage is filled, the module must be removed and delivered to someone who will charge for the usage recorded therein and set the module back to zero usage. Like Hellman's method, this method requires a hardware module which must be incorporated within the computer so the system can control user access. No database publisher will be able to use this method until there are a very large number of units containing such modules. Hardware manufacturers

will be hesitant to include the module in the design of their computers until there is sufficient demand from customers or publishers for this system. The method and apparatus according to the present invention can be implemented entirely in software and hence does not require special, dedicated computer subsystems.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a license management system and method which can ensure that a licensed product is used only on machines under which it is licensed.

It is another object of the present invention to provide a license management system and method which may terminate access to a licensed product once its license has expired.

It is yet another object of the present invention to provide a license management system and method which may terminate access to a licensed product when payment for a license is overdue.

It is a further object of the present invention to provide a license management system and method which can limit the number of concurrent uses of a licensed product.

It is yet another object of the present invention to provide a license management system and method which can bill licensees for the duration of actual usage of a licensed product.

The present invention provides an advantageous feature of quickly and effectively implementing license agreements between a licensor and licensee.

The present invention provides another advantageous feature of allowing logic used to control licenses to be easily changed.

The present invention provides yet another advantageous feature of detecting, at the licensor's site, many types of attempts to alter the license management system.

The present invention provides a further advantageous feature of permitting anyone without special access privileges to install a licensed product.

In the present invention, a licensed product generates request "datagrams," messages transmitted over a communications network. The request datagrams are sent to the licensor's site. At the licensor's site the datagram is compared to information stored in a license database. After the comparison, a reply datagram is sent to the licensee. Upon receiving the reply datagram, the licensed product reacts in accordance with the instructions therewithin. For example if a reply datagram contained a "denial," the licensed product would display an appropriate message to the user and then suspend further execution of its programs.

In the present invention, the licensed product is implemented on a network node attached to a communications network that includes the licensor. The network node may be a computer, a CD-ROM player, a tele-computer or other multimedia machine, or any other appropriate device. The node may also be an intelligent type of consumer electronic device used for presenting information, such as an intelligent television, VCR, videodisk player, music CD player,

audio tape player, telephone or other similar device. Further, the communications network may be any two-way network such as a computer network, telephone network, a cellular telephone network or other wireless network, a two-way cable TV network, or any other equivalent system.

Should the user detach the node from the network, the licensed product will fail to receive reply datagrams. Upon several failures to receive reply datagrams, the licensed product will generate its own denial.

After a request datagram has been sent out, a user may be permitted to use the licensed product for a limited duration. This feature may be necessary because of the delays in network communications. When networks are sufficiently fast, use of a licensed product can be postponed until the reply datagram is received.

In the preferred embodiment of the present invention, licensees' network addresses are used to identify the licensees. Other embodiments may use a licensed product serial number or hardware serial numbers for the identification.

A licensed product as in the present invention generates a request datagram after each period of product use. The number of request datagrams received by the licensor can be used to bill the licensee. For example, if datagrams are sent after every hour of product use, the licensee will be billed for the amount equal to the number of request datagrams received by the licensor multiplied by the hourly rate.

The embodiments of the present invention may incorporate a query system at a licensor's site for reporting on problem datagrams. This would allow the licensors to take appropriate actions in accordance with problems associated with each datagram.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other objects and advantages of this invention will become more apparent and more readily appreciated from the following detailed description of the presently preferred exemplary embodiment of the invention, taken in conjunction with the accompanying drawings, of which:

FIGURE 1 is a general block diagram of the preferred exemplary embodiment of the present invention;

FIGURE 2 shows representative diagrams of the contents and formats of data at licensee's site, contained in datagrams, and at licensor's site;

FIGURE 3 illustrates a sequence of representative operations executed at the licensee's site and at the licensor's site, together with required inputs for the execution of the operations and with outputs produced therefrom;

FIGURE 4 illustrates a sequence of representative operations to send a request datagram, together with required inputs for the execution of the operations and with outputs produced therefrom;

FIGURE 5 illustrates a sequence of representative operations when a reply datagram is overdue, together with required inputs for the execution of the operations and with outputs produced therefrom;

FIGURE 6 shows a sequence of representative operations to process a reply datagram, together with required inputs for the execution of the operations and with outputs produced therefrom;

FIGURE 7 shows a sequence of representative operations to generate an authorization code, together with required inputs for the execution of the operations and with outputs produced therefrom; and

FIGURE 8 shows a sequence of representative operations to send a reply datagram, together with required inputs for the execution of the operations and with outputs produced therefrom.

DETAILED DESCRIPTION OF THE PRESENTLY PREFERRED EXEMPLARY EMBODIMENT

As shown in FIGURE 1, a licensed product 1 is located at a licensee's site. Product 1 may include a data portion 1B and a functional portion 1A such as computer software product or any other kind of information product used to control use of data portion 1B. If data portion 1B is CD-ROM database information, functional portion 1A should enable the licensee to search indexes and display text. If data portion 1B is video information, functional portion 1A should control the display of the video information. For audio information, functional portion 1A should play the audio information. If data portion 1B is an electronic book, functional portion 1A should display and turn pages. The above examples show some of the ways functional portion 1A can control data portion 1B; however, they are hardly exhaustive.

By including in product 1 both information and software which controls the information, product 1 is

an executable product. Non-software information in product 1 is preferably encrypted so that it cannot be easily extracted from the product.

License check monitor 2 sends license datagrams 3 to the licensor and also receives license datagrams 3 from the licensor. License check monitor 2 also prevents further use of product 1 when a datagram 3 containing a "denial" message is received.

License datagrams 3 are messages that describe information related to the use of licensed product 1. Datagrams 3 are sent over a communications network between the licensee and licensor. Initially, the licensee sends a request datagram 3 over the network to the licensor. The licensor then returns a reply datagram containing either an approval or denial. It is also possible to implement the present invention by having the licensor transmit a reply datagram only for approvals.

At the licensor's site, license control system 4 makes licensing decisions by comparing request datagram 3 with license records 5. After the comparison, control system 4 stores information related to request datagram 3 into history of license datagram record 6. It is noted that request datagrams 3 are periodically sent while product 1 is in use. Thus, the history of license datagrams in record 6 provides means for measuring the duration of use of product 1.

Representations of data and records stored at the licensee's site, contained in datagrams, and stored at the licensor's site are illustrated in FIGURE 2. At the licensee's site, network service 7, which handles delivery and transmission of datagrams 3, supplies

network address 8. It is by this address that license control system 4 identifies a location of use of product 1.

Licensed product record 9 is contained within monitor 2. Within the license product record 9 is an identification record 10, which contains the following two items: licensor's network address 11, and product model number 12 that identifies product 1. When a licensor has only one product, or uses different licensor network addresses 11 for each product, product model number 12 may not be needed.

Datagram sent record 13 stores information about the last sent datagram 3. It includes a datagram number 14, which uniquely identifies the last transmitted datagram 3, and the date and time 15 when the last datagram 3 was sent from the licensee's site.

Licensed product record 9 also contains control parameters record 16, which is used for controlling the timing of key events in the communication of license check monitor 2 with license control system 4. Send interval 17 specifies a time interval between each transmission of a new datagram 3 from the licensee to the licensor.

Wait interval 18 is the length of time that monitor 2 waits to receive a reply datagram 3 before resending the same request datagram 3. The duration of this interval depends on the speed of the communications network being used to deliver datagrams 3.

Disconnect allowed interval 19 is the duration of time that monitor 2 allows product 1 to be used without a reply datagram 3 from the licensor. The duration of this interval depends on the reliability

of the communications network. The interval must be long enough to take into consideration network downtime. For example, suppose a message was sent from the licensor and the network went down just afterwards. Disconnect allowed interval 19 should be long enough to allow the network to resume its normal operation and successfully deliver datagrams 3 from the licensor; otherwise, the licensee would be forced to stop using product 1 until the network was operational.

License datagram 3 contains header 20. Header 20 is used during execution of low level communication protocols within the network. Source network address 21 is the network address from where datagram 3 is sent. Destination network address 22 is the network address to where datagram 3 is sent. Additional data may be included in header 20 if required by low level protocols used in delivering datagrams 3.

Data 23, a part of datagram 3, conveys a message, and contains a number of fields. Product model number 24 and datagram number 25 identify product 1 and datagram 3, respectively. It is noted that retransmitted datagrams have an identical datagram number. Duplicate datagrams must be identified at a licensor's site so that they do not all contribute in billing a licensee.

Each datagram number 25 is unique for each request datagram 3 transmitted from the licensee, except for retransmitted datagrams. This allows a reply datagram 3 received by a licensee to be verified as an actual reply to a request datagram 3 from that licensee, as explained below.

Number of processes running 26 is the number of concurrent uses of product 1 at the time datagram 3 is sent. Authorization code 27 is used on reply datagrams 3 to indicate an approval or a denial. Message text 28 contains a message which will be displayed to the user upon a denial.

License database 29 at the licensor's site holds records of information about customers, licenses, and license usage. The types of information within license database 29 of the present embodiment are shown in FIGURE 2. However, a specific license management system may require its license database to hold types of information other than those in FIGURE 2. For example, licensee name and address may be incorporated as a part of a license database 29.

License record 5 contains information on licenses. Licensee network address 30 identifies a precise network node which is licensed to use product 1. If request datagrams are received which do not originate from known licensee network addresses 30, reply datagrams containing denial messages are transmitted. Product model number 31 is the model number of a licensed product. Termination date 32 is the expiration date of a license. When the license of a product is issued for an unlimited duration, termination date 32 should reflect a date very far into the future, relative to the licensing date.

The present embodiment allows licenses to be paid for in a lease-like or rental fashion. If a licensee were to rent or lease product 1, paid through date 33 would reflect the date through which the licensee has paid for using the product. Grace period 34 is the time interval for which the licensee is allowed to be

delinquent before services are disconnected. Grace period 34 would reflect a very large time interval if the license is not of a lease-like or rental type. When the license provides for a limit on the number of concurrent uses of a product 1, number of processes licensed 35 contains the limiting number. When the license does not provide for such a limit, number of processes 35 should be a very large number.

History of license datagrams 6 is an archive of datagrams 3 received from the licensee.

FIGURE 3 illustrates operations executed at the licensee's site and at the licensor's site. An overview of the processing at the licensee's site is described by steps 101.0 to 106.0, and an overview of the processing at the licensor's site is described by steps 107.0 to 110.0.

At the licensee's site, at step 101.0, product 1 invokes monitor 2. This is accomplished by first establishing monitor 2 as a handler for a timer expiration interrupt signal and for received datagrams 3. Next, a timer is set with a very short time to cause an initial call to monitor 2. At step 102.0, monitor 2 computes a time 36 since the last datagram was sent by determining the difference between the current date and sent time and date and time 15 that a datagram was last sent from the licensee's site. When product 1 commences execution, datagram sent date and time 15 is set to "null." Thus, time since send 36 is very large at the beginning of the monitor's execution. At step 103.0, time since send 36 is compared to send interval 17. If time since send 36 is greater than send interval 17, then a request datagram is transmitted, per the steps described in

FIGURE 4. Step 104.0 first checks if a reply to the last datagram has arrived and if wait interval 18 has expired. If a reply has not arrived and the wait interval has expired, steps 104.1-104.3 (FIGURE 5) are executed. Step 105.0 processes authorization code 27 in a reply when the reply is received, in accordance with steps 105.1 to 105.5 (FIGURE 6). At step 106.0, product 1 resumes normal execution of its programs until the next interrupt signal is generated.

At the licensor's site, license control system 4 receives and processes datagram 3, in accordance with steps 107.0 to 110.0. Step 107.0 receives request datagram 3. Step 108.0 generates authorization code 27, per steps 108.1 to 108.8 (FIGURE 7). Step 109.0 creates reply datagram 3 and transmits the datagram to the licensee via steps 109.1 to 109.5 (FIGURE 8).

FIGURE 4 shows the procedure which monitor 2 follows for sending request datagram 3 to the licensor. Step 103.1 sets source network address 21 in datagram 3 to the network address 8 of the licensee's location on the network. Step 103.2 sets destination network address 22 to licensor's network address 11. Step 103.3 encrypts product model number 12 for datagram 3. Step 103.4 assigns a unique number to datagram 3, encrypts the number, and stores it as datagram number 14. This number is altered when an entirely new datagram 3 is sent. Datagrams which are retransmitted have the same datagram number 25 as the original. As already explained, this allows license control system 4 to identify duplicate datagrams.

Step 103.5 counts the number of processes using product 1, currently running, encrypts the count, and stores the encryption as the number of processes

running 26. In the UNIX operating system, this procedure could be performed using the command "ps" to obtain a list of current processes, the command "grep" to extract the processes of product 1, and "wc" to count the number of processes. Step 103.6 sets authorization code 27 to number 255 and encrypts the number.

Number 255 indicates that datagram 3 is a request for authorization. Such an indication is needed to guard the present system against the following steps for circumventing the present invention: intercepting outgoing datagrams; and inputting the intercepted datagrams to monitor 2.

Step 103.7 stores the current date and time as sent date & time 15. This date is needed to compute when to send the next datagram 3. Step 103.8 assigns a value to send interval 17, which sets an alarm for invoking monitor 2 to send the next datagram 3. Step 103.9 sends datagram 3.

In the present embodiment a datagram is transmitted via a connectionless datagram service. Methods for transmission are well documented for some networking systems. For example, TCP/IP (Transport Control Protocol/Internet Protocol) includes a connectionless protocol called UDP (User Datagram Protocol). A method for sending a datagram using UDP protocol from a SUN Microsystem computer is documented in a SUN manual titled, Network Programming Guide, in section 9 titled "Transport Level Interface Programming."

Step 103.10 sets another alarm using wait interval 18 for retransmitting datagram 3, if no reply datagram has been received. The alarm causes monitor

2 to be invoked for checking whether a reply datagram 3 has been received. Monitor 2 will transmit a duplicate of the previously transmitted datagram, if no reply has been received. After the execution of step 103.10, "Send License Datagram" procedure returns system control to step 104.0 in FIGURE 3.

FIGURE 5 shows the operation of the "Reply Datagram is Overdue" procedure. Step 104.1 compares time since the last datagram was sent 36 to disconnect allowed interval 19, which, as described above, is the interval that product 1 is allowed to operate even if a reply is overdue. If time since send 36 is smaller than disconnect allowed interval 19, datagram 3 is retransmitted via executing step 103.9 in FIGURE 4. Step 104.2 "disconnects" product 1 from further service, if time since send 36 is greater than disconnect allowed interval 19.

Step 104.2 comprises a sequence of sub-steps 104.2.1-104.2.3. Step 104.2.1 assigns number 5 to authorization code 27 in the current datagram being processed. Value 5 is interpreted by monitor 2 as a denial. Step 104.2.2 sets message text 28 to the following: "A reply from licensor to numerous authorization requests was never received. This product must be connected to a communications network in order to function." Step 104.2.3 transfers system control to step 105.3 in FIGURE 6. Step 105.3 processes the current denial datagram 3 as if it were just received.

Through the execution of steps 104.1-104.3, the present system permits the use of product 1 for a prescribed period of time. After the prescribed

period of time has elapsed, the present system generates a denial.

FIGURE 6 illustrates the steps which monitor 2 follows in processing a reply datagram 3. Step 105.1 decrypts all encrypted data in the received datagram. Step 105.2 compares datagram number 25 with datagram number 14 associated with the last datagram. If datagram number 25 is not equal to datagram number 14, step 105.2 ignores the current datagram and transfers procedural control to step 103.9 (FIGURE 4) in order to resend the last transmitted datagram. After disconnect allowed interval 19 elapses, monitor 2 generates a denial.

In essence, step 105.2 guards against the circumvention of the present invention via: (1) intercepting a reply datagram 3 (from the licensor) containing an approval (2) storing the reply datagram 3; and (3) inputting the stored datagram to monitor 2.

If the execution of step 105.2 does not transfer its procedural control to step 105.3, and if authorization control 27 is not zero (indicating an unqualified authorization has not been received), step 105.3 processes authorization code 27 via steps 105.3.1 to 105.3.3. Step 105.3.1 retrieves message text 28 from datagram 3. If message text 28 is null, then the current datagram 3 is ignored, and monitor 2 resends the last transmitted datagram 3. Step 105.3.1 further protects the present system from attempts to generate fake datagrams and to feed the fake datagrams to monitor 2 by checking for a proper authorization code of zero.

If message text 28 is not null, step 105.3.2 presents the message 28 to the user on an output

device such as a CRT screen. Step 105.3.3 terminates the current use of product 1. This step may be implemented by subroutine or function call to a simple exit that saves any current user data to a file. Alternatively, product 1 may be designed so that, upon being directed to terminate further execution, it first gives the user an opportunity to save their data.

If authorization code 27 is zero, step 105.4 allows further use of product 1. Step 105.5 returns procedural control to 106.0 on FIGURE 3.

FIGURE 7 shows a sequence of operations within the "Generate Authorization Code" procedure. The procedure produces appropriate authorization code 27 when a request datagram 3 is received at the licensor's site.

Step 108.1 decrypts all encrypted data in the received datagram 3. Using source network address 21 and product model number 24 in the datagram 3, step 108.2 searches the license database 29 for matching licensee network address 30 and product model number 31. If license database 29 does not contain a record of product model number 24 of the product 1 being licensed to the licensee, step 108.3 sets authorization code 27 of its reply datagram 3 to 1 (i.e., the sending node is not a registered address) and authorization is denied.

Step 108.3 prevents copies of product 1 from being installed on multiple nodes independently of whether they are within or outside the licensee's organization. Step 108.3 also prevents the licensee from transporting product 1 from one node to another node without the licensor's approval. This is

important because the two nodes may have different processing capacities, and they may be billable at different rates.

If the date a request datagram is received is later than license termination date 32, step 108.4 sets authorization code 27 to number 2 (i. e., license has expired). Step 108.4 allows the licensor to fix licensing periods, or to determine free trial periods for the use of the product. The licensing period may be extended by resetting license termination date 32 at the licensor's site.

If the date when the datagram is received is later than the paid through date 33 as extended by the grace period 34, step 108.5 sets authorization code 27 to 3 (i.e., payment is past due).

If the number of processes running 26 exceeds a licensed number of concurrent uses of product 1 (at a particular node), then step 108.6 sets authorization code 27 to 4 (i.e. concurrent process usage limit is exceeded).

Step 108.7 sets authorization code 27 to 0 indicating processing can continue. It is noted that steps 108.3-108.7 are a part of a

```
IF (x1) then (y1)
ELSE if (x2) then (y2)
ELSE if (x3) then (y3) ...
```

statement of a procedure (e. g., FORTRAN, PASCAL, C, etc). Thus, only one of the steps 108.3-108.7 is executed. Step 108.7 sets authorization code 27 to 0 (indicating approval of further use) only if steps 108.3-108.6 do not execute the THEN portion of each step. Step 108.7 also stores the received datagram 3 in history of license datagrams 6.

Step 108.8 is the last of authorization processing rules 108.1-108.7. After the execution of steps 108.3-108.7, step 108.8 returns procedural control to step 109.0 in FIGURE 3.

FIGURE 8 illustrates the steps which license control system 4 follows to send reply datagram 3 to the licensee.

Step 109.1 encrypts authorization code 27 and writes the encrypted code into datagram 3. Next, step 109.2 writes message text 28 corresponding to authorization code 27 into datagram 3.

Step 109.2 may be replaced with the following method for relaying proper messages to a product user. Proper messages corresponding to each authorization code is stored in monitor 2 at each licensee's site. Upon reception of a reply datagram 3, monitor 2 would locate within itself the proper message corresponding to the authorization code, and use the message for various purposes. This method would reduce the size of reply datagrams 3. However, if the licensor wanted to implement new denial codes, each product would need to somehow incorporate the new message associated with the new denial code into itself. The list of messages, one of which may be written as message text 28, are as follows:

AUTHORIZATION CODE	TEXT MESSAGE
1	This product is not licensed to run at this location. Please contact the licensor to either license this product, or move an existing license of your organization to this location. Use of this product at this

	location is discontinued until this problem is resolved.
2	Your license on this product has expired. Please contact licensor in order to have your license extended. Use of this product is discontinued until this problem is resolved.
3	Payment on this licensed product is over due and past your grace period. Please have your accounting department send payment in order to continue your license. Use of this product is discontinued until this problem is resolved.
4	Your current use of this licensed product exceeds limits for the number of uses your organization has licensed. Please try again later.
5	A reply from licensor to numerous authorization requests was never received. This product must be connected to a communications network in order to function.
0	Authorization is OK. There is no message.

Step 109.3 swaps source network address 21 and destination network address 22. Step 109.4 transmits datagram 3 back to monitor 2.

At step 109.5, a communications network delivers datagram 3 to monitor 2. Subsequently, procedural control returns to step 107.0 in FIGURE 3 to process the next datagram 3.

Although only a few exemplary embodiments of this invention have been described in detail above, those skilled in the art will readily appreciate that many

modifications are possible in the preferred embodiments without materially departing from the novel teachings and advantages of this invention. For example, product 1 was described as sometimes consisting of information as well as software which controls the information. This approach provides the greatest flexibility, but it is also possible to include the software which controls the information in the networked machine at the licensee's site. In this case, product 1 is split, with part of it on media and part on the licensee's machine. By doing this, some space can be saved on the media containing product 1, but the capabilities of these products will be limited by the standard functions available on these machines.

Also, the presently described embodiment includes a product 1 which is at the licensee's site. This implies that product 1 is on some physical media such as diskette, tape, or CD. However, product 1 can be electronically delivered over communications lines to the licensee and therefore might exist in the memory of the licensee's machine, rather than any physical media. In the case of a product such as music, radio programs and the like, product 1 may even be broadcast to the licensee's site for playback; thus, the product 1 would not even be "resident" in the licensee's machine.

The presently described embodiment allows the licensee to access the licensed product concurrent with the sending and receiving of datagram 3. In this way, the present invention does not inconvenience the legitimate licensee; however, for sensitive licensed products such as confidential information, the license

check monitor 2 can prevent access to the product 1 until an authorization reply datagram 3 is received.

Further, monitor 2 could be realized as an integral part of product 1. Monitor 2 could also be implemented as: 1) a separate process which is the parent process of product 1 (Such a parent process would have the authority to cancel the use of product 1); 2) a single system level task which controls license checking of all products at the licensee's site; and 3) custom logic in a digital integrated circuit (the present invention could be implemented as hardware instead of software).

Also, though the above embodiment has been described as being implemented on a computer system network where operator messages are provided on a CRT monitor or the like, the invention may be practiced on other hardware platforms by incorporating appropriate changes known to those of ordinary skill in the art. For example, in an alternative hardware embodiment such as a music or video playback device, monitor 2 is invoked by the licensee's action of pushing the "play" or similar button, and in a broadcast music application or similar system, the monitor may be invoked simply by turning the device on. The processing of monitor 2 is as described in the presently described embodiment. However, when a denial message is received or generated, monitor 2 must be able to switch "play" to "off".

The presently described embodiment is designed to be used in conjunction with a connectionless UDP (User Datagram Protocol) in the TCP/IP protocol suite as an underlying protocol. However, the present invention could also be realized using a slower,

connectionless protocol such as electronic mail or a variety of connection protocols (e. g., File Transfer Protocols (FTP), Telnet).

It is noted that protocol suites quite different from TCP/IP could be used, such as ISO (International Standards Organization) protocol. In addition, datagrams 3 could be sent over telephone systems with communications protocols such as those specified by CCITT (Consultative Committee on International Telephony and Telegraphy). In this case, telephone numbers could serve as network addresses 21, 22. Communications protocols for wireless communications such as cellular telephone can also be used to send the datagram 3.

Accordingly, all such modifications are intended to be included within the scope of this invention as defined by the following claims.

WHAT IS CLAIMED IS:

1. A method for monitoring the use of a licensed product, comprising the steps of:

generating, at regular time intervals, datagrams including an address in a communications facility, said facility address identifying a licensee;

automatically sending said datagrams from at least one licensee's site over said facility to a licensor's site while said licensed product is in use;

receiving said datagrams at said licensor's site;

storing an indication of receipt of each of said datagrams; and

counting said datagrams from each licensee as an indication of the use by the licensee of said licensed product.

2. A method as in claim 1 further wherein:

said generating step includes the step of incorporating a model number of said product in said datagrams; and

said counting step includes the step of separately counting datagrams for each product model number for each licensee.

3. A method as in claim 1, wherein said generating step includes the step of automatically obtaining said facility address that identifies said licensee from said facility without any data being provided by said licensee.

4. A method for controlling use of a licensed product comprising the steps of:

generating a request datagram including an address in a communications facility, said facility address identifying a licensee;

automatically sending said request datagram from at least one licensee's site over said facility to a licensor's site while said licensed product is in use;

receiving said request datagram at said licensor's site;

comparing said received request datagram with rules and license data at said licensor's site to determine if use of said licensed product is authorized;

sending a reply authorizing datagram to said licensee's site if use of said licensed product is approved; and

receiving said reply authorizing datagram at said licensee's site and denying the use of said product when no reply authorizing datagram is received.

5. A method as in claim 4, wherein:

said generating step includes the step of incorporating a model number of said product in said datagram;

said comparing step includes the step of comparing said rules and license data for a particular model number; and

said sending step includes the step of transmitting said reply datagram for each product model number.

6. A method as in claim 4, wherein said generating step includes the step of automatically obtaining said facility address that identifies said licensee from said facility without any data being provided by said licensee.

7. A method as in claim 4 further comprising the step of sending a reply denial datagram if use of said licensed product is not approved as determined in said comparing step, said step of automatically sending said request datagram from a licensee's site including the step of resending said request datagram if neither a reply authorizing datagram nor a reply denial datagram is received from said licensor's site within a predetermined time from sending said request datagram from said licensee's site.

8. A method as in claim 4, wherein said step of automatically sending said request datagram from said licensee's site includes the step of sending a request datagram at regular time intervals.

9. A method as in claim 4, wherein:

said generating step includes the step of providing a datagram identification code within said datagram;

said reply datagram sending step includes the step of inserting the same datagram identification code in said reply datagram; and

said reply receiving step rejects said reply authorizing datagram if the datagram identification code included in said reply authorizing datagram does

not match the datagram identification code included in said request datagram.

10. A method as in claim 4, wherein:

said comparing step includes the step of comparing said facility address that identifies said licensee with a list of valid licensee addresses to determine if said facility address is a valid address; and

said reply authorizing datagram is not sent if said facility address that identifies said licensee is not valid.

11. A method as in claim 10 further comprising the step of sending a reply denial datagram if said facility address that identifies said licensee is not valid.

12. A method as in claim 4, wherein:

said comparing step includes the step of comparing a license expiration date with a date at which said datagram is received; and

said reply authorizing datagram is not sent if the license expiration date is later than the date at which said datagram is received.

13. A method as in claim 12, further comprising the step of sending a reply denial datagram if the license expiration date is later than the date at which said datagram is received.

14. A method as in claim 4, wherein:

said comparing step includes the step of checking currentness of payments from said license; and

said reply authorizing datagram is not sent if payment is overdue.

15. A method as in claim 14, further comprising the step of sending a reply denial datagram if payment is overdue.

16. A method as in claim 4, wherein:

said generating step includes the step of incorporating in said datagram data indicative of the number of processes currently using said product at said licensee's site;

said comparing step includes the step of comparing the number of processes using said product at the licensee's site to an authorized number; and

said reply authorizing datagram is not sent if said number of processes using said product exceeds said authorized number.

17. A method as in claim 16, further comprising the step of sending a reply denial datagram if said number of processes using said product exceeds said authorized number.

18. A method as in claim 4, wherein said sending step includes the steps of sending said reply authorizing datagram when use of said product is approved and sending a reply denial datagram when use of said product is not approved, said receiving step

denying use of said product when said reply denial datagram is received.

19. A method as in claim 18, wherein said receiving and denying step denies use of said product when neither a reply authorizing datagram nor a reply denial datagram is received within a predetermined time after said request datagram is sent.

20. A method as in claim 18, further comprising the step of indicating, at a licensee's site, a reason for denial when said reply denial datagram is received.

21. A method as in claim 4, wherein:

said licensed product comprises an executable portion and a data portion; and

said method further comprises a step of controlling use of said data portion with said executable portion.

22. A method as in claim 4 further comprising a step of allowing use of said licensed product before a reply datagram is received.

23. A system for controlling licensed product comprising:

a communications facility to which at least one licensee having a license for operating a licensed product from the licensor is connected;

monitoring means, connected to said facility at a site of each said licensee, for generating a request datagram including an address of said licensee

on said facility and transmitting said request datagram over said facility to a site of said licensor, and for receiving and processing a reply datagram; and

controlling means, connected to said facility at said licensor's site, for receiving said request datagram, comparing said request datagram with rules and license data to determine if use of said licensed product is authorized and sending a reply authorizing datagram to said licensee's site if use of said product is approved; and

said monitoring means including means for denying use of said licensed product when no reply authorizing datagram is received.

24. A system as in claim 23, wherein:

said monitoring means sends request datagrams at regular time intervals during use of said licensed product; and

said controlling means further comprises means for counting said request datagrams received at said controlling means and means for computing an amount to be billed to said licensee in response to said counting.

25. A system as in claim 23 wherein:

said monitoring means incorporates a model number for said product in said request datagram; and

said controlling means comprises means for counting datagrams for each product model number for each licensee, in order to compute an amount to be billed to each licensee.

26. A system as in claim 23, wherein said monitoring means automatically obtains said facility address of said licensee from said facility without any input from said licensee.

27. A system as in claim 23, wherein:

said controlling means sends a reply denial datagram to said licensee's site if use of said product is not approved; and

said monitoring means resends said request datagram if no reply authorizing datagram and no reply denial datagram is received within a predetermined period of time after said requesting datagram is sent.

28. A system as in claim 23, wherein said monitoring means transmits request datagrams at predetermined time intervals.

29. A system as in claim 23, wherein:

said monitoring means incorporates a unique identification code in said request datagram;

said controlling means incorporates the same request datagram identification code in said reply authorizing datagram; and

said monitoring means rejects any reply authorizing datagram which does not include the same identification code as included in said request datagram.

30. A system as in claim 23, wherein said controlling means compares said facility address of said licensee with a list of valid licensee facility addresses and does not generate a reply authorizing

datagram if said facility address of said licensee is not valid.

31. A system as in claim 30, wherein said controlling means sends a reply denial datagram when said facility address is not valid.

32. A system as in claim 23, wherein said controlling means compares an expiration date of a license of said product with a date at which said request datagram is received by said controlling means, and does not generate a reply authorizing datagram, thus denying use of said product, if the license expiration date is earlier than the date at which said request datagram is received.

33. A system as in claim 32, wherein said controlling means sends a reply denial datagram if the license expiration date is earlier than the date at which said request datagram is received.

34. A system as in claim 23, wherein said controlling means generates a reply authorizing datagram, thus denying use of said product, if a payment for the use of said product is overdue.

35. A system as in claim 34, wherein said controlling means sends a reply denial datagram if payment for the use of said product is overdue.

36. A system as in claim 23, wherein:

said monitoring means includes in said request datagram data indicative of the number of

processes, at a licensee's site, currently using said product; and

said controlling means does not generate a reply authorizing datagram, thus denying a use of said product, if more than a predetermined number of processes using said product are running at the licensee's site.

37. A system as in claim 36, wherein said controlling means sends a reply denial datagram if more than said predetermined number of processes using said product are running at the licensee's site.

38. A system as in claim 23, wherein said controlling means sends a reply denial datagram if use of said product is not approved.

39. A system as in claim 38, wherein said monitoring means denies use of said licensed product when no reply authorizing datagram and no reply denial datagram is received within a predetermined time from the sending of said request datagram.

40. A system as in claim 38, further comprising means for indicating, at a licensee's site, a reason for denial when said reply denial datagram is received.

41. A system as in claim 23, wherein:

said licensed product comprises an executable portion and a data portion; and

said system further comprises means for controlling use of said data portion with said executable portion.

42. A system as in claim 41, wherein said data portion controlling means is disposed within said executable portion.

43. A system as in claim 41, wherein said data portion controlling means comprises a first partial controlling means disposed within said executable portion and a second partial controlling means disposed within said monitoring means.

44. A system as in claim 23, wherein said monitoring means includes means for permitting use of said licensed product before a reply datagram is received.

45. A system for monitoring product comprising:

a communications facility to which at least one licensee having a license for operating a licensed product from a licensor is connected;

monitoring means, connected to said facility at a site of each said licensee, for generating datagrams including an address of said licensee on said facility and transmitting said datagrams at periodic intervals over said facility to a site of said licensor; and

control means, connected to said facility at said licensor's site, for receiving said request datagrams, storing an indication of receipt of each of said datagrams and counting said datagrams from each

licensee as an indication of the use by the licensee of said licensed product.

46. A system as in claim 45, wherein said monitoring means automatically obtains said facility address of said licensee from said facility without any input from said licensee.

47. A system as in claim 45, wherein:

said monitoring means incorporates a product model number in said request datagrams; and

said controlling means separately counts request datagrams for each product model number for each licensee.

48. A method for monitoring the use of a licensed product comprising the steps of:

generating, at regular time intervals, datagrams including an address in a communications facility, said facility address identifying a licensee; and

automatically sending said datagrams from at least one licensee's site over said communications facility to a licensor's site while said licensed product is in use.

49. A method as in claim 48 further wherein:

said generating step includes the step of incorporating a model number of said product in said datagrams.

50. A method as in claim 48, wherein said generating step includes the step of automatically

obtaining said facility address that identifies said licensee from said communications facility without any data being provided by said licensee.

51. A method for controlling use of a licensed product comprising the steps of:

generating a request datagram including a facility address that identifies a licensee in a communications facility;

automatically sending said request datagram from a licensee's site over said communications facility to a licensor's site while said licensed product is in use; and

receiving a reply authorizing datagram at said licensee's site and denying the use of said product when no reply authorizing datagram is received.

52. A method as in claim 51 wherein:

said generating step includes the step of incorporating a model number of said product in said datagram.

53. A method as in claim 51, wherein said generating step includes the step of automatically obtaining said facility address that identifies said licensee from said communications facility without any data being provided by said licensee.

54. A method as in claim 51, wherein:

said reply datagram is one of at least a reply authorization datagram and a reply denial datagram; and

said step of automatically sending said request datagram from a licensee's site includes a step of resending said request datagram if neither a reply authorizing datagram nor a reply denial datagram is received within a predetermined time from sending said request datagram from said licensee's site.

55. A method as in claim 51, wherein said step of automatically sending said request datagram from said licensee's site includes the step of sending a request datagram at regular time intervals.

56. A method as in claim 51, wherein:

said generating step includes the step of providing a datagram identification code within said datagram; and

said reply receiving step rejects said reply authorizing datagram if the datagram identification code included in said reply authorizing datagram does not match the datagram identification code included in said request datagram.

57. A method as in claim 51, wherein:

said generating step includes the step of incorporating in said datagram data indicative of the number of processes currently using said product at said licensee's site.

58. A method as in claim 51, further comprising the steps of:

receiving a reply denial datagram; and

displaying, at a licensee's site, a reason for denial when said reply denial datagram is received.

59. A method as in claim 51, wherein:

said licensed product comprises an executable portion and a data portion; and

said method further comprises a step of controlling use of said data portion with said executable portion.

60. A method as in claim 51 further comprising a step of allowing use of said licensed product before a reply datagram is received.

61. A system for controlling a licensed product comprising:

a communications facility to which at least one licensee is connected;

monitoring means, connected to said communications facility at a site of each said licensee, for generating a request datagram including an address of said licensee on said communications facility and transmitting said request datagram over said communications facility, and for receiving and processing a reply authorizing datagram; and

means for denying use of said product when no reply authorizing datagram is received.

62. A system as in claim 61, wherein:

said monitoring means sends request datagrams at regular time intervals during use of said licensed product.

63. A system as in claim 61 wherein:

said monitoring means incorporates a model number for said product in said request datagram.

64. A system as in claim 61, wherein said monitoring means automatically obtains said facility address of said licensee from said communications facility without any input from said licensee.

65. A system as in claim 61, wherein:

said monitoring means resends said request datagram if no reply authorizing datagram and no reply denial datagram is received within a predetermined period of time after said requesting datagram is sent.

66. A system as in claim 61, wherein said monitoring means transmits request datagrams at predetermined time intervals.

67. A system as in claim 61, wherein:

said monitoring means incorporates a unique identification code in said request datagram; and

said monitoring means rejects any reply authorizing datagram which does not include the same identification code as included in said request datagram.

68. A system as in claim 61, wherein:

said monitoring means includes in said request datagram data indicative of the number of processes, at a licensee's site, currently using said product.

69. A system as in claim 61, wherein:

said monitoring means denies use of said licensed product when no reply authorizing datagram and no reply denial datagram is received within a predetermined time from the sending of said request datagram.

70. A system as in claim 61, further comprising means for indicating, at a licensee's site, a reason for denial when a reply denial datagram is received.

71. A system as in claim 61, wherein:

said licensed product comprises an executable portion and a data portion; and

said system further comprises means for controlling use of said data portion with said executable portion.

72. A system as in claim 71, wherein said data portion controlling means is disposed within said executable portion.

73. A system as in claim 71, wherein said data portion controlling means comprises a first partial controlling means disposed within said executable portion and a second partial controlling means disposed within said monitoring means.

74. A system as in claim 61, wherein said monitoring means includes means for permitting use of said licensed product before a reply datagram is received.

75. A system for monitoring a licensed product comprising:

a communications facility to which at least one licensee is connected;

monitoring means, connected to said communications facility at a site of each said licensee, for generating datagrams including an address of said licensee on said communications facility and transmitting said datagrams at periodic intervals over said communications facility.

76. A system as in claim 75, wherein said monitoring means automatically obtains said communications facility address of said licensee from said communications facility without any input from said licensee.

77. A system as in claim 75, wherein:

said monitoring means incorporates a product model number in said request datagrams.

78. A method for monitoring the use of a licensed product comprising the steps of:

receiving datagrams at a licensor's site on a communications facility having at least one licensee's site thereon, said datagrams being generated at regular time intervals and including a facility address that identifies a licensee in said communications facility;

storing an indication of receipt of each of said datagrams; and

counting said datagrams as an indication of the use of said licensed product.

79. A method as in claim 78 further wherein:

said datagrams include a model number of each product; and

said counting step includes the step of separately counting datagrams for each product model number for each licensee.

80. A method for controlling use of a licensed product comprising the steps of:

receiving a request datagram at a licensor's site on a communications facility having at least one licensee's site thereon, said request datagram including a facility address identifying a licensee and being automatically sent over said communications facility to said licensor's site while said licensed product is in use;

comparing said received request datagram with rules and license data at said licensor's site to determine if use of said licensed product is authorized; and

sending a reply authorizing datagram if use of said licensed product is approved.

81. A method as in claim 80 wherein:

said datagrams include a model number of said product;

said comparing step includes the step of comparing said rules and license data for a particular model number; and

said sending step includes the step of transmitting said reply datagram for each product model number.

82. A method as in claim 80 further comprising the step of sending a reply denial datagram if use of said licensed product is not approved as determined in said comparing step.

83. A method as in claim 80, wherein:

said datagrams include a datagram identification code; and

said reply datagram sending step includes the step of inserting the same datagram identification code in said reply datagram.

84. A method as in claim 80, wherein:

said comparing step includes the step of comparing said facility address that identifies said licensee with a list of valid licensee addresses to determine if said facility address is a valid address; and

said reply authorizing datagram is not sent if said facility address that identifies said licensee is not valid.

85. A method as in claim 84 further comprising the step of sending a reply denial datagram if said facility address that identifies said licensee is not valid.

86. A method as in claim 80, wherein:

said comparing step includes the step of comparing a license expiration date with a date at which said datagram is received; and

said reply authorizing datagram is not sent if the license expiration date is later than the date at which said datagram is received.

87. A method as in claim 86, further comprising the step of sending a reply denial datagram if the license expiration date is later than the date at which said datagram is received.

88. A method as in claim 80, wherein:

said comparing step includes the step of checking currentness of payments from said license; and

said reply authorizing datagram is not sent if payment is overdue.

89. A method as in claim 88, further comprising the step of sending a reply denial datagram if payment is overdue.

90. A method as in claim 80, wherein:

said datagrams include data indicative of the number of processes currently using said product at said licensee's site;

said comparing step includes the step of comparing a number of processes using said product to an authorized number; and

said reply authorizing datagram is not sent if said number of processes using said product exceeds said authorized number.

91. A method as in claim 90, further comprising the step of sending a reply denial datagram if said

number of processes using said product exceeds said authorized number.

92. A method as in claim 80, wherein said sending step includes the steps of sending said reply authorizing datagram when use of said product is approved and sending a reply denial datagram when use of said product is not approved.

93. A system for controlling a licensed product comprising:

a communications facility to which at least one licensee and a licensor are connected at a licensee's site and at a licensor's site, respectively; and

controlling means, connected to said communications facility at said licensor's site, for: receiving a request datagram, said request datagram including an address of said licensee on said communications facility and being transmitted over said communications facility to a site of said licensor; comparing said request datagram with rules and license data to determine if use of said licensed product is authorized; and sending a reply authorizing datagram to said licensee's site if use of said product is approved.

94. A system as in claim 93, wherein:

said request datagrams are sent at regular time intervals during use of said licensed product; and

said controlling means comprises means for counting said request datagrams received at said

controlling means and means for computing an amount to be billed to said licensee in response to said counting.

95. A system as in claim 93 wherein:

said datagrams include a model number for said product; and

said controlling means comprises means for counting datagrams for each product model number for each licensee, in order to compute an amount to be billed to each licensee.

96. A system as in claim 93, wherein:

said controlling means sends a reply denial datagram to said licensee's site if use of said product is not approved.

97. A system as in claim 93, wherein:

said datagrams include a unique identification code; and

said controlling means incorporates the same request datagram identification code in said reply authorizing datagram.

98. A system as in claim 93, wherein said controlling means compares said facility address of said licensee with a list of valid licensee facility addresses and does not generate a reply authorizing datagram if said facility address of said licensee is not valid.

99. A system as in claim 98, wherein said controlling means sends a reply denial datagram when said facility address is not valid.

100. A system as in claim 93, wherein said controlling means compares an expiration date of a license of said product with a date at which said request datagram is received by said controlling means, and does not generate a reply authorizing datagram, thus denying use of said product, if the license expiration date is earlier than the date at which said request datagram is received.

101. A system as in claim 100, wherein said controlling means sends a reply denial datagram if the license expiration date is earlier than the date at which said request datagram is received.

102. A system as in claim 93, wherein said controlling means generate a reply authorizing datagram, thus denying use of said product, if a payment for the use of said product is overdue.

103. A system as in claim 102, wherein said controlling means sends a reply denial datagram if payment for the use of said product is overdue.

104. A system as in claim 93, wherein:

said datagrams include data indicative of the number of processes, at a licensee's site, currently using said product; and

said controlling means does not generate a reply authorizing datagram, thus denying a use of said

product, if more than a predetermined number of processes using said product are running at the licensee's site.

105. A system as in claim 104, wherein said controlling means sends a reply denial datagram if more than said predetermined number of processes using said product are running at the licensee's site.

106. A system as in claim 93, wherein said controlling means sends a reply denial datagram if use of said product is not approved.

107. A system as in claim 93, wherein:

said licensed product comprises an executable portion and a data portion; and

said system further comprises means for controlling use of said data portion with said executable portion.

108. A system as in claim 107, wherein said data portion controlling means is disposed within said executable portion.

109. A system for monitoring a licensed product comprising:

a communications facility to which at least one licensee and a licensor are connected at a licensee's site and at a licensor's site, respectively; and

control means, connected to said communications facility at a licensor's site, for: receiving request datagrams, said request datagrams

including an address of said licensee on said communications facility and being transmitted at periodic intervals over said communications facility to said licensor's site; storing an indication of receipt of each of said datagrams; and counting said datagrams from each licensee as an indication of the use by the licensee of said licensed product.

110. A system as in claim 110, wherein:

said request datagrams include a product model number; and

said controlling means separately counts request datagrams for each product model number for each licensee.

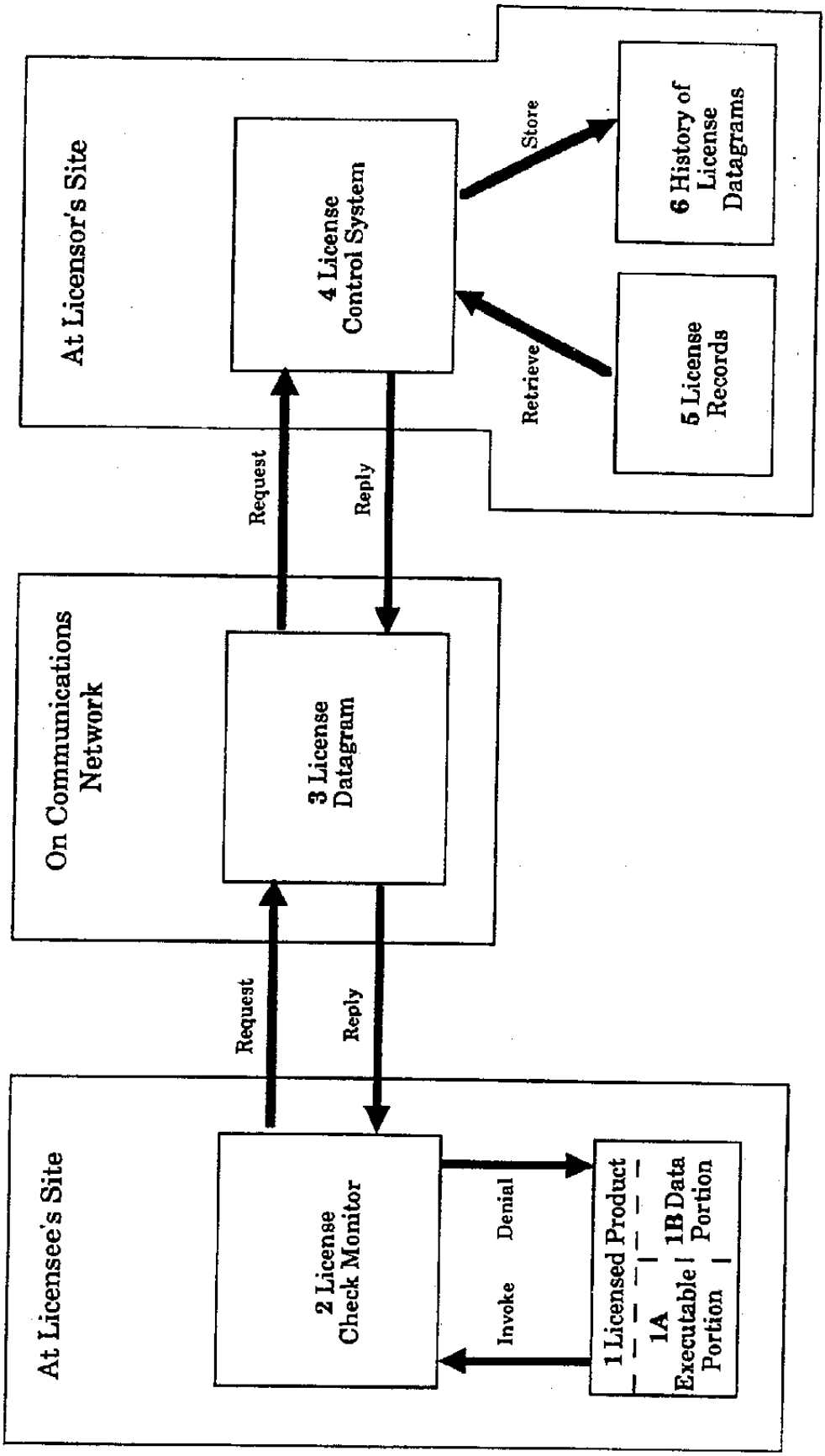

At Licensee's Site
2 License Check Monitor
1 Licensed Product
1A Executable Portion
1B Data Portion
Invoke
Denial
Request
Reply
On Communications Network
3 License Datagram
Request
Reply
At Licensor's Site
4 License Control System
5 License Records
6 History of License Datagrams
Retrieve
Store

FIG. 1

FIG. 2

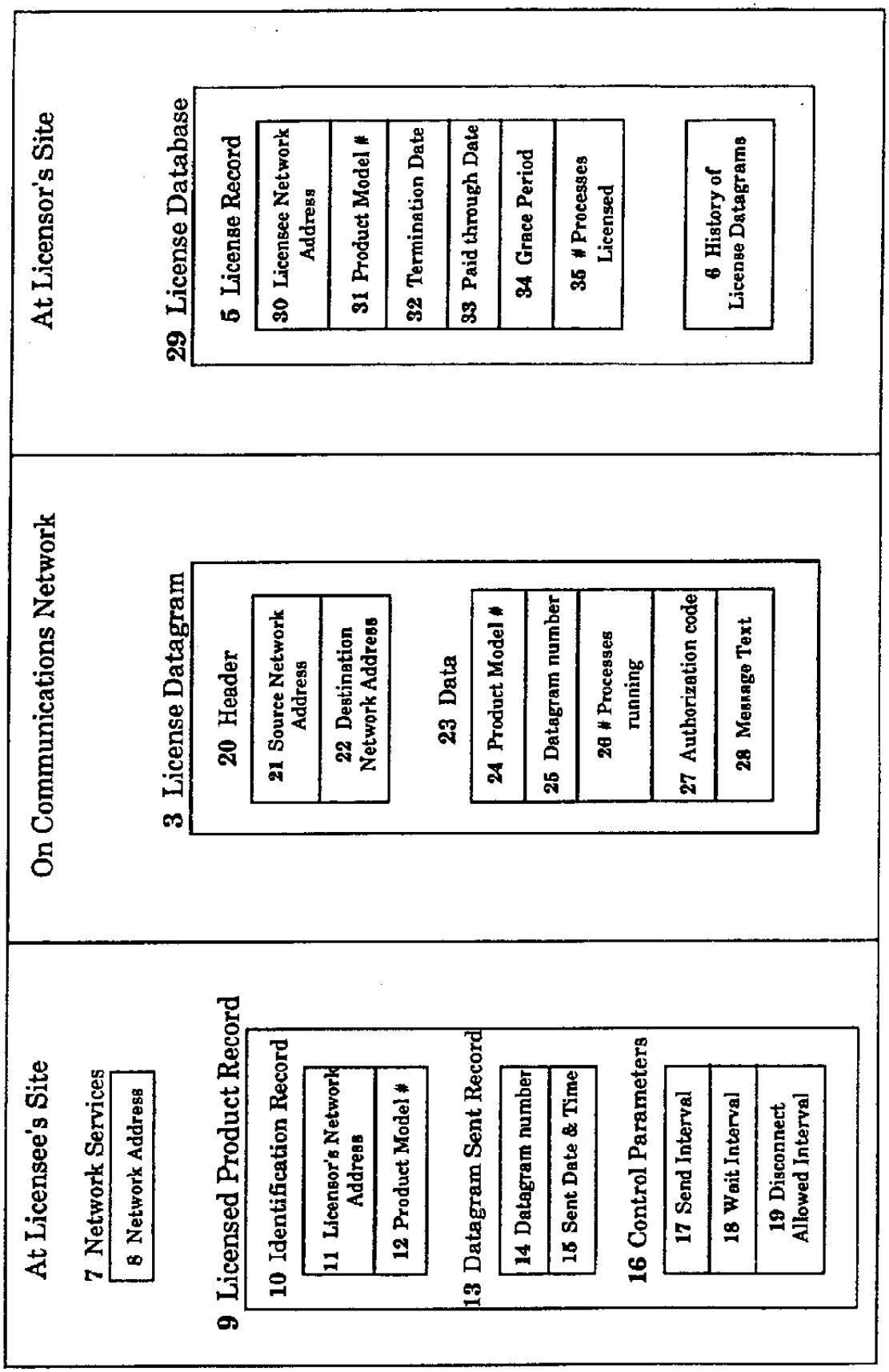
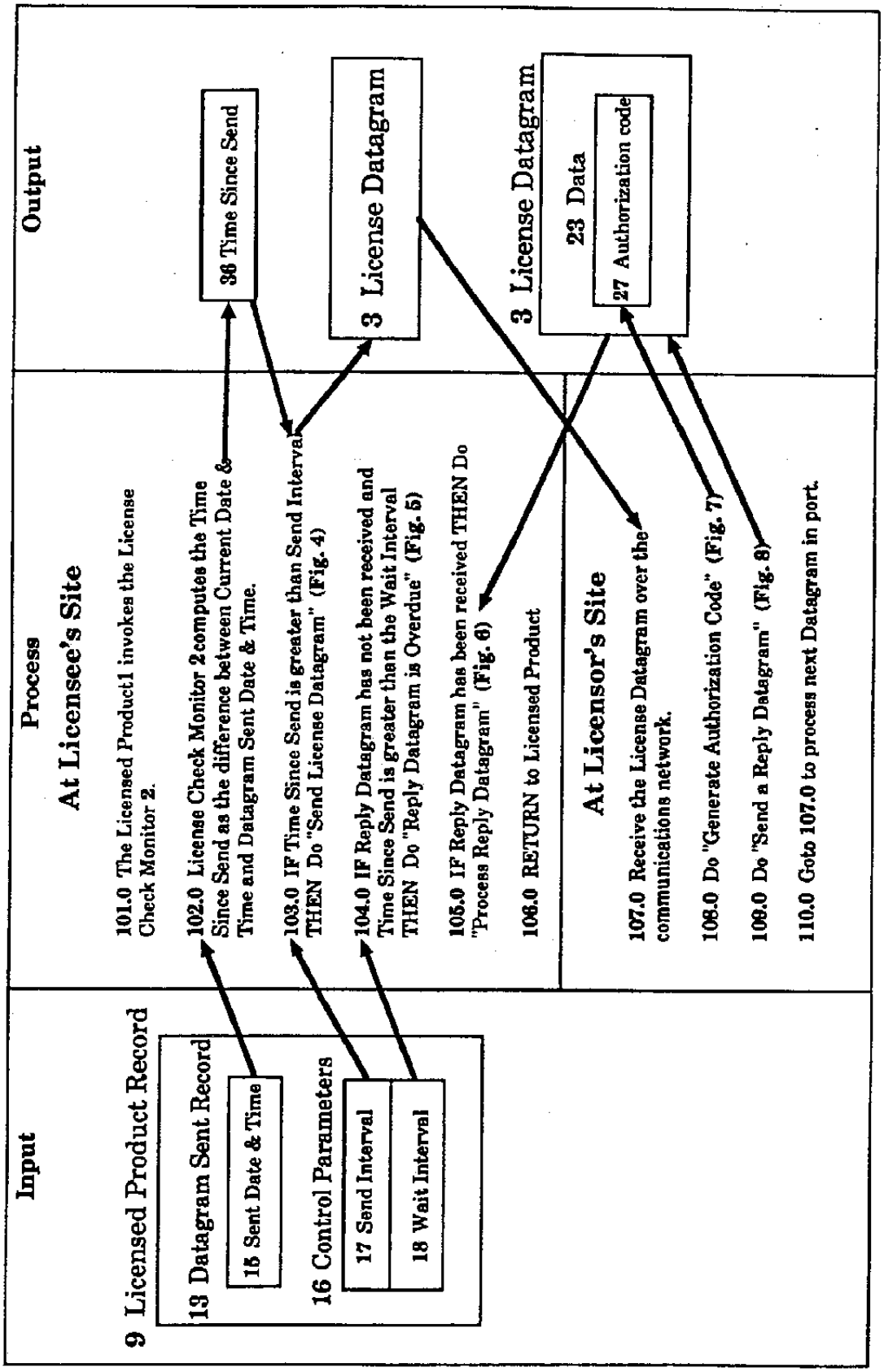
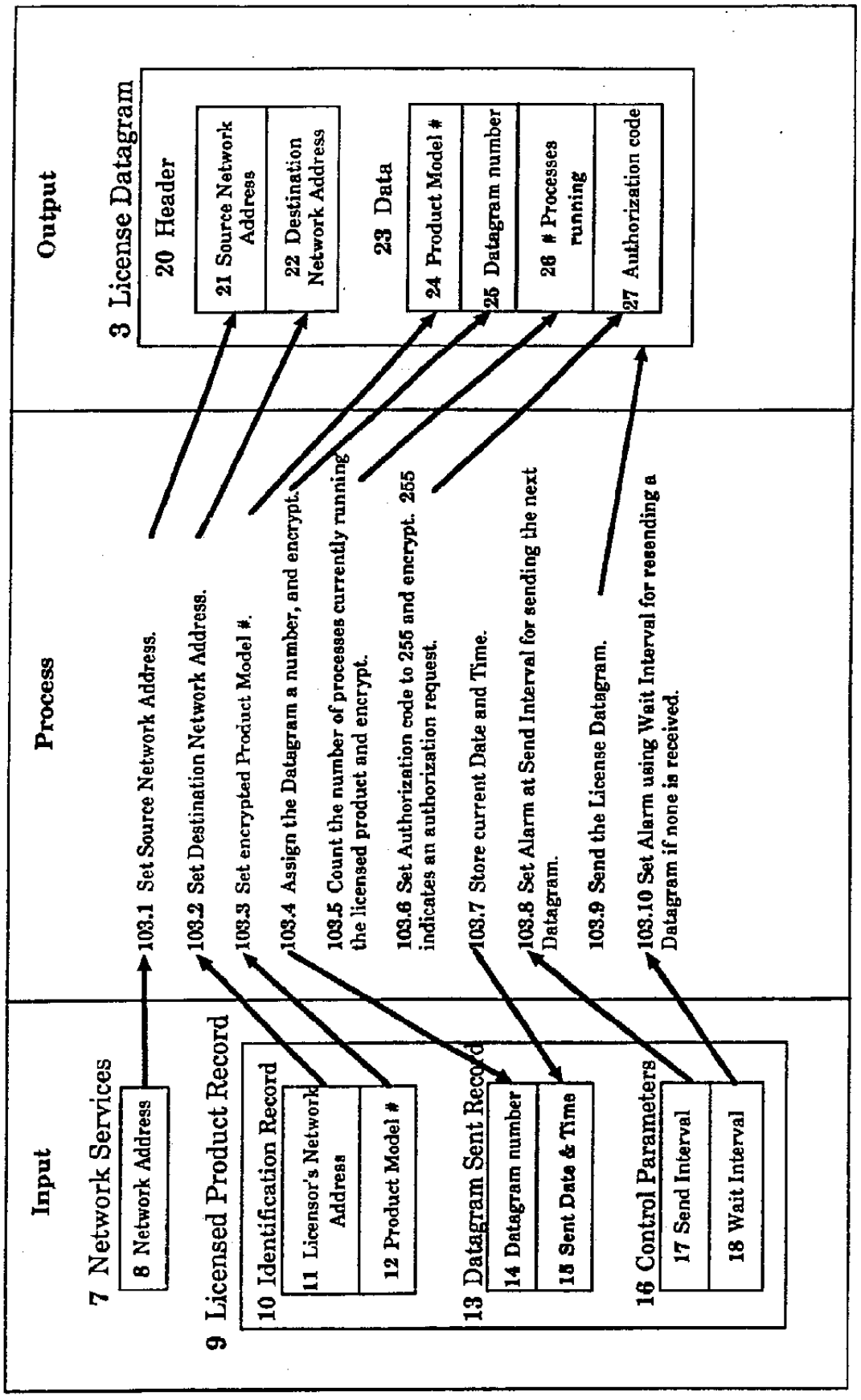

At Licensee's Site
7 Network Services
8 Network Address
9 Licensed Product Record
10 Identification Record
11 Licensor's Network Address
12 Product Model #
13 Datagram Sent Record
14 Datagram number
15 Sent Date & Time
16 Control Parameters
17 Send Interval
18 Wait Interval
19 Disconnect Allowed Interval
On Communications Network
3 License Datagram
20 Header
21 Source Network Address
22 Destination Network Address
23 Data
24 Product Model #
25 Datagram number
26 # Processes running
27 Authorization code
28 Message Text
At Licensor's Site
29 License Database
5 License Record
30 Licensee Network Address
31 Product Model #
32 Termination Date
33 Paid through Date
34 Grace Period
35 # Processes Licensed
6 History of License Datagrams

FIG. 3



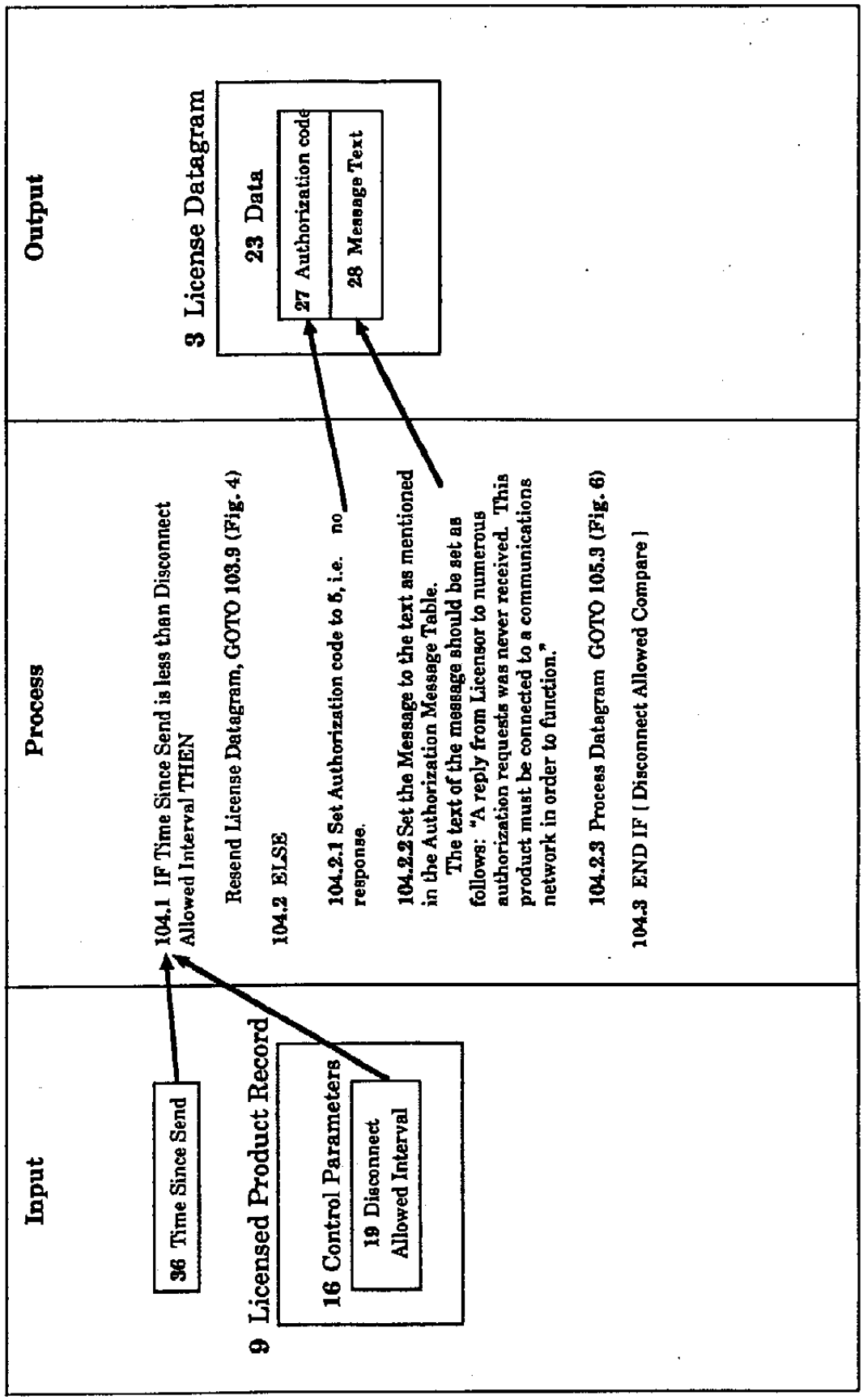
APPLE EXHIBIT 1102, Page 447 of 1048

FIG. 4



APPLE EXHIBIT 1102, Page 448 of 1048

FIG. 5



APPLE EXHIBIT 1102, Page 449 of 1048

FIG. 6

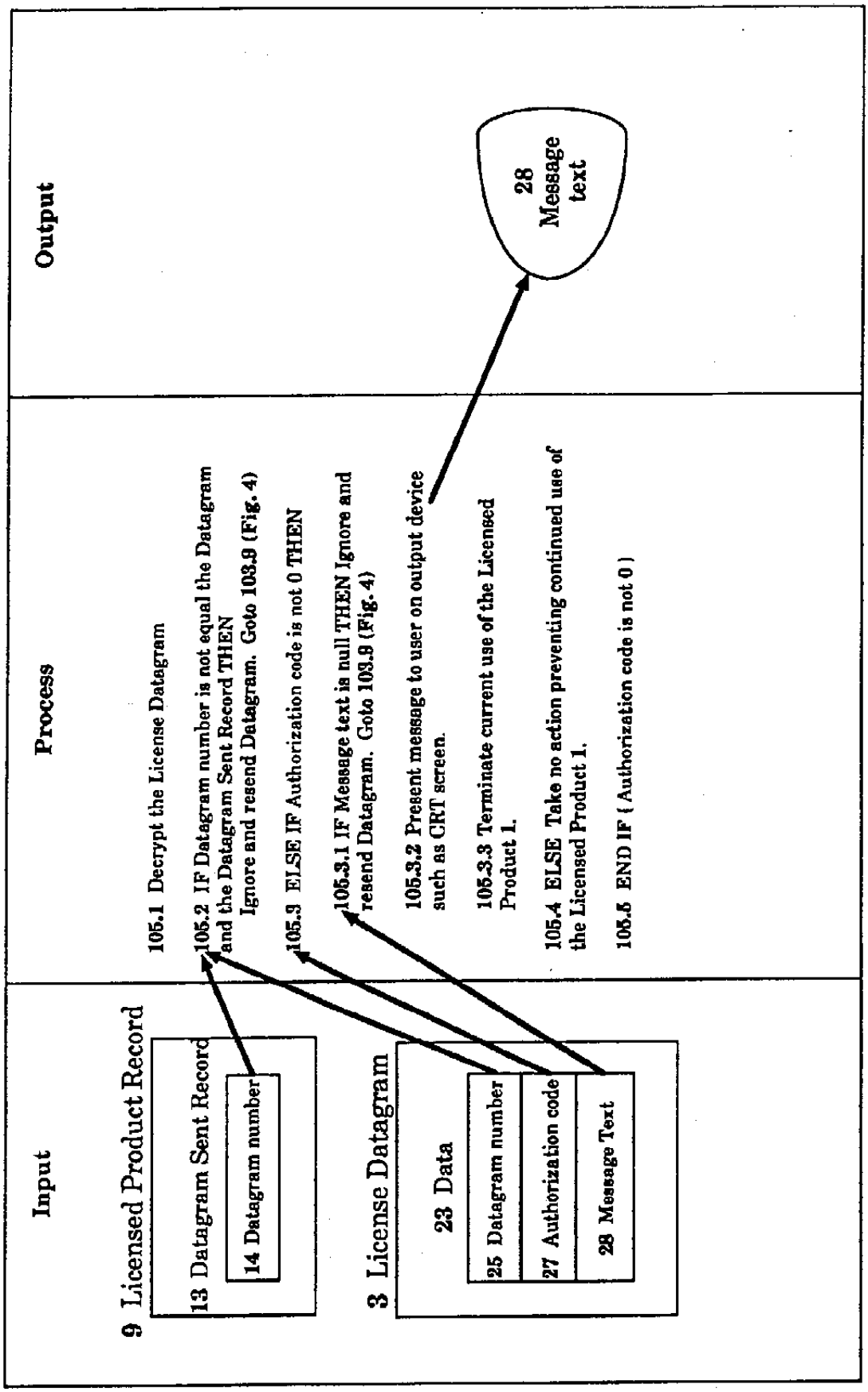
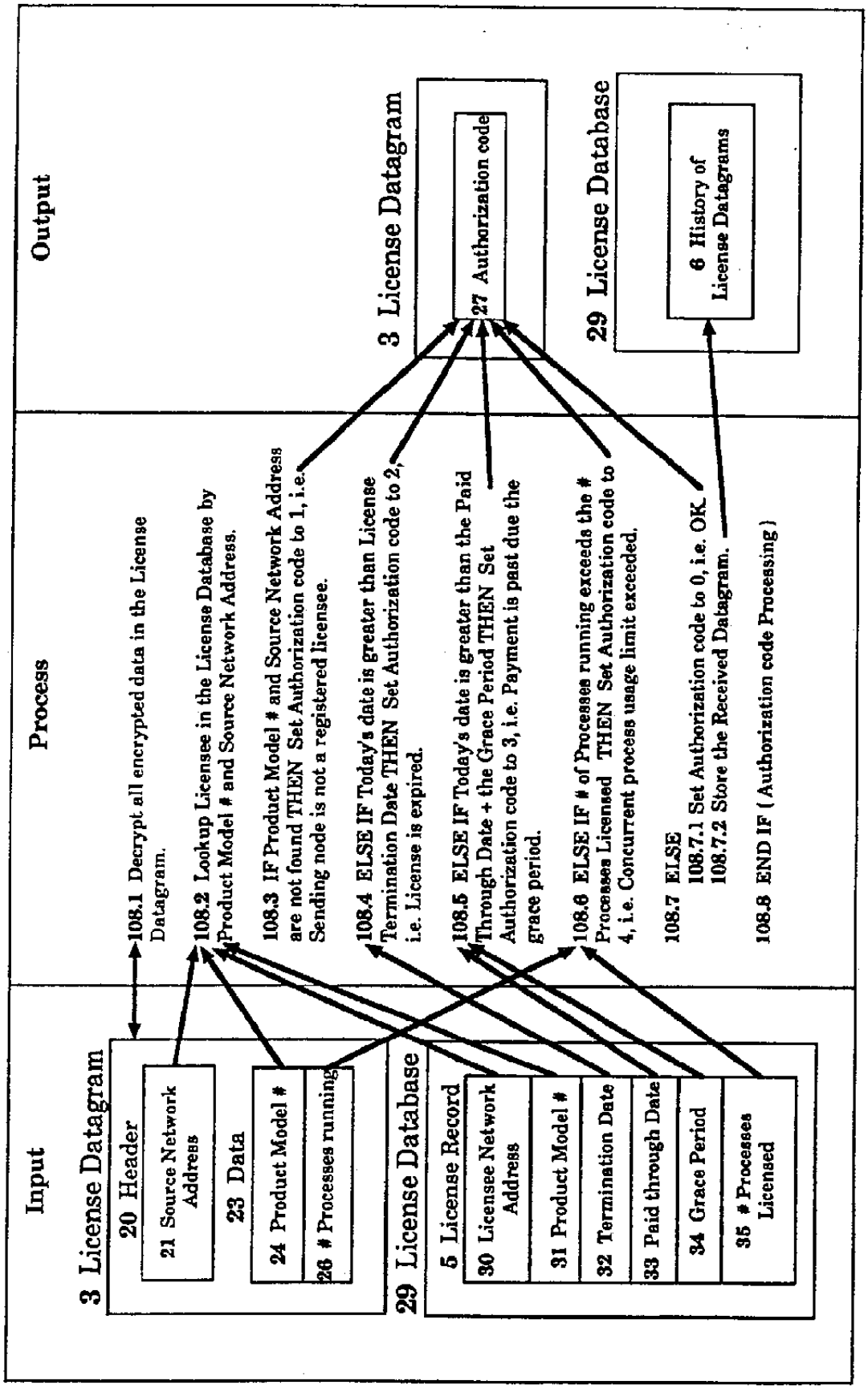
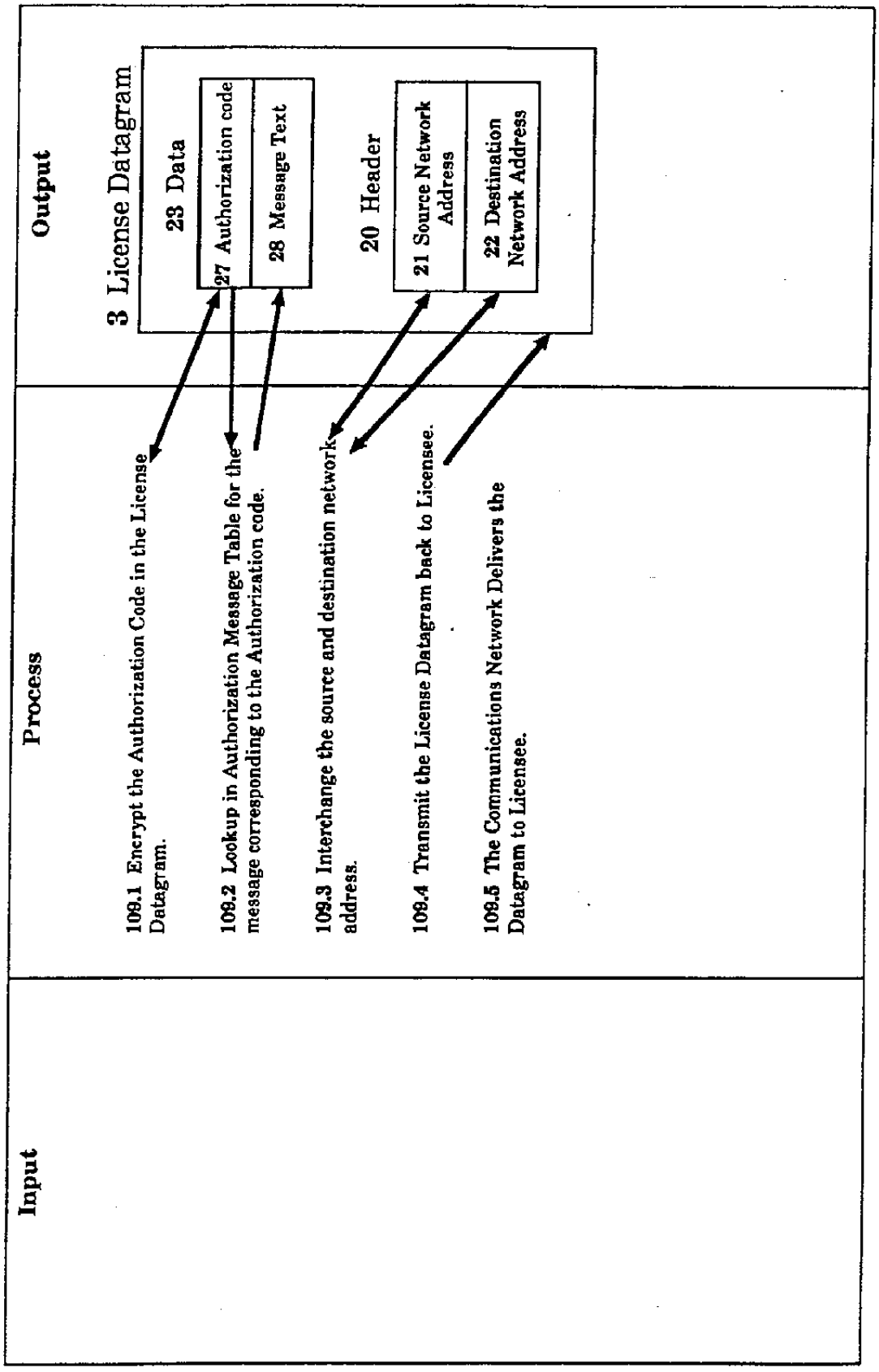

Input
Process
Output
9 Licensed Product Record
13 Datagram Sent Record
14 Datagram number
3 License Datagram
23 Data
25 Datagram number
27 Authorization code
28 Message Text
105.1 Decrypt the License Datagram
105.2 IF Datagram number is not equal the Datagram and the Datagram Sent Record THEN Ignore and resend Datagram. Goto 103.9 (Fig. 4)
105.3 ELSE IF Authorization code is not 0 THEN
105.3.1 IF Message text is null THEN Ignore and resend Datagram. Goto 103.9 (Fig. 4)
105.3.2 Present message to user on output device such as CRT screen.
105.3.3 Terminate current use of the Licensed Product 1.
105.4 ELSE Take no action preventing continued use of the Licensed Product 1.
105.5 END IF (Authorization code is not 0)
28 Message text

FIG. 7



APPLE EXHIBIT 1102, Page 451 of 1048

FIG. 8


Input
Process
Output
3 License Datagram
23 Data
27 Authorization code
28 Message Text
20 Header
21 Source Network Address
22 Destination Network Address
109.1 Encrypt the Authorization Code in the License Datagram.
109.2 Lookup in Authorization Message Table for the message corresponding to the Authorization code.
109.3 Interchange the source and destination network address.
109.4 Transmit the License Datagram back to Licensee.
109.5 The Communications Network Delivers the Datagram to Licensee.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US92/05387

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :G06F 11/34; H04L 9/00
US CL :395/725; 380/4
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/406; 380/25

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

APS DATABASE: Software#, information, usage, monitor?, Licens?

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US,A, 5,103,476 (WAITE ET AL) 07 APRIL 1992 See entire text.	1-110
Y,P	US,A, 5,050,213 (SHEAR) 17 SEPTEMBER 1991 See column 6, lines 27-51.	1-6,9-21,23-26,29-43,45-53,56-59,61-64,67-73,75-110
Y,P	US,A, 5,047,928 (WIEDEMER) 10 SEPTEMBER 1991 See col. 6, lines 16-54.	1-6,9-21,23-36,29-43,45-53,56-59,61-64,67-73,75-110
Y	US,A, 5,023,907 (JOHNSON ET AL) 11 JUNE 1991 See entire document.	1-110

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 05 AUGUST 1992	Date of mailing of the international search report 04 NOV 1992
Name and mailing address of the ISA/ Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer Debbie Williams KENNETH S. KIM Telephone No. (703) 308-1634

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US92/05387

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US,A, 5,014,234 (EDWARDS, JR.) 07 MAY 1991 See col. 3, lines 4-16.	1-110
Y	US,A, 5,010,571 (KATZNELSON) 23 APRIL 1991 See entire document.	1-6,9-21,23-26,29-43,45-53,56-59,61-64,67-73,75-110.
Y	MACMILLAN Publishing Company, 1985, WILLIAM STALINGS, Data and Computer Communications. p199-203.	1-110
Y,P	US,A, 5,113,519 (JOHNSON ET AL) 12 MAY 1992 See col. 6, lines 36-68.	1-110
Y	US,A, 4,937,863 (ROBERT ET AL) 26 JUNE 1990 See col. 3, lines 25-40.	1-6,9-21,23-26,29-43,45-53,56-59,61-64,67-73,75-110

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/00

A1

(11) International Publication Number: **WO 95/35533**

(43) International Publication Date: 28 December 1995 (28.12.95)

(21) International Application Number: PCT/CA95/00354

(22) International Filing Date: 16 June 1995 (16.06.95)

(30) Priority Data:
08/261,496 17 June 1994 (17.06.94) US

(71) Applicant: MEGALODE CORPORATION [CA/CA]; 935 Sheldon Court, Burlington, Ontario L7L 5K6 (CA).

(72) Inventors: PENKAVA, Josef; 1005 Upper Gage Avenue, Hamilton, Ontario L8V 4L2 (CA). CLARK, Robert, C.; 185 Seaton Street, Toronto, Ontario M5A 2T5 (CA). SIRBU, Victor; Apartment 7, 358 Clendenan Avenue, Toronto, Ontario M6P 2X4 (CA). LUNDY, Douglas, H.; 2182 Maplewood Drive, Burlington, Ontario L7R 2C5 (CA). CONFORZI, David, J.; 720 Courtland Place, Burlington, Ontario L7R 2M6 (CA). MAXWELL, W., Norman; R.R.#3, Puslinch, West Flamborough, Ontario N0B 2J0 (CA).

(74) Agent: MACRAE & CO.; Station B, P.O. Box 806, Ottawa, Ontario K1P 5T4 (CA).

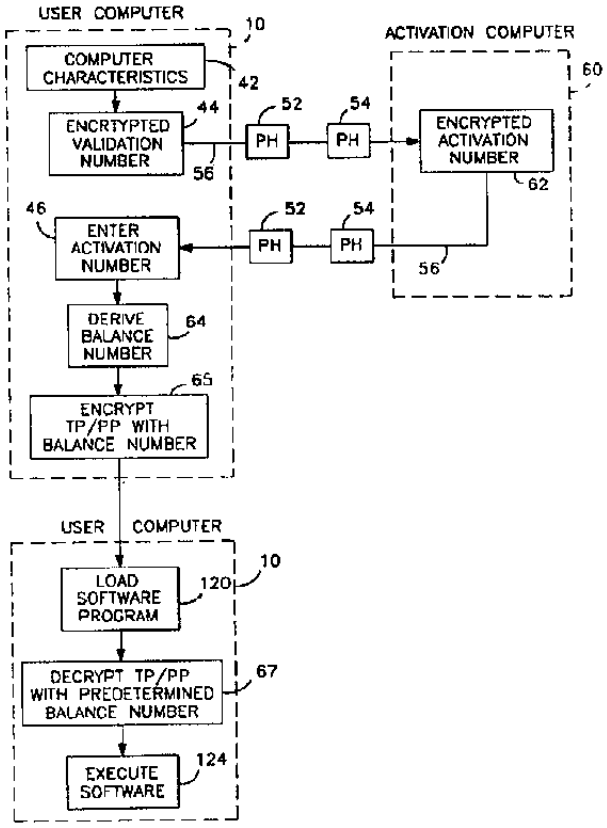
(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TT, UA, UG, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG), ARIPO patent (KE, MW, SD, SZ, UG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD FOR PREVENTING USE OF SOFTWARE ON AN UNAUTHORIZED COMPUTER

(57) Abstract

A method for preventing use of software on an unauthorized computer. The software is programmed to encrypt and output to the user a validation number derived from information received by the software from the computer of one or more computer characteristics providing an unchangeable and unique computer identification. A second computer is operated for the software vendor to encrypt an activation number derived from the validation number and supplied to the user for input to the user's computer. The activation number includes one or more randomly-generated digits which, when a predetermined mathematical operation is performed thereon and on at least one of the digits of the validation number, yields a derived balance number. A preselected signature and other information is randomly scattered among randomly generated bytes along with a product identification number as a thumbprint/productprint which is encrypted by the balance number derived by the user's computer from the validation and activation numbers and which is on the hard disk drive of the user's computer. The software is authorized for use in the user's computer if the preselected signature is retrieved after the predetermined balance number is applied to decrypt the information including the preselected signature.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Ausuralia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

METHOD FOR PREVENTING USE OF SOFTWARE ON AN UNAUTHORIZED COMPUTER

The present invention relates generally to the prevention of unauthorized use of software. More particularly, the present invention relates to the preventing of a computer program from being executed on a computer system or computer network, other than one which has been previously authorized.

As computer systems and software have proliferated, the problem of software piracy has also increased. A computer program is typically installed in a computer with a fixed disk or hard drive by transferring the program from a floppy disk or CD-ROM (purchased from a software publisher) to the fixed disk for subsequent use by the computer system. While the program may have originally been legitimately purchased, the purchaser may thereafter make copies for use by the purchaser or others in other computer systems or may simply use the floppy disk to install the software in other computer systems, without permission of the software publisher, thereby depriving the software publisher of the additional revenues of sale of additional software packages to which the publisher is entitled. Although back-up copies of software are normally considered desirable, it is also desirable for the financial health of the software industry that such "piracy" be stopped.

Various techniques have been proposed for elimination or reduction of software piracy. Many of these techniques are described in U.S. patent 5,113,518 to Durst, Jr. et al, which is incorporated herein by reference. Unfortunately, these techniques have met with only limited success as procedures have been found by the "pirates" or "hackers" for circumventing these

techniques. Some of these techniques may also be so inconvenient as to deter the customer from purchasing the software. For example, one such technique utilizes hardware in the form of a device called a "dongle" which is connected to the computer, and the software must confirm the presence of this device by means of a coded response before it can be activated. Such a device is described in U.S. patents 4,446,519; 4,562,306; 4,685,055; and 5,182,770. However, this undesirably requires the purchase by the customer of such a device for each new software package which is purchased. Further, this technique can be defeated by discovery the correct coded response and providing it through a modification of the program.

As pointed out in Durst, Jr. et al, the problem is not so much in the act of copying the software package as it is in the use of copies of the software on various computer systems without compensating the publisher for the right to use those copies. It is therefore a primary object of the present invention to prevent a computer program from being executed on an unauthorized computer system.

Durst, Jr. et al discloses a technique for preventing a computer program from being used by a computer system other than a designated system. The values of certain characteristics exhibited by the designated computer system first are stored, and then the values of those same characteristics exhibited by the computer system which is intended to use the computer program are measured and compared to the stored values. If the compared values are substantially the same, the computer program may be executed. However, if they are different, the computer system which was intended to use the program is inhibited from executing that program. These characteristics are disclosed to be one or more, and preferably at least two, of the

following: an identification of the processor included in the computer system, the clock speed of the clock generator included in that system, an identification of the ROM normally provided with the processor, the wait time assigned to the processor for accessing a RAM, the actual rotary speed of a disk drive normally provided with the computer system, the access speed of that disk drive, and the sector interleave value of that disk drive.

Inherent characteristics such as proposed in Durst, Jr. et al have a tendency to have no current uniqueness (although perhaps unique at one time, standardization may have resulted in non-uniqueness, for example, there is now a standard disk drive speed), or the characteristics may change over time undesirably making the software unavailable on the computer on which it is originally installed. Furthermore, the values of the characteristics are stored in the software to be compared with the values of characteristics of a computer on which the software is to be used. This has the disadvantage of being easy to circumvent since the values are stored in a known location in all programs, thus being accessible to every level of programmer.

U.S. patent 4,740,890 to William discloses the use of a remote computer to provide unlocking codes derived from master lists or algorithms. In the field of radio-frequency transmission of data, data security has been maintained by the use of coded transmission utilizing a pair of numbers wherein a plurality of randomly-generated digits in one number has a mathematical relationship to the other number so as to yield a prime number for coding the transmission, and the same prime number is used for decoding the transmission.

As discussed in U.S. patent 4,319,079 to Best, various encryption systems have been developed to

provide data security within data processing systems. However, computer-aided techniques for breaking codes are becoming more sophisticated.

Techniques have also been proposed for activating software remotely. For example, U.S. patent 5,222,134 to Waite et al discloses such a technique wherein a computer is provided with a registration shell, and a data link is established between the computer and a registration computer. By providing the registration computer with various information, a potential licensee can register to utilize the program. Once the registration process is complete, a tamper-proof overlay program is constructed at the registration computer and transferred to the user's computer. The overlay includes critical portions of the main program, without which the main program would not operate. This process undesirably requires a modem on the user's computer.

U.S. patent 5,199,066 to Logan, which is incorporated herein by reference, discloses a method and system for protecting a software program recorded within a storage medium for use with or transmission to computer or processor based hardware. A hardware code uniquely associated with the particular hardware and a first software code uniquely associated with the particular embodiment of the software are inputted. The hardware code is stated to be the numeric serial number of the hardware upon which the program is to operate. It is further stated that, in the case of some computers and some storage media, the program may have the ability to recall or otherwise obtain and input the software serial number and possibly the hardware serial number without any specific action by the user. A first predetermined operation is performed upon the hardware code and the first software code to produce an intermediate code. A unique activation code obtained

from the software supplier is inputted and a second predetermined operation is performed upon the intermediate code and the activation code to produce a second intermediate code. The second intermediate code is compared to a second software code uniquely associated with the particular embodiment of the software and stored in a hidden location within the software. The use of the software is enabled only if the second intermediate code and the second software code are identical. By a formula, the hidden software code changes each time the software is copied, for example, by the addition of 7 each time. To obtain the activation number, the user must provide to the software supplier the serial numbers of the hardware and the software and the number of copies which have been made. The software supplier may have a "hot line" phone to permit the user to obtain the activation code.

The Logan method relies on hiding a software code in a hidden location within the software. Thus, this code undesirably is within access by the user to allow formulation of an activation code (without making a telephone call to legitimately obtain it) and subsequent installation of the software to be achievable if the hidden code is located and the process then reverse-engineered. The hardware code which is used to generate the activation code is actually whatever number is inputted by the user and given to the supplier by the user. Since the software is not required to confirm that the activation number is based on the serial number of the specific computer to be authorized, the activation code which is supplied will allow installation of the software on any computer.

Various other techniques for preventing unauthorized software use are disclosed in U.S. patents 4,829,296; 4,866,769; 4,593,353; 4,683,553; 4,796,220; 5,263,157; 5,287,408; 5,311,591; and 5,293,422.

The above software-protection techniques can either be circumvented or rely on codes that can be broken or are so inconvenient to the customer that the competitive position of the software publisher suffers.

It is accordingly an object of the present invention to provide for authorization of a particular computer system or network by means of a technique for uniquely identifying the computer system or network so that the identification doesn't change over time whereby the software does not become unavailable on the authorized computer system or network.

It is a further object of the present invention to provide an easy and convenient means for activation of a software package on a particular computer system or network by a customer.

It is yet another object of the present invention to provide activation of a software package on a particular computer or computer network by means which cannot be discovered and copied by even computer-aided reverse engineering.

It is a still further object of the present invention to provide activation of a software package on a particular computer inexpensively and quickly.

In accordance with the present invention, a method is provided for preventing use of software on an unauthorized computer wherein the software is programmed to generate and output to the user of a computer a first or validation number derived from one or more of the following computer characteristics: serial number of the hard disk, the BIOS data from ROM, the number of sectors per track of the hard disk, the number of heads of the hard disk, and the number of cylinders of the hard disk. A second or activation number derived from the first number is encrypted by operation of a second computer at a remote location inaccessible to the user for input to the user's computer to allow use of the

software on the user's computer. The second number includes one or more randomly generated digits which, when a predetermined mathematical operation is performed thereon and on at least one of the digits of the validation number, yields a derived balance number. This derived balance number is used in the user's computer to encrypt a thumbprint of the computer characteristic including a preselected signature and a productprint. When the computer program is to be executed, the software decodes the thumbprint and productprint using a predetermined balance number. If the derived balance number is equal to the predetermined balance number, the program will execute. Otherwise, it will not execute.

The above and other objects, features, and advantages of the present invention will be apparent in the following detailed description of the preferred embodiments thereof taken in conjunction with the accompanying drawings wherein the same reference numerals denote the same or similar parts throughout the several views.

Brief Description of the Drawings

Fig. 1 is a perspective view of a personal computer and a floppy disk within which is stored a computer program wherein the computer is to be authorized for use of the program therein in accordance with the present invention.

Fig. 2 is a generally diagrammatic view illustrating the hard disk drive therefor.

Fig. 3 is a generally diagrammatic view illustrating the software activation process which embodies the present invention.

Fig. 4 is a block diagram of the process.

Fig. 5 is a flow diagram therefor.

Fig. 6 is a flow diagram of a process for generating a validation number therefor.

Fig. 7 is a flow diagram of a process for generating from the validation number an activation number.

Fig. 8 is a flow diagram for execution of the software.

Fig. 9 is a flow diagram of the entering of the activation number by the user.

Fig. 10 is a flow diagram of generation of a thumbprint in the computer.

Fig. 11 is a diagrammatic view illustrating the thumbprint format in the computer.

Fig. 12 is a view similar to that of Fig. 11 illustrating the productprint format in the computer.

Fig. 13 is a diagrammatic view of the thumbprint/productprint areas illustrating scrambling of the thumbprint.

Fig. 14 is an enlarged view of the thumbprint area of Fig. 13.

Fig. 15 is a view similar to that of Fig. 3 illustrating an alternative embodiment to the present invention.

Fig. 16 is a flow diagram similar to that of Fig. 6 illustrating an alternative method of generating the validation number.

Fig. 17 is a flow diagram similar to that of Fig. 7 illustrating an alternative method of generating the activation number.

Detailed Description of the Preferred Embodiments

Referring to the drawings, there is shown in Fig. 1 a typical personal computer 10 of a type well known in the art and commercially available from a variety of manufacturers, for example, IBM Corporation. The personal computer 10 includes a standard keyboard

12, a standard cathode ray tube (CRT) or screen 14, and a pair of floppy disk drives 16. The keyboard 12 is employed to facilitate communication between an individual user, illustrated at 40 in Fig. 3, and the computer 10 in a manner which is generally well known in the computer art. The CRT 14 also functions in a manner well known in the computer art for displaying information inputted through the keyboard 12 as well as information outputted by the inner workings of the computer 10. The disk drives 16 are employed in a manner well known in the computer art for receiving one or more floppy disks to facilitate the loading or entry of computer software or programs stored within a floppy disk into the computer 10. A typical floppy disk 18 is illustrated in Fig. 1. As used herein, the terms, "program," "computer program," "software" and "software program" are interchangeably used to mean a series of instructions which are used to control the operation of computer hardware or other computer-based or process-based hardware. The reference numeral 18 will be used herein to refer interchangeably to the floppy disk as well as the program contained thereon.

While in the present description of a preferred embodiment of the invention, a personal computer 10 is shown and described, it will be appreciated by those skilled in the art that the present invention may be employed in conjunction with any other type of computer, including standard computers such as a microcomputer, a mini-computer, a main-frame computer, a computer network, and/or special purpose computers. In addition, the present invention may be employed in connection with any other type of computer or processor-based hardware such as computer or processor controlled machinery or equipment. By "computer network" is meant a plurality of computers which

communicate via a client server, peer-to-peer, or the like.

Likewise, while in connection with the description of the presently preferred embodiment, the computer program or software is illustrated as being stored within a floppy disk 18, it will be appreciated by those skilled in the art that the program or software could alternatively be stored in any other type of storage medium, for example, a different magnetic medium, such as a CD-ROM drive, a hard disk drive, magnetic tape, etc.; a semiconductor based storage medium, such as a random access memory (RAM), a read only memory (ROM), a programmable read only memory (PROM), etc.; or a nontraditional storage medium, such as a digital audio or video tape or disk or network of storage devices. Accordingly, it should be clearly understood that the present invention is not limited to the particular computer hardware 10 or storage medium 18 used to illustrate the preferred embodiment of the invention.

Referring to Fig. 2, there is illustrated at 20 a fixed or hard disk drive for computer 10 which includes a multiplicity of platters 22 rotatable about a hub 24. Each platter 22 contains a plurality of concentric circular tracks 26 each containing a plurality of sectors 28 used for storage of digital information. Although there are physically fewer sectors 28 in the tracks 26 closer to the hub 24, the hard drive controller, illustrated at 32, manages the space so that, as seen by the computer 10, there are on average typically 17 sectors 28 per track 26. Each platter 22 is two-sided and has on each side a read/write head 30 which magnetically stores onto and reads digital information from the platter 22.

For a track 26, there is a similarly situated track on the opposite side of its platter 22 and on each

of the sides of the other platters, which multiplicity of tracks together is defined herein as a cylinder, illustrated at 34. A cylinder 34 is a logical ordering so that the controller 32 can simultaneously write to both sides of each of a multiplicity of platters 22.

Referring to Figs. 3 and 4, in accordance with the present invention, when a purchaser 40 of a publisher's software package 18 wishes to use the software on the computer 10, the software requires that it first be authorized. The software 18 is embedded with a program which prevents use of the software (or copies thereof) on a computer unless authorization is obtained for use on the particular computer. In a computer network, a maximum number of concurrent users may be authorized for use of the software, as described hereinafter.

The program 18 encrypts from one or more computer characteristics, as indicated at 42, a first or validation number, as indicated at 44, which appears on the computer screen along with instructions for obtaining a second or activation number for inputting to the computer 10, as indicated at 46, for executing the software 18, as indicated at 48.

In order that there be minimal inconvenience to the user 40, he or she is preferably instructed to call an "800" or the like phone number at an activation center, illustrated at 61, at another location (remote location) which is provided as a service to the publisher of the software 18. Thus, phones 52 and 54 respectively are used to orally communicate the validation number (and other information to be described hereinafter) over phone line 56 to the activation center operator 50 who then inputs via keyboard 58 the validation number to a second computer 60, which may be similar to computer 10 or another suitable conventional computer. This number is then used by the program 63 in

computer 60 to generate and encrypt an activation number, as indicated at 62. The reference numeral 63 refers to a hard disk drive in computer 60 as well as a program stored thereon. The activation number is generated to be related to the validation number so that a number, herein called a "derived balance number," may be derived therefrom, as hereinafter discussed. The activation number is then provided by the operator 50 to the user 40 over phone line 56, who then inputs it to computer 10 by means of keyboard 12. The software program 18 then utilizes the validation and activation numbers, as indicated at 64, to obtain the derived balance number. If the validation and activation numbers have been correctly generated and inputted to the user's computer, the derived balance number will be equal to a predetermined balance number. This derived balance number is then used to encrypt a thumbprint of the computer characteristics including a preselected signature (TP) and a productprint (PP), as indicated at 65. For the software to be executed, as indicated at 124, the program is loaded to the hard disk 20, as indicated at 120, and the thumbprint and productprint are decrypted using the predetermined balance number, as indicated at 67. It is envisioned that, with CD-ROM or some other medium, the software program may not be loaded to the hard disk. If the preselected signature is retrieved, the program 18 proceeds with execution of the software, as indicated at 48.

As used herein and in the claims, a "predetermined balance number" is a number which is embedded in the software 18 or otherwise provided to decrypt the preselected signature, and a "derived balance number" is a number which is derived mathematically from the validation and activation numbers for encrypting the signature. As used herein and in the claims, a "signature" or "preselected

signature" is information in the form of a preselected set of digits or characters which the software 18 is programmed to recognize or locate upon use of a decryption process using the predetermined balance number in order that the software be authorized for use. Therefore, if the derived balance number is the same as the predetermined balance number, the signature will be correctly encrypted and can as a result be decrypted by the predetermined balance number to yield the preselected signature whereby the program may be executed. Otherwise, the preselected signature cannot be found and the program will not execute.

Referring to Fig. 15, there is illustrated an alternative embodiment wherein person-to-person phone communication over telephone line 56 is replaced by modem-to-modem communication. Thus, modems 53 and 55 may be provided for computers 10 and 60 respectively for transmitting and receiving the needed information.

Fig. 5 illustrates in greater detail at 65 the process for activation of the software 18. As illustrated therein, the user 40 begins the process by inserting the diskette or CD-ROM or the like containing the software 18 in the respective drive 16. Alternatively, the user may have previously down-loaded (by modem) an embedded software package from a computer bulletin board service or other electronic distribution service. In this case, the software will be residing on the hard disk drive, awaiting activation. In all cases, the user selects the "activate" or "install" option. The software application code then checks for previous activation of this software package 18 on this particular computer system 10, i.e., is there a valid thumbprint/productprint (TP/PP) for this product. If "yes," the program may proceed with installation or re-installation of the software 18 without a call to the activation center. If "no," a first screen appears

which greets the user 40 in the publisher's name and prompts the user to exit or to proceed with software activation.

If the user elects to proceed with software activation, the application code reads the system characteristics, which will be discussed hereinafter, and a second screen appears showing the publisher's name, product and version, customer identification, and product identification. The user is then requested to enter the publisher's product serial number after which it is validated for transcription errors. The user is requested to have basic demographic information available before making a "1-800," "1-900," "DDD," or the like telephone call to the activation center 61 and is then requested to call the activation center 61.

At the activation center 61, the operator 50 requests the customer's identification number, the product identification number, and published product serial number and displays the customer screen. The operator then receives and enters this information in the activation center computer 60. The last two digits of each of these numbers are check digits, determined in accordance with principles commonly known in the art to which this invention pertains, by means of which the program 63 checks whether the numbers are valid numbers. The operator may then receive and enter demographic information from a new customer or updated demographic information from an existing customer. The program 18 then proceeds to generate from the system characteristics a validation number which then appears on the screen. The operator 50 then requests and enters the validation number in the activation center computer 60, and the program 63 in the activation center computer proceeds to generate an activation number, as described hereinafter. This activation number is then relayed by phone from the operator 50 to the user 40, who then

enters the information in computer 10. As previously discussed, this information may alternatively be transmitted back and forth by modem-to-modem communication. After deriving the balance number, the program 18 then "writes" the product identification, the computer characteristics, and the preselected signature in the form of a thumbprint/productprint (TP/PP) encrypted by the derived balance number, as described hereinafter, to the hard disk drive 20. If the activation number is not a correct number to generate a derived balance number which is the same as the predetermined balance number, then the TP/PP will be encrypted and written using a different number, and the preselected signature will not be found when subsequently applying the decryption process using the predetermined balance number. As a result, future efforts to execute previously authorized computer programs on this computer system will be unsuccessful. The screen will then prompt the user to proceed with installation of the computer program or to exit. If the user selects "proceed", the publisher package installation proceeds, and, when complete, the user system returns to the operating system prompt.

If the system characteristics on which the validation number is based have a tendency to change over time or are not sufficiently unique, as are the characteristics disclosed in the Durst, Jr. et al patent, then authorization of a computer may be unreliable in that the authorization may be lost if the characteristics change or the software may not reliably be prevented from use on an unauthorized computer system. Thus, the characteristics of the computer system on which the validation number is based are chosen to be unique and unchanging so that subsequent program execution on the same computer system is seamless yet attempts to execute the program on a

different computer system will result reliably in the program being prevented from executing without a further authorization from the activation center. A suitable set of computer characteristics (32 bytes), which are available on standard industry hardware by accessing various interrupts and direct read functions in "C" language, using principles commonly known to those of ordinary skill in the art to which this invention pertains, are the serial number of the hard disk 20 (20 bytes), the BIOS data from ROM (read only memory), i.e., the date (MM/DD/YY) the system board for computer 10 was manufactured (8 bytes), and disk information consisting of the number of sectors 28 per track 26 (1 byte), the number of heads 30 (1 byte), and the number of cylinders 34 (2 bytes). It should be understood that the set of characteristics may be less than the above as long as the desired uniqueness is obtained. For example, the serial number of the hard disk 20, which includes a unique manufacturer identification number, may be sufficient. For another example, the combination of the BIOS data and the hard disk information may be sufficient.

Hereinafter, specific processes for generation of the validation and activation numbers, along with examples, will be provided. It should be understood that various variations may be made in these specific processes. Thus, neither the specific process steps nor the examples should be viewed as limiting the present invention but are instead to be taken as exemplary thereof.

Referring to Fig. 6, after the program 18 retrieves internal characteristic information, as indicated at 70, these 32 bytes of information are reduced to 4 internal random bytes (for example, A*!Ø), as indicated at 72, by the conventional technique of a recursive modulus 256 check-sum procedure, a technique

commonly known to those of ordinary skill in the art to which this invention pertains. Each of the four bytes correspond to numbers between 0 and 255, for example, 61, 128, 85, 40. The reduction in the number of bytes is primarily to reduce the volume of information to be transmitted over the phone by the user and operator. However, with modem-to-modem communication, as previously discussed relative to Fig. 15, it may be unnecessary to reduce the 32 bytes to 4 since convenience of the user and operation would no longer be a consideration.

As indicated at 74, 5 check digits are calculated from these four bytes by a conventional weighted technique wherein the summation of the products of the bytes and weighted numbers, using the weighting 2, 3, 4, and 5 respectively, is divided by 10, and the remainder is the first check digit D_1. Thus, D_1=2 as follows:

$$[5(61)+4(128)+3(85)+2(40)]/10$$
$$= 115, \text{ remainder } 2$$

Check digit D_1 is appended to the four bytes, i.e., 61, 128, 85, 40, 2, for calculation of check digit D_2, and the summation of the products of the bytes (with D_1) and numbers 2, 3, 4, 5, and 6 (shifted to the right) respectively is divided again by 10, and the remainder is the second check digit D_2. Thus, D_2=0 as follows:

$$[6(61)+5(128)+4(85)+3(40)+2(2)]/10$$
$$= 147, \text{ remainder } 0$$

The remaining check digits D_3, D_4, and D_5 may be calculated similarly with "shifting to the right" occurring for each check digit. As illustrated at 76, these check digits are placed in an intermediate storage

buffer to await the generation of 5 random digits, as hereinafter discussed.

Meanwhile, as indicated at 78, the program 18 generates the 5 random digits R_1 to R_5. As indicated at 80, these random digits R_1 to R_5 are added respectively to the check digits D_1 to D_5 (and any resulting digit in the 10s column dropped) to obtain a set of digits C_1 to C_5. For example, assuming D_1 to D_5 = 3, 5, 1, 9, 2, and R_1 to R_5 = 8, 3, 6, 2, 5, C_1 to C_5 are calculated as follows:

3	5	1	9	2	D_1 to D_5
8	3	6	2	5	R_1 to R_5
1	8	7	1	7	C_1 to C_5

As indicated at 82, the digits C_1 to C_5 and the random digits R_1 to R_5 are assembled as $R_1 \ldots R_5$, $C_1 \ldots C_5$, i.e.,

8 3 6 2 5 1 8 7 1 7

As indicated at 84, two check digits C_6 and C_7 are calculated similarly as discussed for check digits D_1 to D_5. Thus, C_6 is calculated by summing the products of the 10 digits and 2, 3, 4, 5, 6, 7, 8, 9, 2, 3 respectively and dividing by 10, the remainder being C_6 which, in this example, is 6, as follows:

$$[2(7)+3(1)+4(7)+5(8)+6(1)+7(5)+8(2)+9(6)+2(3)+3(8)]/10 = 22, \text{ remainder } 6.$$

Using the resulting 11 digit number and shifting to the right, check digit C_7 = 8, as follows:

$$[2(6)+3(7)+4(1)+5(7)+6(8)+7(1)+8(5)+9(2)+2(6)+3(3)+4(8)]/10 = 23, \text{ remainder } 8.$$

As indicated at 86, random numbers R_1 to R_5, digits C_1 to C_5, and the check digits C_6 and C_7 are assembled into the validation number $R_1 \ldots R_5$, $C_1 \ldots C_5$, C_6, C_7 which, in the example, is:

Validation no: 8 3 6 2 5 1 8 7 1 7 6 8

Thus, the resulting pseudo-random validation number generated by the program 18 in the user's computer 10 comprises digits which are meaningless to the user and have no meaning relative to the computer characteristics, except that the computer characteristics can be derived therefrom by means of a program which traces backwardly the validation code to the original 32 bytes. Since the process is pseudo-random, the derivation of such a program by a hacker is not envisioned. By re-calculation of check digits C_6 and C_7, the activation computer 60 can confirm that the validation number provided by the user 40 is a correct and not a fabricated or incorrectly given validation number.

Referring to Fig. 16, there is illustrated an alternative method of generating the validation number which allows the authenticity of the customer and product identification and the product serial no. to be checked for relational correctness and whether the information given over the phone corresponds to what is entered in the computer 10. Often, the product identification and product serial numbers are within a range of numbers, permitting a further check on their correctness.

As indicated at 200, the customer and product identification numbers, the publisher's serial number, and the preliminary validation number (including check digits) are first assembled into a number (customer

ID....C_7), the preliminary validation number in this embodiment being defined to be the same as the 12-digit validation number previously discussed. This assembled number is then used to generate from all of the bytes thereof two check digits C_8 and C_9, as indicated at 202, in a manner as previously discussed for generation of check digits. As indicated at 204, the resulting number with these check digits appended (customer ID....C_9) is then summed. Two more check digits C_{10} and C_{11} are then generated based on the sum, as indicated at 206, again using similar principles for check digit generation. The check digits C_8, C_9, C_{10}, and C_{11} are appended to the preliminary validation number to obtain a final validation number (R_1...R_5, C_1....C_{11}), as indicated at 208.

At the activation center, the check digits C_8 to C_{11} will be used to determine if the information given by the user checks, i.e., the activation center will double-check to determine if the user really gave the correct information.

Unless otherwise noted, the term "validation number" will refer in this specification to the 12-digit validation number but may refer in the claims to either validation number or another suitable validation number.

Referring to Fig. 7, there is indicated the process of generation of the nine digit activation number A_1 to A_9 by program 63 in the remote activation computer 60. As indicated at 90, after the validation number is inputted, the check digits C_6 and C_7 are re-calculated and compared with the corresponding digits in the validation number as supplied over the phone by the user to confirm the validation number as a correct one which has not been fabricated or incorrectly given by the user.

As indicated at 92, the sum of the digits of the validation number is calculated, this sum being a

number which is defined herein as "Balance 1." Thus, in this example,

$$\text{Balance 1} = 8+3+6+2+5+1+8+7+1+7+6+8 = 62$$

It should however be understood that Balance 1 may be obtained from the validation number by any other suitable mathematical process.

As indicated at 94, digits A_2, A_6, and A_7 are calculated from the validation number as follows. A_2 is set equal to the unit's value of balance 1, and A_6 is set equal to the ten's value thereof. Thus, in the example, $A_2 = 2$ and $A_6 = 6$. The digits $R_1 \ldots R_5$, $C_1 \ldots C_5$ are multiplied respectively by 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 (hexadecimal weighting, i.e., the digits being multiplied respectively by a set of numbers with each being double the preceding number, beginning with 1), and the summation of the products is divided by 10, the remainder being A_7. Thus, $A_7 = 4$, calculated as follows:

$$[1(8)+2(3)+4(6)+8(2)+16(5)+32(1)+64(8)+128(7)+256(1)+512(7)]/10 = 541, \text{ remainder } 4$$

As indicated at 96, three random digits a, b, and c are generated by the program 63. A number d is calculated as a(b)+c, as indicated at 98. As indicated at 100, if d is greater than or equal to a predetermined balance number, d is subtracted from Balance 1, giving e. Otherwise, d is added to Balance 1, giving e. As indicated at 102, if the result e is not equal to the predetermined balance number, a new set of 3 random digits is generated and steps 96, 98, and 100 re-applied until a set of 3 digits a, b, and c is randomly selected such that e is equal to the predetermined balance number. The number d for those three digits (wherein, e = the predetermined balance number) is defined herein as "Balance 2," and those three digits a, b, and c are set

equal to A_3, A_1, and A_5 respectively, as indicated at 104. It should be understood that Balance 2 may be obtained from the three (or other suitable number) of randomly-generated digits by any other suitable mathematical process. Thus, Balance 2, in this example, is equal to Balance 1 less the predetermined balance number, i.e., Balance 2 = 62 - 5 = 57.

It is preferred that the predetermined balance number be a prime number such as, in the example, 5, since a factorable number is weak mathematically so that the code may be more easily cracked. More preferably, the prime number is a higher number such as a 2, 3, or 4 digit prime number since more digits of information are involved, making any effort to determine the predetermined balance number even more difficult.

With e = 5, a random set of values for a, b, and c may be 7, 8, and 1 respectively whereby A_1, A_3, and A_5 are 8, 7, and 1 respectively. This is because Balance 2 = 7(8) + 1 = 57, and
e = Balance 1 - Balance 2 = 62 - 57 = 5.

As indicated at 106, a determination is made whether Balance 1 is greater than or equal to the predetermined balance number e. If Balance 1 is less than the predetermined balance number, 5, then A_4 is set to a random number of 0 to 4, as indicated at 108. If Balance 1 is greater than or equal to the predetermined balance number, 5, as it is in the example, then A_4 is set to a random number of 5 to 9. For example, A_4 may be set randomly to 6.

As indicated at 112, the digits A_1 to A_7 are then assembled, as follows:

8 2 7 6 1 6 4

Similarly as check digits C_6 and C_7 were calculated for the validation number, check digits A_8 and

A_9 are calculated for the activation number, as indicated at 114, except the multipliers of A_1 to A_7 and then A_1 to A_8 begin with different digits, i.e., 7 and 8 respectively. Thus, in the example, $A_8 = 6$ as follows:

$$[7(4)+8(6)+9(1)+2(6)+3(7)+4(2)+5(8)]/10$$
$$= 16, \text{ remainder } 6$$

$A_9=6$ as follows:

$$[8(6)+9(4)+2(6)+3(1)+4(6)+5(7)+6(2)+7(8)]/10$$
$$= 22, \text{ remainder } 6$$

As illustrated at 116, the activation number A_1 to A_9 is assembled and displayed on the screen to the operator 50, as follows:

8 2 7 6 1 6 4 6 6

This number is delivered over the phone, by modem, or otherwise to the user for inputting to computer 10.

Referring to Fig. 17, there is illustrated an alternative method of generating the activation number, which allows a greater check on the authenticity of the digits thereof. After assembly of the preliminary activation number, as illustrated in Fig. 7 and indicated at 116, this assembled number is then used to generate from all of the bytes thereof two check digits A_{10} and A_{11}, as indicated at 250, in a manner as previously discussed for generation of check digits. As indicated at 252, the resulting number with these check digits appended ($A_1....A_{11}$) is then summed. Two more check digits A_{12} and A_{13} are then generated based on the sum, as indicated at 254, again using similar principles for check digit generation. The check digits A_{10}, A_{11}, A_{12}, and A_{13} are appended to the preliminary activation number to obtain a final activation number, as indicated

at 256. When the final activation number is inputted to the user's computer, the program will utilize these additional check digits to determine if the activation number is a correctly generated number.

Unless otherwise noted, the term "activation number" will refer in this specification to the nine-digit activation number but may refer in the claims to either activation number or another suitable activation number.

As with the validation number, the resulting pseudo-random activation number generated by the program 63 in the remote computer 60 comprises digits which are meaningless to the user and have no meaning relative to the validation number, which is also meaningless to the user. If, however, the user were to successfully generate an activation number which would cause the TP/PP to be encrypted and written, the TP/PP cannot thereafter be decrypted to retrieve the preselected signature for execution of the software unless the random digits were also selected to give a derived balance number which is the same as the predetermined balance number.

The activation number is given over the phone, modem, or the like to the user 40 and inputted to the computer 10 being authorized. The program 18 then generates a derived balance number and causes the customer and product identification, computer system unique characteristics, and the preselected signature to be written on the hard disk drive 20 as the thumbprint/productprint (TP/PP), encrypted by use of the derived balance number, as described hereinafter, preferably in several locations to facilitate data integrity/recovery across all operating systems, i.e., DOS, Windows, OS/2, and the like: (1) one or more locations in the root of the hard disk drive 20, i.e. a non-hidden file in the directory, (2) track 0 of the

first cylinder 34, i.e., a hidden file, and (3) several (perhaps 3) locations on the diagnostic cylinder, i.e. hidden files. By "hidden files" is meant that there is no directory in the system which indicates their existence. The information is also written to several different locations as a back-up, i.e., in case it gets inadvertently deleted at one or more locations. There are situations where it may be desirable to recover the information. For example, during unloading and reloading, the "non-hidden file" and its root directory will be recovered, i.e., rewritten. Therefore, it is desirable that the information be written to the root directory even though it is not a hidden file.

Referring to Fig. 8, for execution of the software package 18, the user "runs" the software, as indicated at 120, and the "executable" code portion thereof checks for whether a valid thumbprint/productprint (TP/PP) exists on the hard disk drive 20, as indicated at 122. If a valid TP/PP has been written to the hard disk drive 20, the software executes, as indicated at 124. A wrapper in each software package may have several "enabling" function calls to the embedded, encrypted "code." However, if a valid TP/PP does not exist, then the software causes the computer screen to display a message prompting the user to insert the activation/installation diskette, CD ROM, or the like medium in order to activate, as indicated at 126. If the medium is inserted, the activation process begins, as indicated at 128. Otherwise, the program terminates and a return to the Operating System occurs.

Referring to Fig. 9, after the activation number A_1 to A_9 is inputted to the computer 10 by the user 40, as indicated at 130, check digits A^1_8 and A^1_9 are calculated (in the same way check digits A_8 and A_9 were calculated) and compared with digits A_8 and A_9, as indicated at 132 and 134 respectively. If $A_8 = A^1_8$ and

$A_9 = A^1_9$, then the program proceeds to a calculation of A^1_2, A^1_6, and A^1_7 (in the same way A_2, A_6, and A_7 were calculated) and a comparison made with A_2, A_6, and A_7, as indicated at 136 and 138 respectively. If $A_2 = A^1_2$, $A_6 = A^1_6$, and $A_7 = A^1_7$, then A_1, A_3, and A_5 are extracted from the activation number, Balance 1 is set equal to the sum of the digits of the validation number, and Balance 2 is set equal to $A_3(A_1) + A_5$, as indicated at 140, 142, and 144 respectively. A_4 is abstracted from the validation number. If A_4 is greater than or equal to 5, then the derived balance number is equal to Balance 1 less Balance 2, as indicated at 146. Otherwise, the derived balance number is equal to the sum of Balance 1 and Balance 2, as indicated at 148.

Referring to Figs. 10 to 14, the derived balance number is used to encrypt and write to the hard disk drive 20 the thumbprint in a thumbprint format, indicated at 209 in Fig. 11, and the productprint (containing the publisher's product identification number in a productprint format), illustrated at 221 in Fig. 12, contained within a cluster of perhaps 4 sectors 28 (2048 bytes), as seen in Fig. 13. The thumbprint 150 is contained within one of the sectors (512 bytes).

As indicated at 160 in Fig. 10, the program first checks for whether a thumbprint 150 exists. If it does, it is then updated for a new productprint, as indicated at 161, and a random number generator is run to determine randomly a "pointer" start position, as indicated at 163. If it doesn't, a thumbprint 150 must be generated. This is done by running a unique random number generator for the "pointer" portion 154 (right side 256 bytes) of the thumbprint area 150, as indicated at 162, running a non-unique random number generator for the "data" portion (left side 256 bytes) of the thumbprint area 150 and the 3 sectors for productprints, as indicated at 164, and running a random number

generator to determine randomly a "pointer" start position, as indicated at 166.

Referring to Fig. 11, the thumbprint is assembled in the area 150 in a format, indicated at 209, of perhaps 35 bytes including (1) the authorizer's signature (16 bytes), illustrated at 226, (2) the customer identification number (4 bytes), illustrated at 210, (3) the number of products for this customer number (2 bytes, based on how many productprints have been written), illustrated at 212, (4) a productprint encryption key (2 bytes, a random number used to encrypt the productprint by a suitable conventional process), illustrated at 214, (5) the 4-byte internal machine characteristic data, illustrated at 216, (6) four pointers (1 byte each), illustrated at 218, used for recovery of the TP/PP in track zero since they identify 4 particular sectors previously allocated by the operating system therefor, and (6) a check sum (3 bytes), i.e., which is derived by the modulus 256 process as previously discussed, illustrated at 220.

Referring to Fig. 12, the productprint, encrypted by encryption key 214 and then XOR'd to reverse bits in accordance with principles commonly known to those of ordinary skill in the art to which this invention pertains, is assembled in the area 152 in a format, illustrated at 221, of perhaps 11 bytes including (1) product identification (2 bytes), illustrated at 222, (2) "try & buy" indicators (3 bytes), illustrated at 223, (3) network indicators (3 bytes), illustrated at 224, and (4) a check sum (3 bytes), illustrated at 225. The "try & buy" and "network" indicators 223 and 224 respectively will be discussed hereinafter. It should be understood that these indicators 223 and 224 are optional and need not be provided if the software package is not to have these features.

In order to determine if a software program is authorized or "maximum" users in a network are exceeded, the embedded software about to execute on a "node" is programmed to communicate back to the client server or peer-to-peer node to obtain authorization prior to executing. Character 6 in the "network" indicator 224 is a "type of network" designator, i.e., perhaps using the characters "N" for Novell, "B" for Banyan, "W" for Windows, "L" for Lantastic, and "A" for "not applicable." Characters 7 and 8 contain the maximum number of users allowed concurrently. If character 6 is "A" or another character indicating that the software contains no provision for network use, then characters 7 and 8 are random digits.

The thumbprint 209 also contains the preselected signature (16 bytes), illustrated at 226, which is a set of characters which are the same for each item of software 18. The preselected signature 226 may be determined randomly or in any other suitable way. For example, the signature may be generated by beginning with 28 and adding 91 (if the sum is greater than 255, then 255 is subtracted to get the number) until the 16 characters are generated. It is this signature which must be retrieved by the program 18 before execution of the software is permitted.

Referring to Figs. 13 and 14, the numbers generated in the pointer portion 154 of 256 bytes are random and unique, i.e., each number appears only once. Thus, in the example of Fig. 12, the first 6 bytes randomly contain unique numbers 56, 1, 14, 255, 48, and 4. A start-point byte is randomly selected, for example, at the third byte, indicated at 158, containing the number 14.

The 35 (or more) bytes of the thumbprint 209 are scrambled or randomly scattered in the "data" portion 156 as controlled by the "pointer" portion 154.

Thus, to assemble the thumbprint data, as indicated at 168, the start-point byte 158 determines the byte-position of the first byte of the thumbprint, i.e., byte number 14 in the "data" portion. The next pointer byte containing number 255 determines the byte-position of the second byte of the thumbprint, i.e., data portion byte number 255. The locations of the remaining thumbprint bytes are determined similarly, and the remaining or unused bytes in the "data" portion retain their randomly-generated numbers.

The program 18 proceeds to decompose the validation and activation numbers and obtain a derived balance number, as previously discussed relative to Fig. 9, which is used to encrypt the TP/PP, as illustrated at 174, by any suitable encryption method. For example, each encrypted byte may be used to encrypt the next byte in a ripple effect.

After the thumbprint is assembled or updated, the productprint data is assembled, as indicated at 170. New check sums are calculated and stored for the TP, PP, and cluster, as indicated at 172, followed by encrypting the PP with the randomly generated number in the TP (then XOR'd) and the TP/PP with the derived balance number, as indicated at 174. It is this encrypted TP/PP which is then written to the hard disk drive 20, as indicated at 176.

For execution of the software 18, the program effects decryption using the predetermined balance number. If the predetermined balance number is the same as the derived balance number (meaning that the validation and activation number set was correctly decomposable to yield a derived balance number which is equal to the predetermined balance number), then the preselected signature 226 as well as the remainder of the TP/PP will be retrieved. If the derived balance number is not the same as the predetermined balance

number, the decryption will not yield the preselected signature 226, and the program 18 will not be executed. To throw a hacker further off guard, the application software is preferably decrypted and re-encrypted on the fly, i.e., as it is being run.

The predetermined balance number is suitably encrypted in object code which is given to the publisher to embed in the program 18, using principles commonly known to those of ordinary skill in the art to which this invention pertains. ^The publisher may not therefore know the balance number. A series of confusing processes are used, in accordance with principles commonly known to those of ordinary skill in the art to which this invention pertains, to deny access to the predetermined balance number to the user or a hacker.

Due to the pseudo-random nature of the validation and activation codes, different validation and activation numbers are obtained each time software is installed on the same computer, and these numbers disappear once the derived balance number is obtained.

In order to allow a potential customer to "try and buy", the software 18 is preferably programmed to allow activation then shut down (or provide a "nagging" message periodically) after a number of uses and/or number of days, as specified in the productprint 223. This would allow software to be, for example, placed on a bulletin board, downloaded, activated, and tried for a period of time, as specified by the publisher, and then be purchased during a brief telephone call to the activation center. The publisher selects the "nag" or "shutdown" version prior to package embedding.

Referring to Fig. 12, character 3 of the "try & buy" indicator 223 is an indication of whether or not the activated package has been purchased. If it has, character 3 may, for example, be a "P" for "purchased."

If it is in "try" mode, character 3 may be a character which indicates either "nag" (continue to operate when the specified number of units of time and "tries" have been used, but a reminder message on a regular basis) or "no nag" (shut down when the specified number of units of time or "tries" have been used). Character 3 also specifies the unit of time, i.e., seconds, minutes, hours, days, or months. Character 4 indicates the number of units of time allowed, and character 5 indicates the number of tries allowed. If character 3 contains a "P," then characters 4 and 5 are random characters. When the user purchases the software, character 3 is changed to the "buy" character.

Referring to Fig. 6, the following is an alternative method for implementing the "network" and "try & buy" features. Instead of listing a check digit, D_5 is selected to provide information relative to which of these features is to be implemented to be passed from the user 40 to the operator 50 (or between the respective computers) encoded within the digit D_5. The software 18 is programmed to check for the "network" and "try & buy" states and select a digit D_5 indicative thereof. The possible states for each feature are "yes" and "no." If the feature is not included as an option for the type of software, it is "inactive." For example, the digit D_5 may be selected as follows:

Network state	Try & Buy state	D_5
No	No	1
No	Yes	2
Yes	No	3
Yes	Yes	4
Inactive	Inactive	0

Thus, D_5, in this embodiment would not be a check digit but would be a digit selected to represent the "network" and "try & buy" states.

When the operator 50 enters this digit D_5 as part of the validation number, the computer 60 is programmed to update its information database 63 appropriately to reflect the user's "network" and/or "try & buy" implementation.

As previously discussed, the pseudo-random encrypting of the validation and activation numbers and the random scattering of the thumbprint/productprint information provides numbers which appear to be meaningless and would not be expected to be decoded by a hacker even by the sophisticated programs and techniques currently in use. The maintenance of the program for generating the activation number at the activation center is inaccessible to the user and maintains secure that information which is needed to decode the activation number. The process of the present invention therefore does not require hiding of codes within the software.

Even if the hacker were successful in determining the derivation of the validation number and

the non-random activation number digits from the validation number, he or she may still be stumped by a failure to realize that the group of randomly generated digits A_3, A_1, A_5 must have balance relative to the digits of the validation number. Thus, a set of random digits A_3, A_1, A_5 will not allow correct installation of the software 18 unless the derived balance number is the same as the predetermined balance number. This "balance number" approach uniquely provides with the encrypted number generation a two-piece approach which is not a simple comparison which can be branched around by a hacker. Thus, even if the hacker is able to successfully break the code and generate an activation number which will cause an encrypted TP/PP to be written, his failure to select a set of random digits which will provide the predetermined balance number still prevents execution of the software since, upon decryption, the preselected signature cannot be located.

Thus, although a method for authorizing the use of a software package on a particular computer system is provided by the present invention so as to be virtually impossible, given the current state of the art, to decode or reverse engineer, the process is made convenient and easy for the software user, i.e., he or she need only make a phone call and follow some easy directions. With modem-to-modem communication, personal communication with an activation center operator is not

even required. Further, the unchanging and unique nature of the computer characteristics on which authorization is based allow the authorization process to be reliable, i.e., an authorization on one machine does not include others, and the user can be assured that the authorization will not be lost just because the computer characteristics may have changed since authorization.

Although the invention has been described in detail herein, it should be understood that the invention can be embodied otherwise without departing from the principles thereof, and such other embodiments are meant to come within the scope of the present invention as defined in the appended claims.

What is claimed is:

1. A method for authorizing use of software on a computer comprising the steps of:

a. programming the software to encrypt and output to the user of a computer a first number derived from information received by the software from the computer of at least one characteristic of the computer providing an unchangeable and unique identification of the computer; and

b. Operating an other computer thereby encrypting a second number derived from the first number for input to the user's computer to allow use of the software on the user's computer only if a predetermined relationship exists between the first and second numbers.

2. A method according to claim 1 further comprising selecting the at least one computer characteristic from the group of computer characteristics consisting of the serial number of the disk drive, the BIOS data from ROM, the number of sectors per track of the hard disk, the number of heads of the hard disk, and the number of cylinders of the hard disk.

3. A method according to claim 2 comprising programming the software to encrypt the first number from information as to all of said group of computer characteristics.

4. A method according to claim 2 comprising programming the software to encrypt the first number from information as to the BIOS data from ROM, the number of sectors per track of the hard disk, the number of heads of the hard disk, and the number of cylinders of the hard disk.

5. A method according to claim 1 comprising securing information relative to the process for encrypting the second number from access thereto by the user.

6. A method according to claim 1 comprising encrypting the second number pseudo-randomly.

7. A method according to claim 1 further comprising programming the software to encrypt the first number from the serial number of the hard disk.

8. A method according to claim 1 further comprising programming the software to provide an option to the user of the computer for trial of the software

for a specified period such that a new authorization for use is required after the trial period is concluded.

9. A method according to claim 1 further comprising programming the software to allow the authorization for use of the software to cover a specified number of computers in a network.

10. A method for authorizing use on a computer of software which has been programmed to output to the user of a computer a first number derived from at least one characteristic of the computer, the method comprising operating an other computer thereby encrypting for inputting to the user's computer a second number derived from the first number and including at least one randomly generated digit which, when a predetermined mathematical operation is performed on said at least one randomly generated digit and on at least one of the digits of the first number, yields in the user's computer a derived balance number used to encrypt a preselected signature in the user's computer whereby the software may be used in the user's computer upon use of a predetermined balance number equal to the derived balance number to retrieve the preselected signature by decryption thereof.

11. A method according to claim 10 further comprising selecting the predetermined balance number to be a prime number.

12. A method according to claim 10 comprising securing information relative to the process for encrypting the second number from access thereto by the user.

13. A method according to claim 10 comprising encrypting the second number pseudo-randomly.

14. A method for authorizing use of software on a computer comprising the steps of:

a. programming the software to output to the user of a computer a first number derived from at least one characteristic of the computer; and

b. operating an other computer thereby encrypting for inputting to the user's computer a second number derived from the first number and including at least one randomly generated digit which, when a predetermined mathematical operation is performed on said at least one randomly generated digit and on at least one of the digits of the first number, yields in the user's computer a derived balance number used to encrypt a preselected signature whereby the software may be used in the user's computer upon use of a

predetermined balance number equal to the derived balance number to retrieve the preselected signature by decryption thereof.

15. A method according to claim 14 further comprising programming the software to write information including said preselected signature encrypted by the derived balance number in the hard drive of the user's computer.

16. A method according to claim 15 further comprising programming the software to scatter the bytes of said encrypted information among randomly-generated bytes.

17. A method according to claim 14 further comprising programming the software to write information including said preselected signature encrypted by the derived balance number in the root of the hard disk of the user's computer, track 0 of the first cylinder of the user's computer, and at least one location on the diagnostic cylinder of the user's computer.

18. A method according to claim 14 further comprising selecting the at least one computer characteristic from the group of computer characteristics consisting of the serial number of the

hard disk, the BIOS data from ROM, the number of sectors per track of the hard disk, the number of heads of the hard disk, and the number of cylinders of the hard disk.

19. A method according to claim 14 further comprising selecting the at least one computer characteristic to provide an unchangeable and unique identification of the computer.

20. A method according to claim 14 further comprising programming the software to receive information relative to the at least one computer characteristic from the computer.

21. A method according to claim 14 further comprising programming the software to provide an option to the user of the computer for trial of the software for a specified period such that a new authorization for use is required after the trial period is concluded.

22. A method according to claim 14 further comprising programming the software to allow the authorization for use of the software to cover a specified number of computers in a network.

23. A method for authorizing use of software on a computer comprising the steps of:

a. programming the software to pseudo-randomly encrypt and output to the user of a computer a first number derived from information received by the software from the computer of at least one characteristic of the computer;

b. inserting the software in the computer and operating the computer to obtain the first number;

c. operating another computer thereby pseudo-randomly encrypting a second number derived from the first number and including at least one randomly-generated digit which, when a predetermined mathematical operation is performed on said at least one randomly generated digit and on at least one of the digits of the first number, yields a derived balance number;

d. inputting the second number to the user's computer;

e. operating the user's computer thereby obtaining the derived balance number;

f. operating the user's computer thereby assembling information including a preselected signature;

g. operating the user's computer thereby encrypting the assembled information using the derived balance number; and

h. operating the user's computer thereby writing the encrypted assembled information to the

user's computer whereby the software may be used in the user's computer upon use of the predetermined balance number to retrieve the preselected signature by decryption thereof.

24. A method according to claim 23 further comprising connecting the user's computer and another computer by modems for communication between the user's computer and the another computer.

25. A method according to claim 23 further comprising selecting the predetermined balance number to be a prime number.

26. A method according to claim 23 further comprising operating the user's computer thereby writing the encrypted assembled information including the preselected signature to at least one location in the root of the hard disk drive, track 0 of the first cylinder, and at least one location on the diagnostic cylinder.

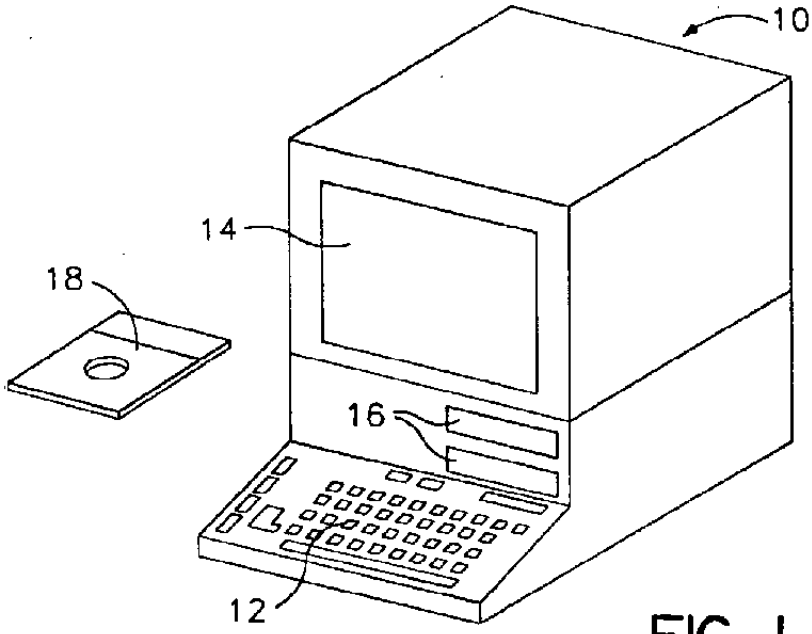


FIG. 1

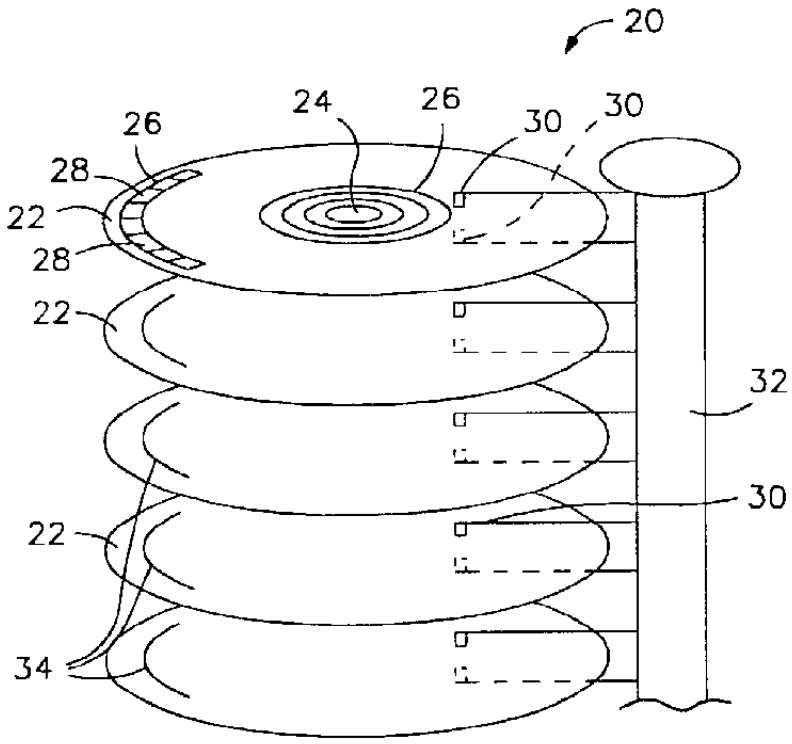


FIG. 2

SUBSTITUTE SHEET

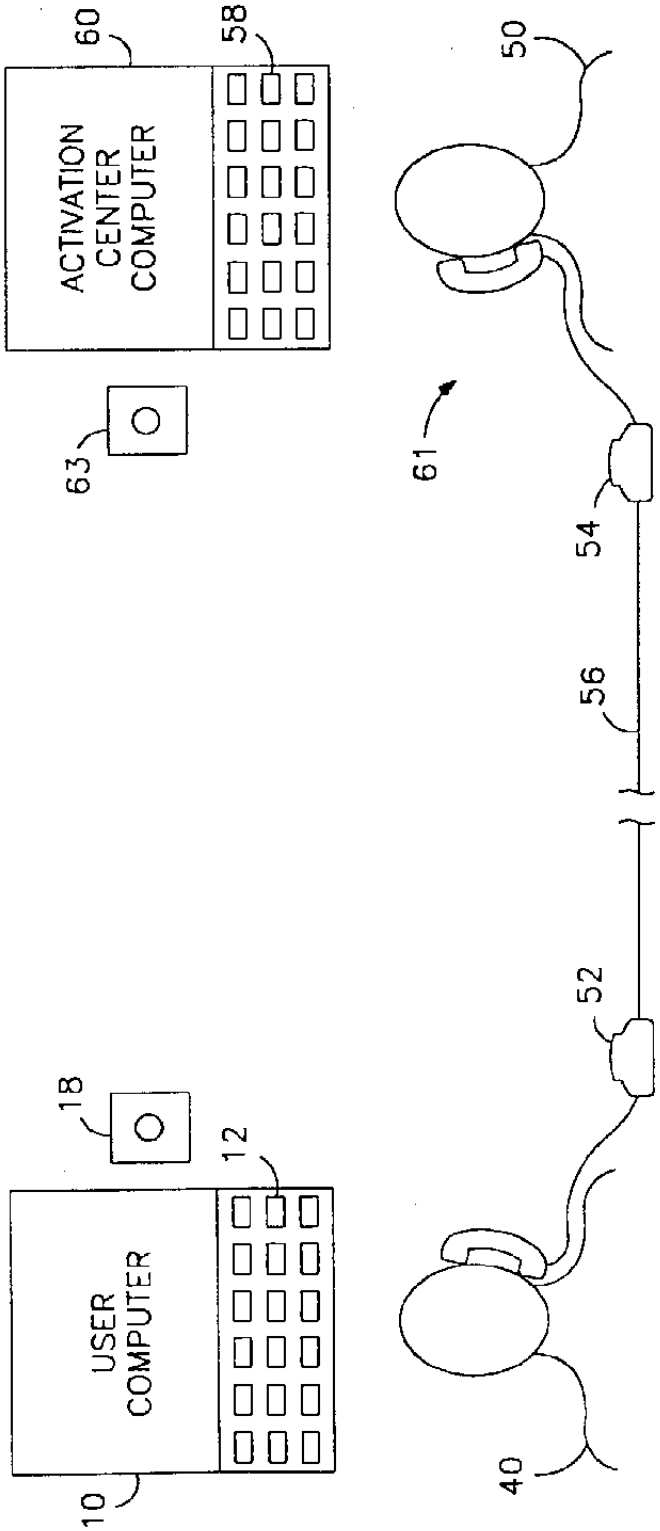

ACTIVATION CENTER COMPUTER
60
58
63
50
61
54
56
52
18
12
USER COMPUTER
10
40

FIG. 3

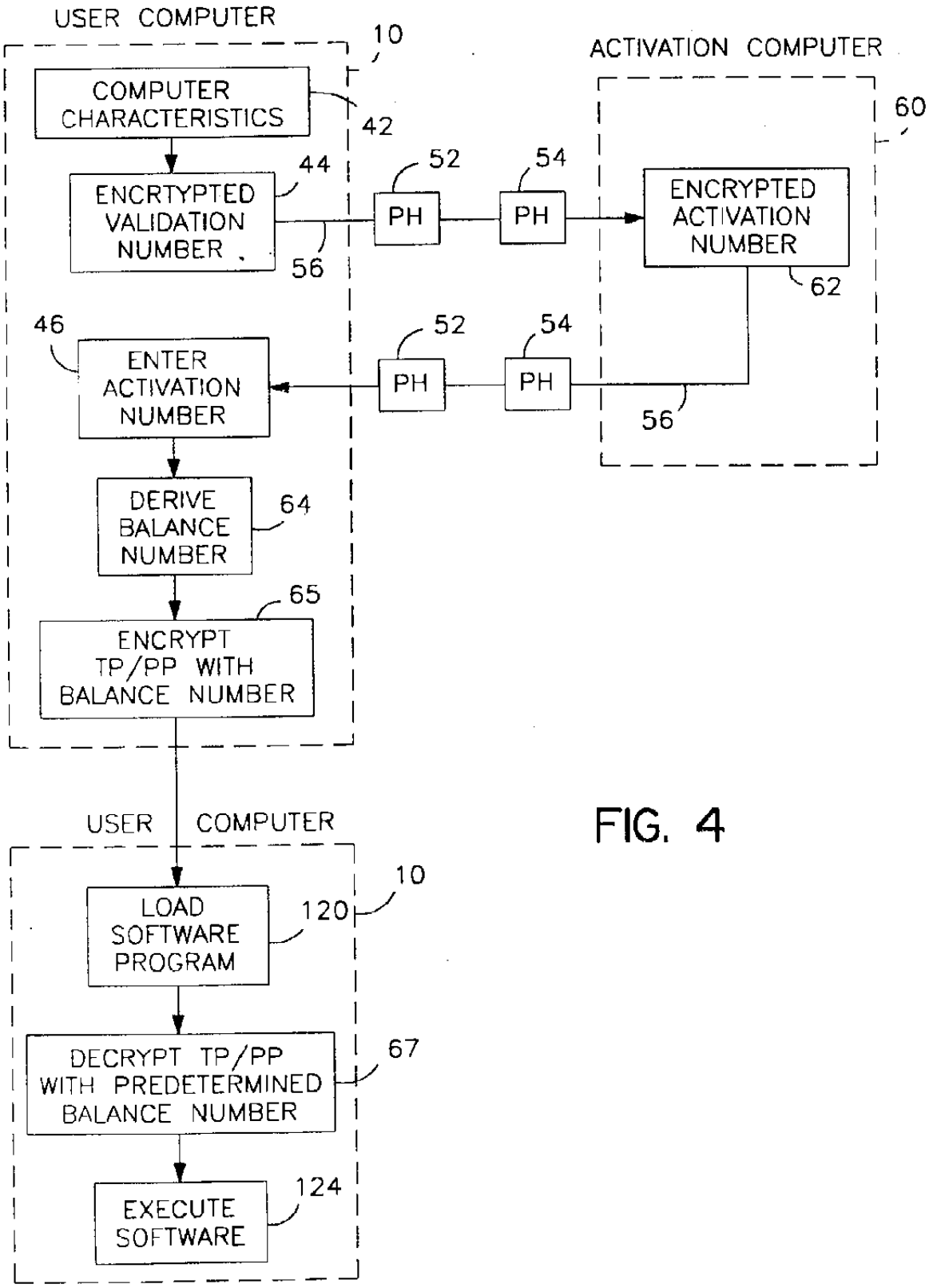

USER COMPUTER
10
COMPUTER CHARACTERISTICS
42
ENCRTYPTED VALIDATION NUMBER
44
56
52
PH
54
PH
ACTIVATION COMPUTER
60
ENCRYPTED ACTIVATION NUMBER
62
56
46
ENTER ACTIVATION NUMBER
52
PH
54
PH
DERIVE BALANCE NUMBER
64
65
ENCRYPT TP/PP WITH BALANCE NUMBER
USER COMPUTER
10
LOAD SOFTWARE PROGRAM
120
DECRYPT TP/PP WITH PREDETERMINED BALANCE NUMBER
67
EXECUTE SOFTWARE
124

FIG. 4

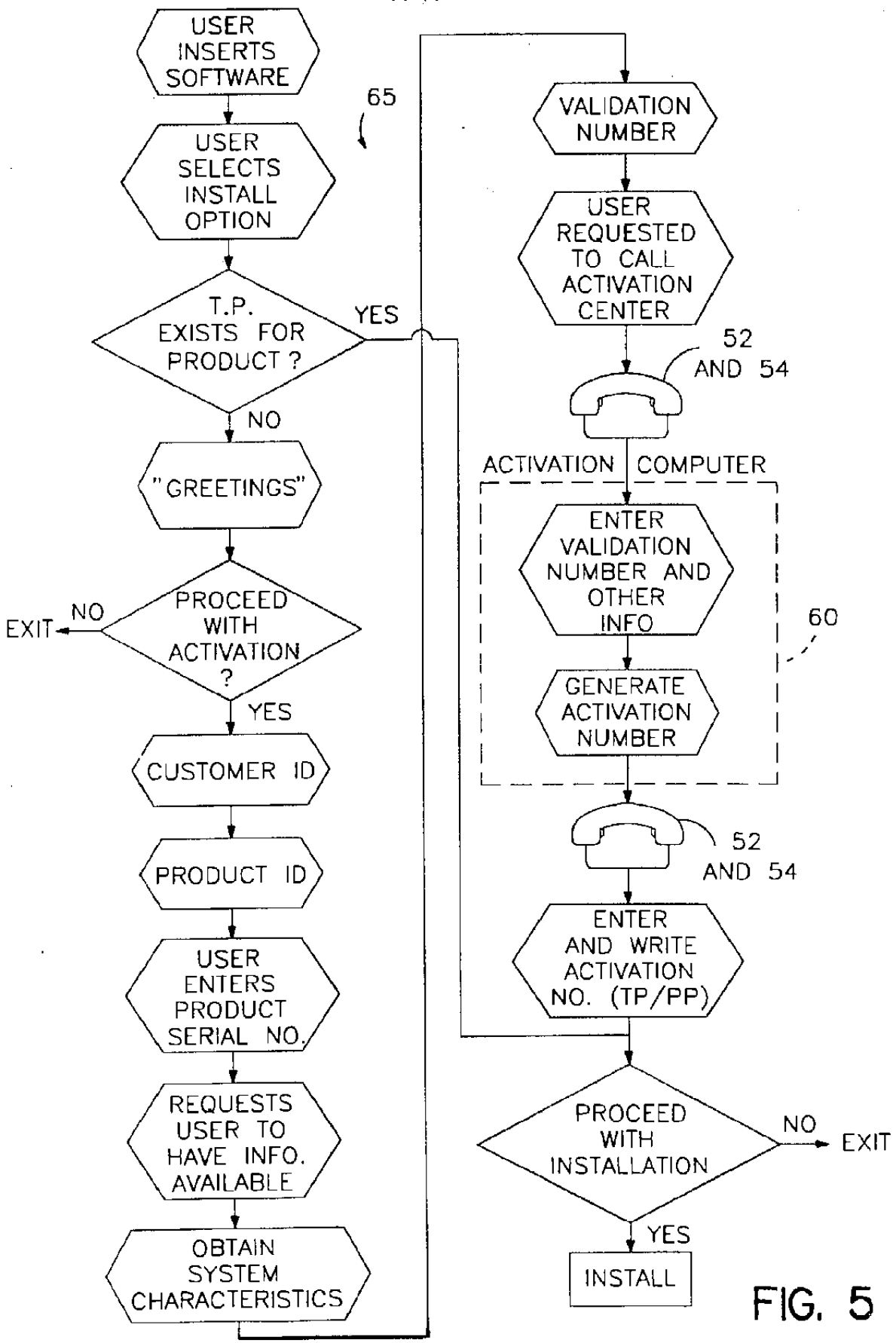


FIG. 5

APPLE EXHIBIT 1102, Page 502 of 1048

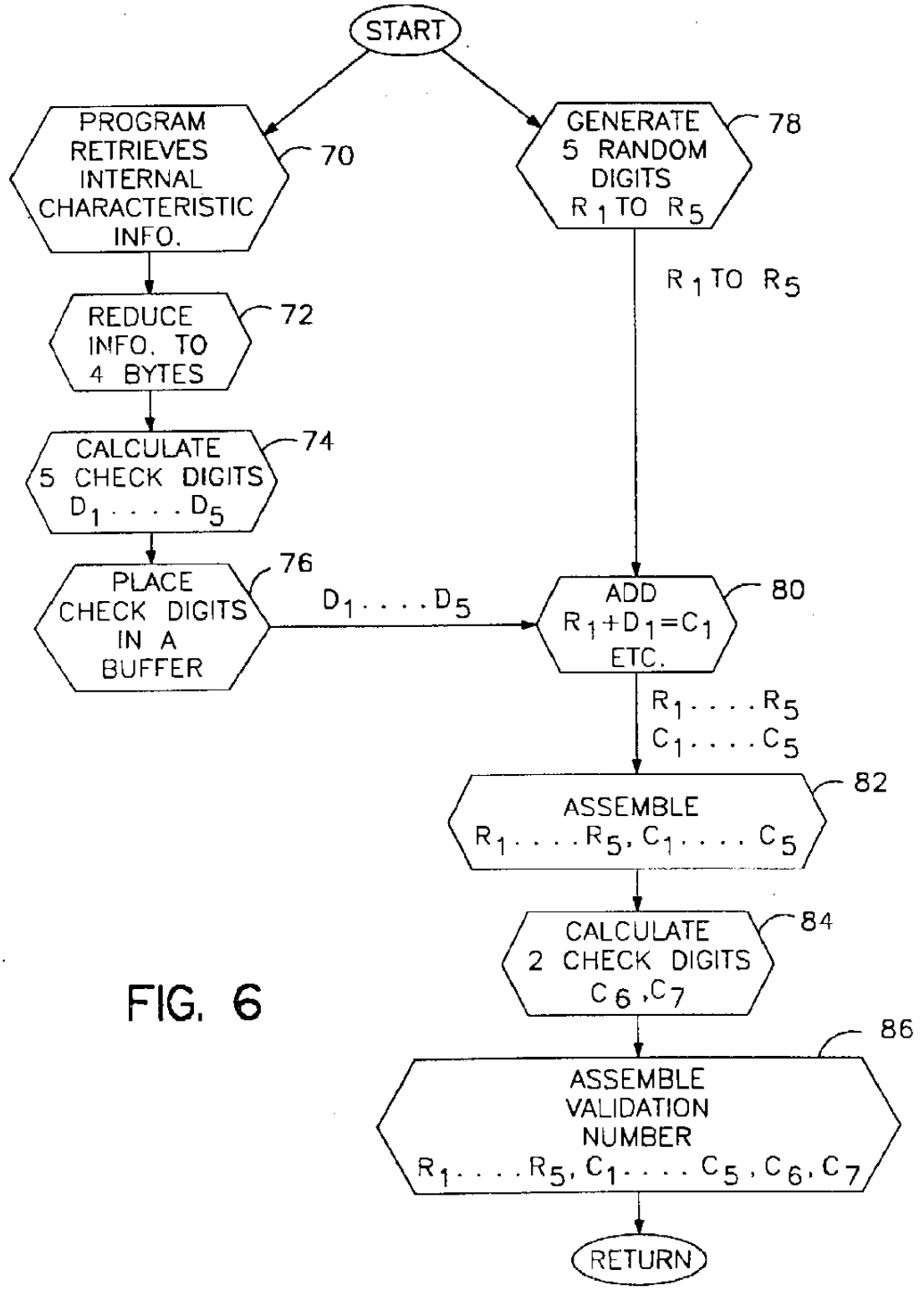


FIG. 6

SUBSTITUTE SHEET

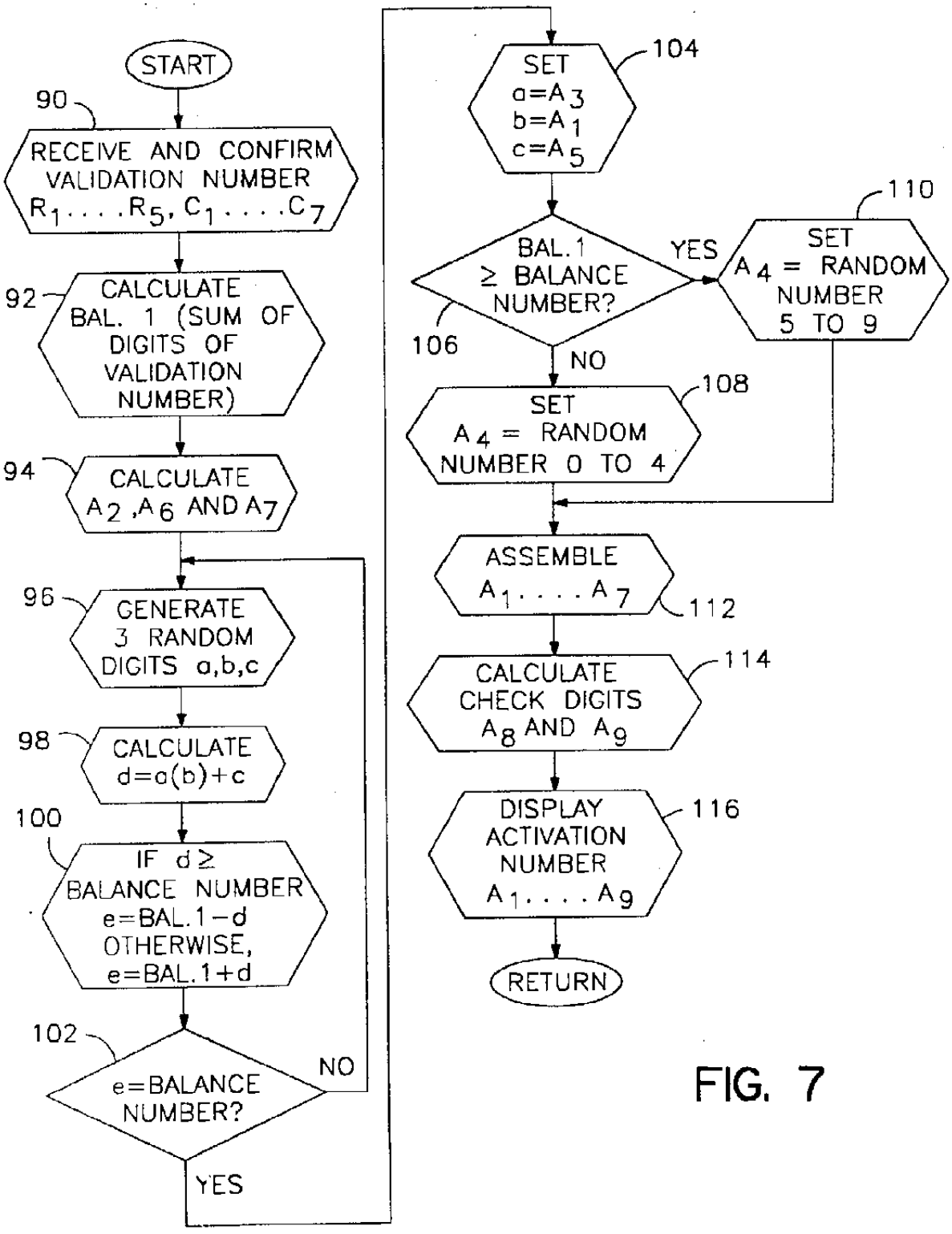

START
90
RECEIVE AND CONFIRM VALIDATION NUMBER $R_1 \ldots R_5, C_1 \ldots C_7$
92
CALCULATE BAL. 1 (SUM OF DIGITS OF VALIDATION NUMBER)
94
CALCULATE A_2, A_6 AND A_7
96
GENERATE 3 RANDOM DIGITS a,b,c
98
CALCULATE d=a(b)+c
100
IF d ≥ BALANCE NUMBER e=BAL.1−d OTHERWISE, e=BAL.1+d
102
e=BALANCE NUMBER?
NO
YES
104
SET a=A_3 b=A_1 c=A_5
106
BAL.1 ≥ BALANCE NUMBER?
YES
110
SET A_4 = RANDOM NUMBER 5 TO 9
NO
108
SET A_4 = RANDOM NUMBER 0 TO 4
ASSEMBLE $A_1 \ldots A_7$
112
CALCULATE CHECK DIGITS A_8 AND A_9
114
DISPLAY ACTIVATION NUMBER $A_1 \ldots A_9$
116
RETURN

FIG. 7

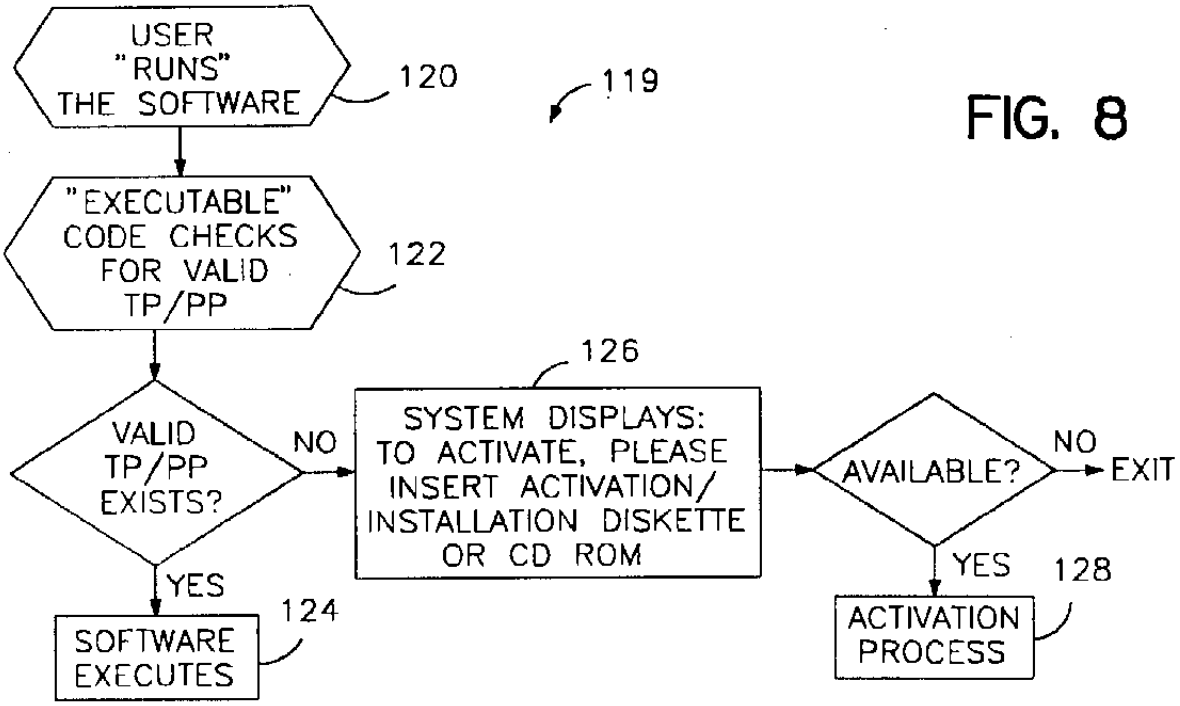


FIG. 8

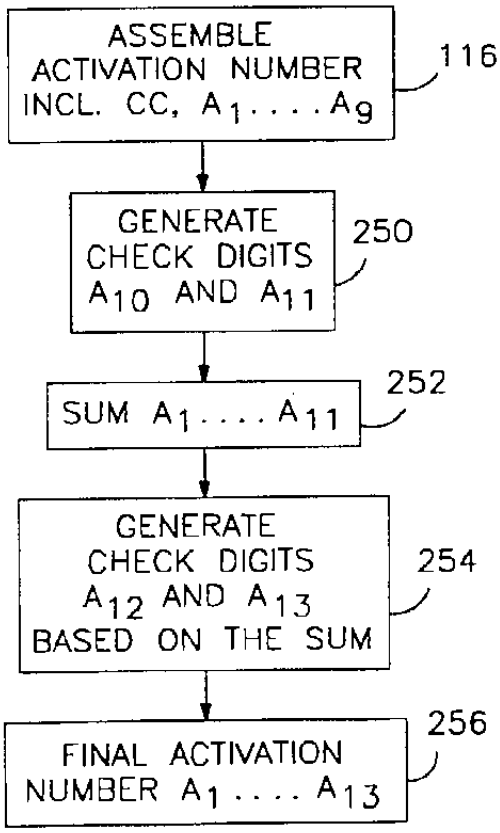


FIG. 17

SUBSTITUTE SHEET

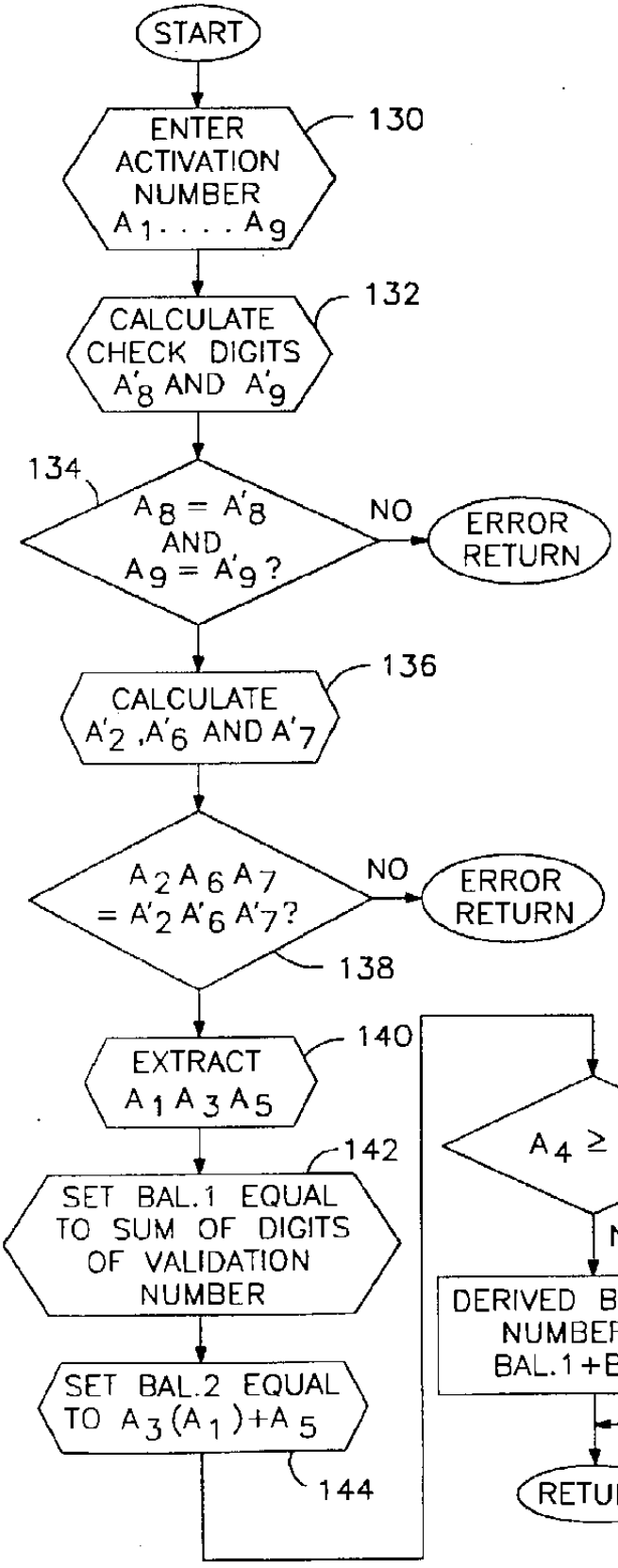

START
ENTER ACTIVATION NUMBER $A_1 \ldots A_9$
130
CALCULATE CHECK DIGITS A'_8 AND A'_9
132
134
$A_8 = A'_8$ AND $A_9 = A'_9$?
NO
ERROR RETURN
CALCULATE A'_2, A'_6 AND A'_7
136
$A_2 A_6 A_7 = A'_2 A'_6 A'_7$?
NO
ERROR RETURN
138
EXTRACT $A_1 A_3 A_5$
140
SET BAL.1 EQUAL TO SUM OF DIGITS OF VALIDATION NUMBER
142
SET BAL.2 EQUAL TO $A_3(A_1)+A_5$
144
$A_4 \geq 5$?
YES
146
DERIVED BALANCE NUMBER = BAL.1−BAL.2
NO
148
DERIVED BALANCE NUMBER = BAL.1+BAL.2
RETURN

FIG. 9

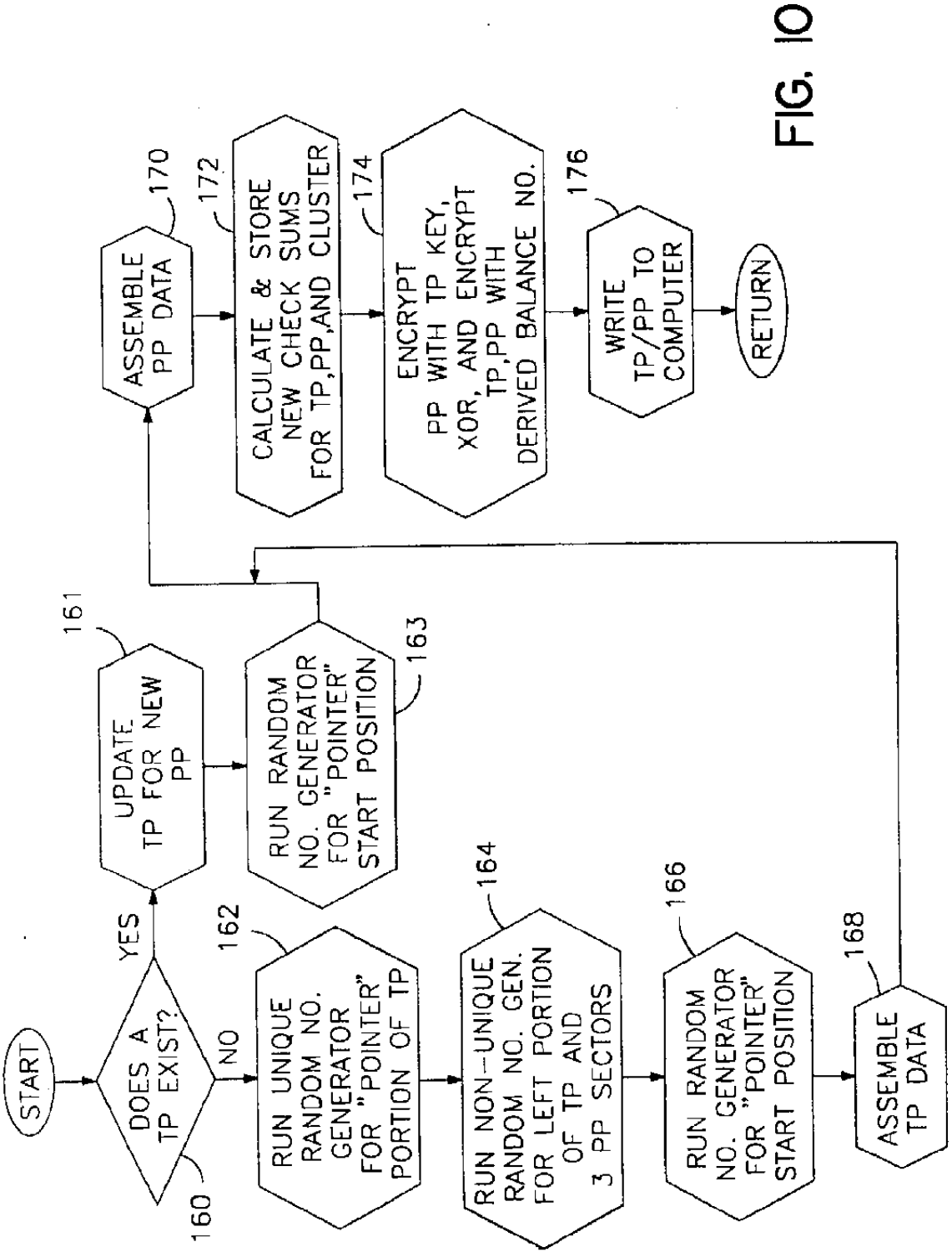


FIG. 10

SUBSTITUTE SHEET

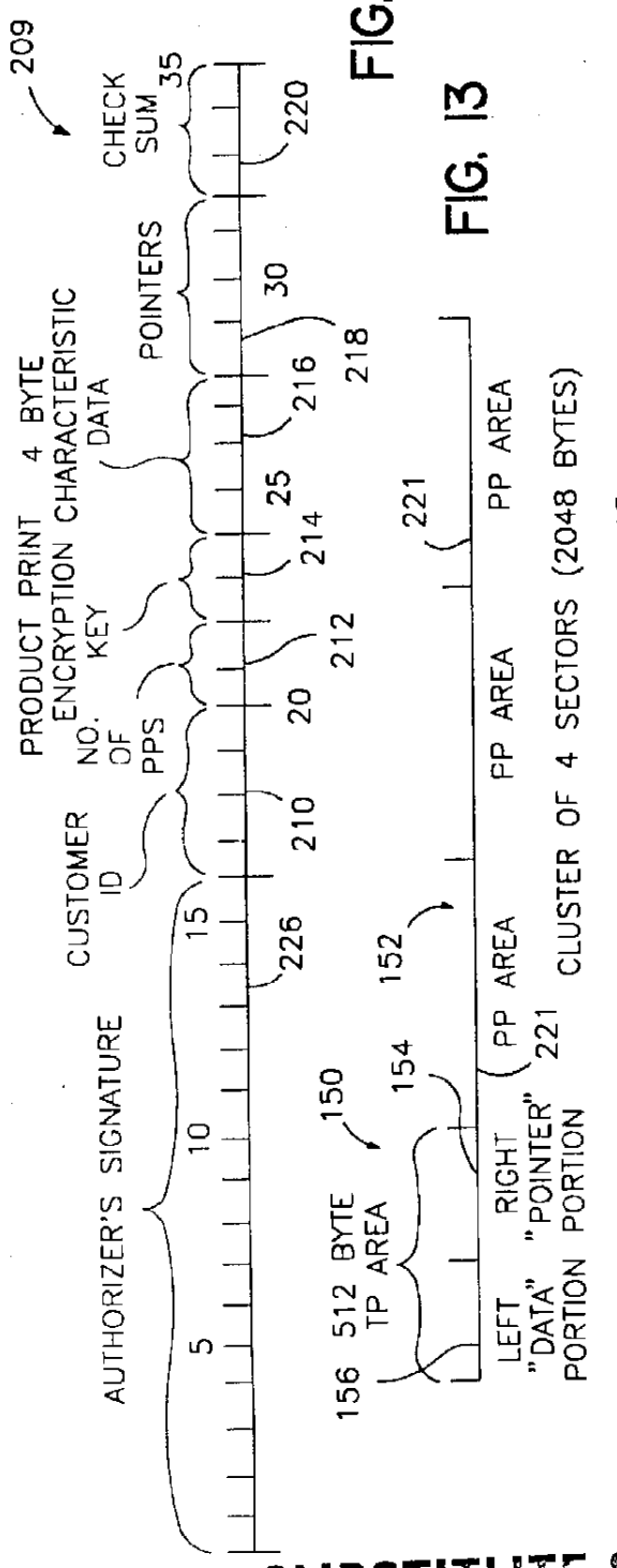


FIG. 11

FIG. 13

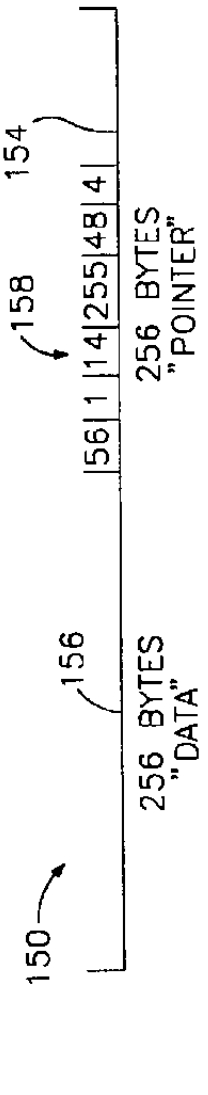


FIG. 14

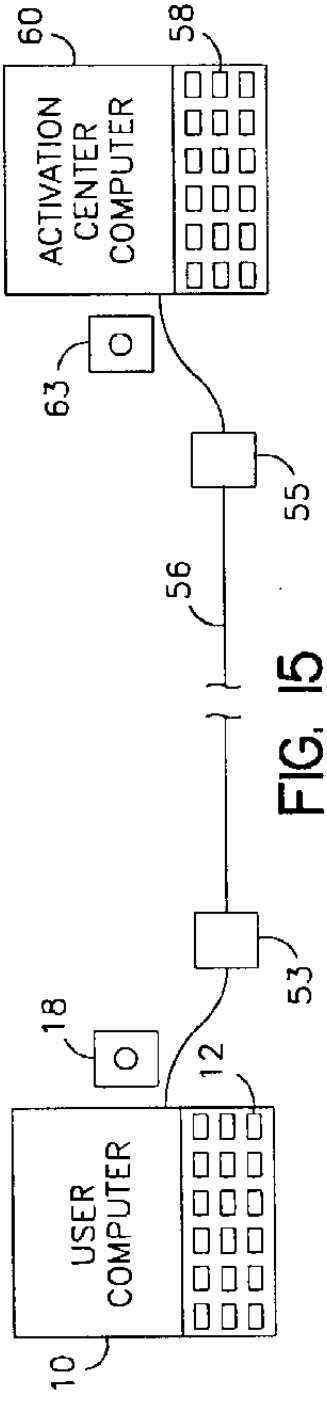


FIG. 15

APPLE EXHIBIT 1102, Page 508 of 1048

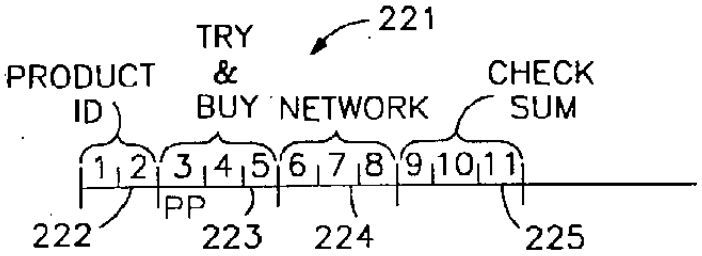


FIG. 12

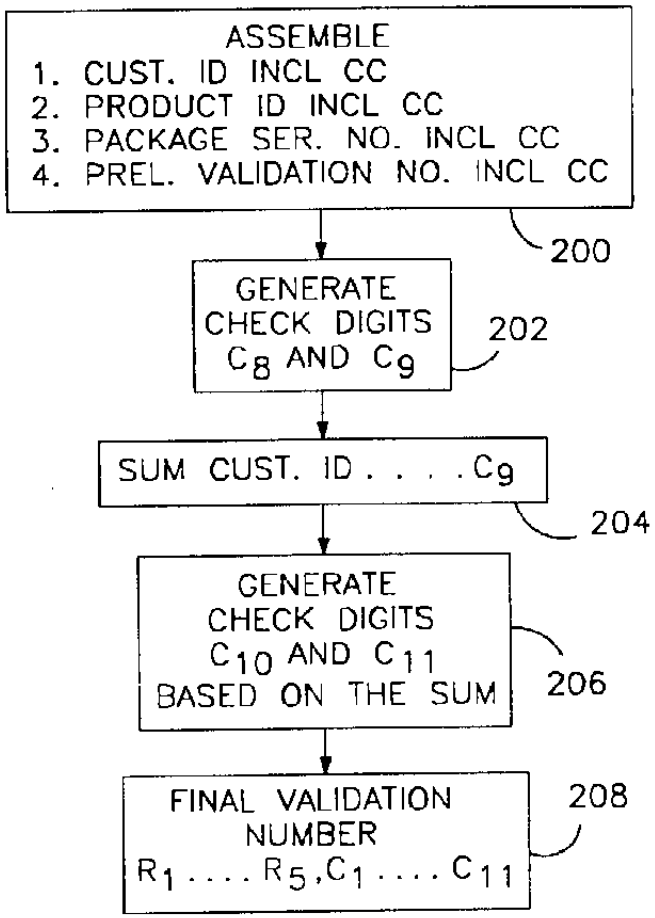


FIG. 16

SUBSTITUTE SHEET

INTERNATIONAL SEARCH REPORT

International Application No
PCT/CA 95/00354

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
P,X	US,A,5 337 357 (CHOU ET AL) 9 August 1994 see abstract; figure 1 see column 2, line 57 - column 3, line 17	1,5,10, 12,14,20
P,Y		6,8,9, 13,21

X	US,A,4 796 220 (WOLFE) 3 January 1989 cited in the application see abstract; figures 1-3 see column 3, line 41 - column 4, line 48 see column 6, line 4 - column 8, line 68	1,2,5
A		8,13,15, 17,19, 21,24,26

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 October 1995	18. 10. 95

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+ 31-70) 340-2040, Tx. 31 651 epo nl, Fax (+ 31-70) 340-3016	Powell, D

Form PCT/ISA/210 (second sheet) (July 1992)

APPLE EXHIBIT 1102, Page 510 of 1048

INTERNATIONAL SEARCH REPORT

International Application No
PCT/[illegible] 95/00354

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO,A,94 07204 (UNILOC) 31 March 1994 see abstract; figure 8 see page 7, line 4 - page 8, line 28 see page 22, line 14 - page 23, line 9	6,8,13, 21
A	---	1,2
Y	US,A,5 023 907 (JOHNSON ET AL) 11 June 1991 see abstract; figure 2 see column 3, line 66 - column 5, line 41	9
A	-----	8,21

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

APPLE EXHIBIT 1102, Page 511 of 1048

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internatic Application No
PCT/CA 95/00354

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
US-A-5337357	09-08-94	CA-A-	2120816	18-12-94
		EP-A-	0636962	01-02-95
US-A-4796220	03-01-89	NONE		
WO-A-9407204	31-03-94	AU-B-	4811393	12-04-94
		CA-A-	2145068	31-03-94
		CN-A-	1103186	31-05-95
US-A-5023907	11-06-91	NONE		

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 November 2005 (10.11.2005)

PCT

(10) International Publication Number
WO 2005/104686 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/US2005/012558

(22) International Filing Date: 14 April 2005 (14.04.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/562,335 14 April 2004 (14.04.2004) US

(71) Applicant *(for all designated States except US)*: **IPASS INC.** [US/US]; 3800 Bridge Parkway, Redwood Shores, California 94065 (US).

(71) Applicants and
(72) Inventors: **ELGRESSY, Moshe** [IL/IL]; Hagail 207, P. O. Box 165, 36001 Nofit (IL). **BOB, Kenneth** [US/US]; 34 Gilbert Drive, Plainview, New York 11803 (US).

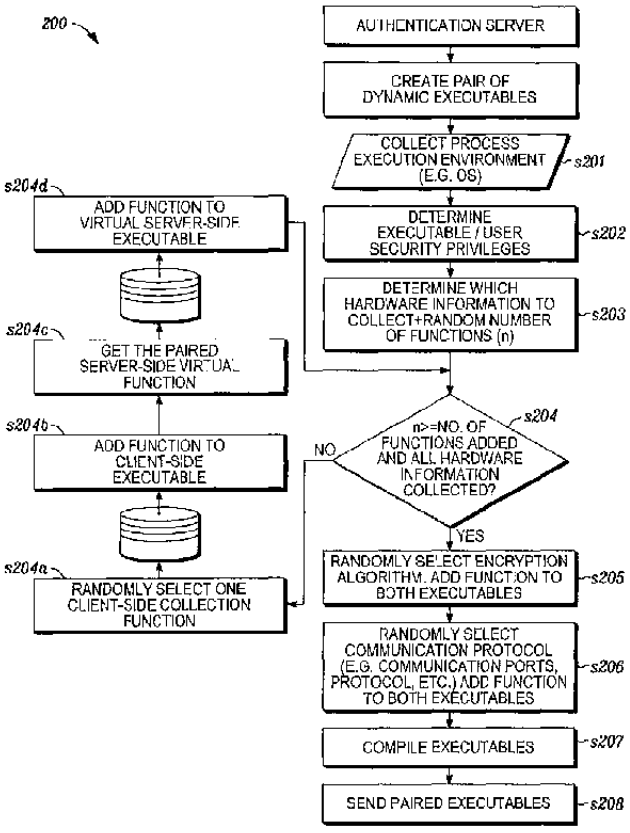
(74) Agents: **STEFFEY, Charles, E.** et al.; Schwegman, Lundberg, Woessner & Kluth, P.A., P.O. Box 2938, Minneapolis, Minnesota 55402 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: DYNAMIC EXECUTABLE



(57) Abstract: A system to identify and/or authenticate an entity includes a. client-side executable and an associated server side executable, each of which may be dynamically generated. The server-side executable may be executed on a server to generate a first result. The client-side executable may be executed on an entity to generate a second result. The first result and the second result are compared to identify and/or to authenticate the entity.

WO 2005/104686 A2

FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Published:

— *without international search report and to be republished upon receipt of that report*

DYNAMIC EXECUTABLE

CROSS-REFERENCE TO RELATED APPLICATION

[0001] The present application claims the priority benefit of U.S. provisional application serial number 60/562,335, filed April 14, 2004, which is incorporated herein by reference.

FIELD

[0002] The disclosed subject matter relates generally to a method and system for authentication and authorized access control.

BACKGROUND

[0003] In computers, executables may be downloaded and installed onto a machine for the purpose of executing a set of instructions. One form of executable is a file that contains a program - that is, a particular kind of file that is capable of being executed. In this context, execution may refer to the process of running the program or performing an operation called for by an instruction contained therein. Thus, a computer processor may, for example, execute the instruction, meaning that it performs the operations called for by that instruction.

[0004] Executables may include a wide variety of computer programs, including, for example, application programs, such as spreadsheets and word processors. In a Disk Operating System (DOS) or Windows operating system, the executable may have, for example, a file name extension of .bat, .com, or .exe. Such executables may sometimes be referred to as binaries since the format of the executable includes a sequence of binary values.

[0005] Executables may also include, for example, an applet, which is a program designed to be executed from within another application residing, for example, on remote machine. Unlike an application, applets may be restricted from direct execution within the operating system. Applets may be mini-programs that are downloaded and used by any computer equipped, for example, with a Java-compatible browser. In this context, Java refers to the programming language designed primarily for writing software on World Wide Web sites and downloaded over the Internet to a personal computer (PC).

[0006] Executables may also include, for example, a Windows Programming Interface 32-bit subset, sometimes referred to as Win32s, which is a software package that may be added to Windows 3.1 and Windows for Workgroups systems to give them the ability to run 32-bit applications. In this regard, Win32s may be used to convert between 32-bit and 16-bit memory addresses, an operation called "thunking". When installing such an application on a 16-bit Windows system, the installation procedure may automatically install the Win32s system if necessary.

[0007] Executables that are not stored and executed in a tamper-proof environment may be reversed engineered, and may be used to obtain identification credentials. Moreover, executables such as an applet or Win32s may return structured information, whose form may be static and widely known. Accordingly, the executables may be susceptible to reverse engineering and therefore unsuitable for authentication and authorization.

[0008] Authentication typically includes the process of identifying an entity, for example based on a username and password. In security systems, authentication may be regarded as distinct from authorization, which is the process of giving an entity access to protected resources based on their identity. Accordingly, authentication may ensure that the entity is who it claims to be, while authorization may determine the access rights of the entity.

SUMMARY

[0009] An exemplary method and a system to identify and/or authenticate an entity is described. A client-side executable and an associated server-side executable is dynamically generated. The server-side executable is executed on a server to generate a first result. The client-side executable is executed on an entity to generate a second result. The first result and the second result are compared to identify and/or to authenticate the entity.

[0010] Other features will be apparent from the accompanying drawings and from the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Embodiments of the present invention are illustrated by way of example and not limitation in the Figures of the accompanying drawings, in which like references indicate similar elements and in which:

[0012] Figure 1A shows a method and system to provide authentication and/or authorized access control of protected resources according to an example embodiment.

[0013] Figure 1B shows an implementation for the system of Figure 1A, according to an example embodiment.

[0014] Figure 2A shows a method to dynamically create a virtual pair of executables, according to an example embodiment.

[0015] Figure 2B shows assembly language instructions of an executable, according to an example embodiment.

[0016] Figure 2C shows representations in hexadecimal and ASCII form for 608 bytes of an executable, according to an example embodiment.

[0017] Figure 3A shows an illustration of hardware signature components according to an example embodiment.

[0018] Figure 3B shows a method to perform a hardware signature comparison, according to an example embodiment.

[0019] Figure 4 shows a method to compare obfuscated information in client-side and server-side executables, according to an example embodiment.

[0020] Figure 5 shows an authentication method using a pass/fail threshold level, according to an example embodiment.

[0021] Figure 6 illustrates a diagrammatic representation of a machine in the form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed, according to an example embodiment.

APPLE EXHIBIT 1102, Page 517 of 1048

DETAILED DESCRIPTION

[0022] Embodiments describe a method and a system to identify and/or authenticate an entity. A client-side executable and an associated server-side executable may be dynamically generated. The server-side executable may be executed on a server to generate a first result. The client-side executable may be executed on an entity to generate a second result. The first result and the second result may be compared to identify and/or to authenticate the entity. The at least one instruction may include a collection instruction, and/or a processing instruction, which may include an encryption instruction. A limited time period may be imposed on the client-side executable to send the second result.

Platform Architecture

[0023] Figure 1A shows an example method and system 100 to provide authentication and/or authorized access control of protected resources. The method and system 100 may be used, for example, to identify entities and better ensure that only "trusted" entities may access certain resources of a network or system. The access may include, for example, access to a web page, a wireless network, or an embedded application.

[0024] The example system 100 includes a client 151, an authenticator 152, and an authentication server 153. The client 151 may be an entity or application acting on behalf of the entity, which attempts access and/or requests authorization. The authenticator 152 provides an access point or proxy by which the client 151 may request access or authorization. The authentication server 153 provides an interface for data exchange with the access point. In this context, the client 151 may reside in a non-trusted environment, whereas the authenticator 152 and authentication server 153 may reside in a trusted environment. The authenticator 152 and authentication server 153 may be located on the same machine, or alternatively, the authenticator 152 and authentication server 153 may reside separately on different machines and/or different networks.

[0025] The client 151 may act on behalf of an entity to request authentication and/or access to protected resources. In this context, an entity, may be, for example, a machine, software-based application, or human, and may include, for example, a personal computer (PC), laptop, public terminal,

APPLE EXHIBIT 1102, Page 518 of 1048

handheld device, instance of a software program, or user(s) thereof. The client 151 may be implemented, for example, at least partially, via a Component Object Model (COM) object, which may be downloaded and installed via an electronic software distribution system or via the installation of a computer disc (CD). The client 151 may have, for example, administrative privileges on the resident machine.

[0026] The authenticator 152 may act as a gateway to facilitate an authentication session between the client 151 and the authentication server 153. In this regard, the authenticator 152 may initiate the authentication session on behalf of the requesting client 151, listen for an authentication result as determined the authentication server 153, and control access based on an authentication result. The authenticator 152 may be implemented, for example, as an edge device or edge switch, which is located at the edge of the network or system to be protected. The authenticator 152 may also be implemented, for example, at least partially as a web page. The authentication server 153 may validate information supplied by the client 151. In this regard, the authentication server 153 may access one or more databases, which store credential information regarding the entity.

[0027] Figure 1B shows an example implementation of the client 151, authenticator 152, and authentication server 153 of Figure 1A, based on the IEEE 802.1x standard for security of wireless local area networks (WLANS). The 802.1x standard provides an authentication framework that allows one or more entities to be authenticated by a central authority. The actual algorithm that is used to determine whether an entity is authentic is left open and multiple algorithms may be provided.

[0028] The 802.1x standard uses the Extensible Authentication Protocol (EAP), as defined in Request for Comment (RFC) 2284 published by the Internet Engineering Task Force (IETF), to exchange messages related to the authentication process during the link establishment phase. The authenticator 152 and authentication server 153 may further communicate using the Remote Authentication Dial-In User Service (RADIUS) as defined in IETF RFC 2139 and 2865 to validate information, sometimes referred to as credentials verification. It will be appreciated that other protocols may be suitable to

APPLE EXHIBIT 1102, Page 519 of 1048

exchange messages between the client 151, authenticator 152, and authentication server 153.

[0029] Figure 1A also shows an example method to provide authentication and/or authorized access, which demonstrate example interactions between the client 151, authenticator 152, and/or authentication server 153.

[0030] In block s101, a session is established and an authentication request is made on behalf of the entity requesting access. In this regard, the authentication may be requested for continued access as long as the session is active. Alternatively, the authentication may be provided, for example, on a one-time basis for each requested access. The request for access may include, for example, a request to participate in an Ethernet network, create a virtual private network (VPN), or launch an embedded application.

[0031] In block s102, access is denied. Access may be denied, for example, if a reported machine id does not match a corresponding stored machine id, or an irregularity is detected, for example, with respect to an element or aspect of client software, hardware or firmware.

[0032] In block s103, a client id is communicated from the client 151 to the authentication server 153. The client id is a logical identifier that may be used, for example, to provide so-called real cardinal machine identification, that is, an association between information collected and a client, despite the changing nature of which and how information is collected. Thus, for example, although the type of information collected may change, the client id should remain the same. The client id may be selected, for example, to optimize indexing during the validation of hardware signatures.

[0033] In block s104, one or more executables may be dynamically created, or alternatively, the executables may have been created in advance and selected as a subset from the executables created for each use. The executables may include, for example, a pair of executables - a client-side executable and a server-side executable. The client-side executable is intended to collect information regarding an entity or resource(s) associated with the client, and the server-side executable is intended to validate the information collected by performing identical or at least similar operations using information previously stored about the entity or associated resource(s). In this regard, the information

APPLE EXHIBIT 1102, Page 520 of 1048

may be stored, for example, using the Lightweight Directory Access Protocol (LDAP) as defined in IETF RFC 2251, or any other suitable database protocol.

[0034] The client-side executable may be created dynamically by selecting a subset of verification functions. The server may include a mapping of verification functions.

[0035] The information collected by the client-side executable may include, for example, low-level information regarding the entity's machine or operating system, sometimes referred to as cardinal information. For example, the executable may include COM object or a dynamic link library (DLL) operation to collect hardware information. DLL is a feature of OS/2 and Windows, which allows executable code modules to be loaded on demand and linked at run time.

[0036] The executable may also include, for example, a JavaScript or VBScript program, which may collect machine and browser properties. JavaScript and VBScript (Virtual Basic Script) are interpreted languages, which are evaluated during runtime.

[0037] The executable may also include, for example, a device driver to scan the entity's machine and return its hardware signature, based on the devices found. In particular, a hardware signature may be generated using information collected from, for example, a central processing unit (CPU), a Desktop Management Interface (DMI), Integrated Drive Electronics (IDE), an AT Attachment Packet Interface (ATAPI), and a Small Computer System Interface (SCSI).

[0038] The client-side and server-side executables may also be configured so that the collected information is obfuscated in a manner that is unique to each created pair. For example, if several pairs of executables were created for a given entity, each created pair may be configured to return a different result - even though the information used to identity has not changed.

[0039] The dynamic elements used to obfuscate the information collected or retrieved by the dynamic executables may include, for example, functions, implementation, workflow, encryption algorithms, communications protocols, or any combination thereof.

APPLE EXHIBIT 1102, Page 521 of 1048

[0040] The dynamic executables may be created, for example, on the authentication server 153, or alternatively on a different server and a subset may be randomly selected therefrom for each use.

[0041] In block s105, the client-side executable is sent to the client 151. The client-side executable may be sent, for example, over a secure encrypted channel. In particular, Rivest-Shamir-Adleman (RSA) and Data Encryption Standard (DES) encryption technology may be employed and encryption keys may change several times. Other encryption technologies may be used as well. The sending of the client-side executable may serve as the "challenge" in a "challenge/response". A "challenge/response" may be regarded as a type of authentication/authorization procedure, in which a correct response to the challenge is required to confirm an identity or to gain access. In the context of authenticating a hardware component, the challenge may be referred to as a machine hardware challenge (MHC).

[0042] In block s106, the client-side executable collects entity information, which may be gathered from or computed by using various hardware and/or software components of the client 151. The collected information may be processed, for example, hashed and/or digested using the Message Digest version 5 algorithm (MD5). As defined in IETF RFC 1321, MD5 is an algorithm (a.k.a., a one-way hash function), which takes an input message of arbitrary length, and produces a fixed-length output in the form of a 128-bit digital signature, sometimes referred to as a "fingerprint" or "message digest". It will be appreciated that algorithms other than MD5 may be used to process and/or hash the collected information.

[0043] The MD5 operation may be performed independently on information collected separately for each resource component, resulting in multiple individual digital signatures. Depending upon the instruction set of the particular instance of client-side executable, the information may undergo other mathematical manipulations to further obfuscate their representation. The digital signatures may also be concatenated together and/or their bytes transposed into a different order.

[0044] The MD5 operation may be performed on information collected from any type of resource component of the client, including both hardware and software components. If, for example, the MD5 operation is performed on

APPLE EXHIBIT 1102, Page 522 of 1048

information collected from a hardware component, the resulting digital signature may be referred to as a "device signature". In this regard, each device signature may be prefixed with a 4-byte identifier of the device type, which together may be collectively referred to as the "device block". Device blocks may undergo additional mathematical manipulations as instructed by a particular instance of the executable to further obfuscate their representation. The obfuscated device blocks may be concatenated together and each byte may be transposed into a different ordering.

[0045] In block s107, the information collected by the client-side executable, having been processed, such as hashed, obfuscated, and/or transposed, is sent to the authentication server 153, thereby serving as a "response" to the previously sent challenge. A secure channel may be used for communication.

[0046] The particular communications channel and protocol used to communicate the information may be specified by the client-side executable as well. For example, the client-side executable may include instructions to use a particular communication port and/or a particular communication protocol.

[0047] In block s108, the information sent by the client 151 is compared with information generated by the server-side executable. In particular, the bytes of received information may be "un-transposed" to their original consecutive order, the mathematical manipulations previously performed by the client-side executable reversed, and the results compared with results of the operations performed by the server-side code on the previously stored information for the entity.

[0048] In block s109, the requested access is permitted or denied depending, for example, on whether or not the information collected by the client-side executable is returned to the authentication server 153 with a predefined time period as measured from the time that the client-side executable was sent to the client 151, and whether such return information compares favorably with that which is generated by the server-side executable. For example, access may be permitted if the information is returned within the predefined time period (e.g., 50 milliseconds) and the results of the comparison properly match. Otherwise, access may be denied, for example, if the information is returned after the predefined time period (e.g., more than 50

APPLE EXHIBIT 1102, Page 523 of 1048

milliseconds later) or the client-side and server-side executable produce different results. To this end, the authenticator 152 may include an expiration indicator to render the entity as not identified if the client-side generated result is not received by the authenticator 152 during a limited time frame.

Flowchart

[0049] Figure 2A shows an example method 200 to dynamically create a virtual pair of executables. In this regard, the request to create the executables may originate, for example, within the authentication server 152 as described in connection with Figure 1A.

[0050] In block s201, information regarding an execution environment of the executable is collected. The information may include, for example, an operating system type, such as the Unix or Windows operating systems, a CPU characteristic, such as 32-bit or 64-bit microprocessor, or a device type, such as, a portable handheld device or stationary network node. The information may be collected, for example, by challenging the client or by challenging the associated agent, who then may respond with the appropriate information regarding the execution environment.

[0051] In block s202, executable/user security privileges are determined so that an appropriate instruction set may be constructed which is capable of valid execution in the target environment. In this context, security privileges may refer to, for example, permitted actions or access rights with respect to a particular system resource, such as a hardware device, directory, file, or program.

[0052] In block s203, the particular hardware information and number of randomly selected functions is determined. In this regard, the hardware information may include, for example, information related to a hard drive or Compact Disc - Read Only Memory (CD-ROM), or any other device information available via an appropriate interface, including, for example, information available via a Desktop Management Interface (DMI) or a Small Computer System Interface (SCSI). The randomly selected functions may be determined, for example, to be several different functions to collect a specific hardware serial number three times.

APPLE EXHIBIT 1102, Page 524 of 1048

[0053] In blocks s204a and s204b, a client-side collection function is randomly selected and added to the client-side executable. In this context, the client-side collection function and client-side executable are intended for performance and execution in a non-trusted environment, sometimes also referred to as a hostile environment.

[0054] In blocks s204c and s204d, a paired server-side virtual function is obtained and added to the server-side virtual executable. In this context, the server-side virtual function and server-side virtual executable are intended for performance and execution in a secure environment in order to "simulate" the performance and execution of the client-side collection function and client-side executable.

[0055] At block s204, blocks s204a through s204d may be repeatedly performed until the appropriate number of functions is added for all hardware information intended to be collected, as previously determined in block s203.

[0056] In block s205, a processing algorithm, such as an encryption algorithm, may be randomly selected and added to both executables. In this regard, the selected algorithm may be symmetric or non-symmetric, including, for example, such encryption algorithms as Rivest-Shamir-Adleman (RSA) and Data Encryption Standard (DES). The encryption key size may be varied as well.

[0057] In block s206, a communication protocol is randomly selected and associated with each executable. The communication protocol may specify, for example, a specific transport layer, such as the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP). The communication protocol may also specify, for example, a specific communications port.

[0058] In block s207, the executables are compiled. In this context, "compile" may refer, for example, to the act of translating the instructions of the executable into machine or object code. "Compile" may also refer, for example, to a mere adjustment of the executable in order to conform to the target environment or ensure its valid execution therein.

[0059] In block s208, the executables are sent. In this regard, one executable may be sent to the client and the other executable may be sent to the authentication server. The executables may be sent, for example, via a Secure

APPLE EXHIBIT 1102, Page 525 of 1048

Socket Layer (SSL) channel or proprietary Transmission Control Protocol/Internet Protocol (TCP/IP).

Data Structures

[0060] Figure 2B shows example assembly language instructions 253 of an executable. It will be appreciated that the instructions of an executable may be provided in a variety of computer languages.

[0061] Figure 2C shows example representations for 608 bytes of an executable. The example representations include an example hexadecimal (base-16) representation 251 and an American Standard Code for Information Interchange (ASCII) character representation 252. It will be appreciated that an executable may be provided in any suitable format to accommodate a wide variety of processing environments.

[0062] Figure 3A shows an example representation of components of a hardware signature 350. In this regard, the hardware signature 350 may include a header 352, which specifies the version of the hardware signature 350, and a quantity of device blocks 358 contained in the hardware signature 350. In this context, the hardware signature version may refer to a particular format of the hardware signature 350, and the device block may refer to a combined representation of the device signature 356 and corresponding device type 354.

Flow Chart (s)

[0063] Figure 3B shows an example method 300 to perform a hardware signature comparison. In this context, a hardware signature may be, for example, a combination of all device signatures associated with a particular entity. The device signature may be, for example, a hash of information identifying a particular device.

[0064] In block s301, hardware signature HWSig #1 is compared to hardware signature HWSig #2. The HWSig #1 may refer to the hardware signature associated with the client machine image previously saved on the server at a time when the client machine may have been previously been matched and/or authenticated. HWSig #2 may refer to a hardware signature based on currently collected information from the client machine.

APPLE EXHIBIT 1102, Page 526 of 1048

[0065] At block s301a, the process queries whether hardware signature HWSig #1 is equal to hardware signature HWSig #2, and if so, then a matching machine may be assumed at block 305. If hardware signature HWSig #1 is not equal to hardware signature HWSig #2, then the hardware signature versions are compared at block s301b. If the hardware signature versions do not match at all, then a non-matching machine may be assumed at block s306. Otherwise, if the hardware signature versions match at least partially, then in block s302 the hardware signatures are arranged by device block. In this context, arranging the hardware signatures by device block may refer, for example, to dividing the hardware signatures into known device blocks as anticipated by a particular version of the hardware signature.

[0066] At times hardware components may change due to, for example, failures or swapping of components. For instance, a particular internal CD-ROM may be swapped or exchanged for a newer model. To accommodate such changes in hardware components, a threshold level of matching hardware signature components may be specified rather than requiring an absolute match. In particular, the threshold may be a numerical score, which is obtained by classifying devices based on their ability to provide positive or strongly identifying characteristics. For example, a CPU identifier may be classified as a relatively strong identifier and accordingly assigned a relatively high score, whereas a hard drive with no serial number may be classified as a relatively weak identifier and accordingly assigned a relative low score. Hence, the devices may be associated with a score that reflects how strongly the devices act as identifiers relative to other devices.

[0067] At block s303, and in blocks s303a through s303e, a device block by device block comparison is performed for each and every hardware signature device block, and a total machine device score is computed at block 303e. In particular, for each device block of hardware signature HWSig #1, a comparison is made with all device blocks of hardware signature HWSig #2. If the current device block of hardware signature HWSig #1 is greater than the current device block of hardware signature HWSig #2 (e.g., there are more devices identified in HW Sig #1 than in HW Sig #2), then in block s303b the hardware signature HWSig #2 is advanced. When the hardware signature HWSig #2 is advanced, extra devices identified in HWSig #1 are not considered and are taken out of the

APPLE EXHIBIT 1102, Page 527 of 1048

comparison (e.g., not scored). If information regarding a specific device is not collected anymore, then that device is not considered or scored. For example, there may be a known way to exchange or adjust identification information on a network card. Matching may not be based upon this network card, and this network card may not be scored.

[0068] Else, if the current device block of hardware signature HWSig #1 is less than the current device block of hardware signature HWSig #2 (e.g., there are more devices identified in HW Sig #2 than in HW Sig #1), then in block s303c the hardware signature HWSig #1 is advanced. That is, the extra devices identified in HW Sig #2 are not considered and not scored. In an example, the extra device(s) of HW Sig #2 may not be used, but may be saved on the server. The HW Sig #2 image may be updated on the server as discussed in regard to Figure 5.

[0069] Otherwise, if the current device block of hardware signature HWSig #1 equals the current device block of hardware signature HWSig #2, then in block s303d the matching device is multiplied by a device strength score. For example, a CPU ID is something that the user is unlikely to change, because the CPU is unlikely to change. Accordingly, a CPU ID that matches may be given a relatively high score. However, for a hard drive and/or a CD-ROM, they may need to be replaced, and may correspondingly have a lower score.

[0070] In block S303e, the device scores are summed or added for all devices found. Thereafter, the method 300 returns to block s303a until each device identified in HWSig #1 is compared, and an associated score is calculated.

[0071] If, after comparing all devices of the hardware signature HWSig #1 at block s304, the total machine device score is greater than or equal to the selected threshold, then a matching machine may be assumed at block s307. Otherwise, if the total machine device score is less than the selected threshold, then a non-matching machine may be assumed at block s306.

[0072] It will be appreciated that the example method 300 to perform a hardware signature comparison demonstrates a dynamic aspect that may, in addition to randomly selecting functions, provide a flexible approach to add functions on a continuing basis.

APPLE EXHIBIT 1102, Page 528 of 1048

[0073] Figure 4 shows an example method 400 to compare obfuscated information in client-side and server-side executable pairs.

[0074] In block s401a through s405a, hard drive serial numbers, CPU serial numbers, Basic Input/Output System (BIOS) serial numbers, Desktop Management Interface (DMI) particulars, and other hardware information is collected for the client-side executable.

[0075] In block s401b through s405b, hard drive serial numbers, CPU serial numbers, BIOS serial numbers, DMI particulars, and other hardware information is received for the server-side executable from a database 410. It will be appreciated that the database 410 may be local or remote, or may be obtained, for example, from an encrypted image resident on any machine, including, the client, authentication server, or third-party machine.

[0076] In block s406a, the hardware information collected on behalf of the client-side executable is processed, for example: encrypted, hashed, and/or scrambled in an appropriate manner.

[0077] In block s406b, the hardware information received on behalf of the server-side executable is processed, for example: encrypted, hashed, and/or scrambled in an appropriate manner.

[0078] In blocks s407a and s407b, the processed (e.g., encrypted, hashed, and/or scrambled) data is sent to an appropriate application for comparison, such as, for example, an authentication server, which may reside, for example, locally or remotely, and may be provided, for example, by a third party.

[0079] In block s408, the processed (encrypted, hashed, and/or scrambled) data is received and the results are compared to determine if they match. Block s408 may occur within the authenticator 152.

[0080] At block s409, the results (e.g., the data from the client-side and the data from server-side) may be compared. If the results are identical, the method 400 proceeds to block s410. If the results are not identical, the method 400 proceeds to block s411.

[0081] Figure 5 shows an example method 500 for authentication of a client by, for example, the authentication server 153 using a pass/fail threshold level. In this regard, the example method may use, for example, a virtual pair of executables, which may include, for example, a client-side executable and a server-side executable.

APPLE EXHIBIT 1102, Page 529 of 1048

[0082] In block s501, the method 500 queries whether an execution environment indication returned by the client-side executable is validated. In this regard, the execution environment indication may include, for example, an indication of a particular operating system. If the execution environment indication is invalid, the client is deemed not authenticated and the method 500 proceeds to block s511. Otherwise, if the execution environment indication is valid, the example method 500 proceeds to block s502.

[0083] In block s502, the method 500 queries whether the hardware signature HWSig returned by the client-side executable is validated. In this regard, the hardware signature may be validated, for example, for a proper version and/or format. The hardware signature HWSig may also be validated for proper size, including, for example, a proper header size or proper device block size. If the hardware signature HWSig is found to be invalid, then the client is deemed not authenticated and the method 500 proceeds to block s511. Otherwise, if the hardware signature HWSig is valid, then the example method 500 proceeds to block s503.

[0084] In block s503, an attempt is made to find a matching machine id. In this regard, a table of machine ids may be maintained. The table of machine ids may be maintained, for example, in a database. The database may be, for example, local or remote. If a matching machine id is not found, then the client is not authenticated and the method 500 proceeds to block s511. Otherwise, if a matching machine id is found, then the example method proceeds to block s504.

[0085] In block s504, information required for the server-side executable is extracted and inputted to the server-side executable. In this regard, the extracted information may include, for example, one or more devices previously known to be associated with the client.

[0086] In block s505, the information extracted and input to the server-side executable is compared with information found by the client-side executable. In this regard, the information found by the client-side executable may include, for example, one or more devices associated with the client.

[0087] In block s506, a system-wide threshold for authentication is obtained. In this regard, the authentication threshold may be, for example, a hardware-related and/or software-related threshold.

APPLE EXHIBIT 1102, Page 530 of 1048

[0088] In block s507, if the system is configured for identical authentication and the information found by the client-side executable and the information extracted and input to the server-side executable are non-matching as defined by the threshold, then the client is deemed not authenticated and the method 500 proceeds to block s511. Else, if the information found by the client-side executable is identical, the example method 500 proceeds to block s509.

[0089] Alternatively in block s507, if the system is configured for threshold authentication then if the information found by the client-side executable is similar to the information extracted and input to the server-side executable, as defined by a computed strength score equaling or exceeding the authentication threshold, then the example method 500 proceeds to block s508. For example, the threshold score may be set at 80% of identical.

[0090] In block s508, the stored hardware signature HWSig is updated in the server with information related to new device(s) of the client machine. In this regard, the update may include, for example, an update of one or more hardware signature components, including, for example, one or more device blocks or contents thereof. The server side image of the client machine may be updated at block s508. The example method 500 then proceeds to block s509.

[0091] In block s509, the stored hardware signature is examined to determine a particular hardware signature version. In this regard, the stored hardware signature version may be, for example, a hardware signature version 1 (HWSig-v1) or a hardware signature version 2 (HWSig-v2).

[0092] If, for example, the stored hardware signature version is not the latest version, e.g., hardware signature version 1 (HWSig-v1), then the example method 500 proceeds to block s510.

[0093] Otherwise, if the stored hardware signature version is the latest version, e.g., hardware signature version 2 (HWSig-v2), then the client is deemed authenticated and the method 500 proceeds to block s512.

[0094] In block s510, the stored hardware signature version of the client machine is updated at the server side to reflect the new version and the client is deemed authenticated and the method 500 proceeds to block s512. In this regard, the stored hardware signature version may be updated, for example, from hardware signature version 1 HWSig-v1 to hardware signature version 2 HWSig-v2.

APPLE EXHIBIT 1102, Page 531 of 1048

[0095] Embodiments may provide a system and method to collect information from an entity in a secure manner so that the identity of the entity may be authenticated and/or access to certain protected resources by the entity may be authorized. The collected information may include, for example, hardware and/or software-related information regarding the entity or resources associated with the entity.

[0096] According to an example embodiment of the present invention, the information may be collected via a virtual pair of executables. In this context, a virtual pair may include two executables: one executable to collect information about an entity and the other executable to compare the collected information with information previously stored for the entity. In this regard, each executable may include a set of instructions intended to be executed in a suitable processing environment.

[0097] The executables may be created dynamically, that is, "on the fly," to prevent reverse engineering. Alternatively, one executable may be static, created only once to include, for example, all potential functions, while the second executable may be created dynamically by selecting a subset of all potential functions. In this regard, the dynamically created executable may perform the functions in a non-trusted environment (sometimes referred to as a hostile environment) while a mapping of selected functions may be retained on a server in the trusted environment.

[0098] The type, order, and/or amount of information, or the method of collecting and sending the information, may be determined in a dynamic manner as well. For example, a first pair of executables may concern the collection of a central processing unit (CPU) id and a hard drive serial number obtained from a registry, whereas a second pair of executables may concern the collection of only the hard drive serial number as obtained not by the registry but by accessing the device at the lowest level over a certain port. Consequently, the information returned by a particular executable pair may be unique, or at least unlikely to be duplicated by another pair of executables.

[0099] According to an example embodiment, the virtual pair of executables may be configured to return an identical, or at least similar, result for a common set of dynamically-created functions so that the collected information may be verified despite the collection occurring, for example, in a so-called non-

APPLE EXHIBIT 1102, Page 532 of 1048

trusted or hostile environment. A limited amount of time may be imposed to collect and return the information so that, if the information is not returned prior to this stipulated time, the information may be deemed unreliable and appropriately disregarded, For example, if information is not collected and returned within 50 milliseconds, any information returned thereafter is considered suspect and may be accordingly disregarded. Hence, by dynamically creating the executable and imposing a time limit to return collected information, the time available to perform reverse engineering may be effectively limited.

[00100] According to an example embodiment, an executable may be configured to return a hash. In this context, the hash (also referred to as a message digest) is a value generated from the collected information. The hash may be substantially smaller than the collected information itself, and may be generated by a formula in such a manner that it is unlikely that some other collected information will produce the same hash value.

[00101] The executable may also be configured to return a one-way hash, which is an algorithm or function that "turns" the collected information into a fixed-length value. As with other hash types, it may be nearly impossible to derive the original collected information from only the one-way hash value.

[00102] A one-way hash function may be used to create digital signatures, which is a digital code that may be used to uniquely identify the sender. Like a written signature, the purpose of a digital signature is to ensure that the individual sending the information really is who he or she claims to be. There are a number of different encryption techniques to provide this level of security.

[00103] The information collected by an executable may also be obfuscated, that is, the information collected for the purposes of authentication or authorization may be rendered more difficult to perceive to third parties (e.g., would-be listeners). Hence, if several pairs are sent to identify a given entity, each pair may return a different answer - even though the entity's identifying information has not changed. The elements used to obfuscate the information in the pair of executables may include, for example, functions, implementation, workflow. Other elements used during the challenge/response may include, for example, the dynamic selection of an encryption algorithm and/or communication protocol.

APPLE EXHIBIT 1102, Page 533 of 1048

[00104] Embodiments may also provide authentication despite a configuration change, including, for example, hardware changes. At times the hardware may require change due to, for example, a failure or a desire to swap components. For example, an internal CD-ROM drive may fail or a faster model may be desired. To accommodate the change, a threshold level of pass/fail authentication may be provided, which is configurable. In this regard, a level of identification strength may be specified for each hardware component so that the presence of certain hardware components (e.g., CPU) may provide a stronger basis for identification of an entity as compared to other hardware components (e.g., CD-ROM drive). Accordingly, limited changes in hardware may be flexibly accommodated by setting the threshold to an appropriate value (e.g., a required match of at least one strongly identifying hardware component or a required match of at least three weaker identifiers), whereas if an absolute match of hardware components is required the threshold level may be set to the highest level (e.g., 100% match of all hardware component identifiers). If during an authentication attempt, the specified threshold level is met, the hardware signature used to validate the identity of the requesting entity may be automatically updated to include the signatures of any newly configured components. If, however, during the authentication attempt, the specified threshold level is not met, it may be assumed that the requesting entity is not properly identified and therefore refused access.

[00105] Embodiments may also provide the ability to deny authentication. For example, authentication may be denied upon the detection of an unauthorized device (e.g., wireless local area network (LAN) card or a Universal Serial Bus (USB) key) or the presence of unauthorized software (e.g., virtual machine). Moreover, authentication may be denied despite identical or matching hardware and/or software identifiers. For example, authentication may be denied even if there is a 100% match of all hardware and software component identifiers.

[00106] In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of embodiments of the present invention. It will be evident, however, to one skilled in the art that embodiments of the present invention may be practiced without these specific details.

APPLE EXHIBIT 1102, Page 534 of 1048

Computer System

[00107] **Figure 6** shows a diagrammatic representation of a machine in the example form of a computer system within which a set of instructions, for causing the machine to perform any one or more of the methodologies discussed herein, may be executed. In alternative embodiments, the machine operates as a standalone device or may be connected (e.g., network) to other machines. In a network deployment, the machine may operate in the capacity of a server or a client user machine in server-client user network environment, or as a peer machine in a peer-to-peer (or distributed) network environment. The machine may be a server computer, a client user computer, a personal computer (PC), a tablet PC, a set-top box (STB), a Personal Digital Assistant (PDA), a cellular telephone, a mobile device, a palmtop computer, a laptop computer, a desktop computer, a personal digital assistant, a communications device, a wireless telephone, a land-line telephone, a control system, a camera, a scanner, a facsimile machine, a printer, a pager, a personal trusted device, a web appliance, a network router, switch or bridge, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by that machine.

[00108] Further, while a single machine is illustrated, the term "machine" shall also be taken to include any collection of machines that individually or jointly execute a set (or multiple sets) of instructions to perform any one or more of the methodologies discussed herein.

[00109] The example computer system 600 includes a processor 602 (e.g., a central processing unit (CPU), a graphics processing unit (GPU), or both), a main memory 604 and a static memory 606, which communicate with each other via a bus 608. The computer system 600 may further include a video display unit 610 (e.g., a liquid crystal display (LCD) or a cathode ray tube (CRT)). The computer system 600 also includes an input device 612 (e.g., a keyboard), a cursor control device 614 (e.g., a mouse), a disk drive unit 616, a signal generation device 618 (e.g., a speaker) and a network interface device 620.

[00110] The disk drive unit 616 includes a machine-readable medium 622 on which is stored one or more sets of instructions (e.g., software 624) embodying any one or more of the methodologies or functions described herein.

APPLE EXHIBIT 1102, Page 535 of 1048

The instructions 624 may also reside, completely or at least partially, within the main memory 604, the static memory 606, and/or within the processor 602 during execution thereof by the computer system 600. The main memory 604 and the processor 602 also may constitute machine-readable media.

[00111] The instructions 624 may further be transmitted or received over a network 626 via the network interface device 620.

[00112] Applications that may include the apparatus and systems of various embodiments broadly include a variety of electronic and computer systems. Some embodiments implement functions in two or more specific interconnected hardware modules or devices with related control and data signals communicated between and through the modules, or as portions of an application-specific integrated circuit. Thus, the example system is applicable to software, firmware, and hardware implementations.

[00113] While the machine-readable medium 622 is shown in an example embodiment to be a single medium, the term "machine-readable medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store the one or more sets of instructions. The term "machine-readable medium" shall also be taken to include any medium that is capable of storing, encoding or carrying a set of instructions for execution by the machine and that cause the machine to perform any one or more of the methodologies of the present invention. The term "machine-readable medium" shall accordingly be taken to include, but not be limited to, solid-state memories, optical and magnetic media, and carrier wave signals.

[00114] The illustrations of embodiments described herein are intended to provide a general understanding of the structure of various embodiments, and they are not intended to serve as a complete description of all the elements and features of apparatus and systems that might make use of the structures described herein. Many other embodiments will be apparent to those of skill in the art upon reviewing the above description. Other embodiments may be utilized and derived therefrom, such that structural and logical substitutions and changes may be made without departing from the scope of this disclosure. Figures 1 to 6 are merely representational and may not be drawn to scale. Certain proportions thereof may be exaggerated, while others may be minimized.

APPLE EXHIBIT 1102, Page 536 of 1048

Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

[00115] The following description includes terms, such as "up", "down", "upper", "lower", "first", "second", etc. that are used for descriptive purposes only and are not to be construed as limiting. The elements, materials, geometries, dimensions, and sequence of operations may all be varied to suit particular applications. Parts of some embodiments may be included in, or substituted for, those of other embodiments. While the foregoing examples of dimensions and ranges are considered typical, the various embodiments are not limited to such dimensions or ranges.

[00116] The Abstract is provided to comply with 37 C.F.R. §1.74(b) to allow the reader to quickly ascertain the nature and gist of the technical disclosure. The Abstract is submitted with the understanding that it will not be used to interpret or limit the scope or meaning of the claims.

[00117] In the foregoing Detailed Description, various features are grouped together in a single embodiment for the purpose of streamlining the disclosure. This method of disclosure is not to be interpreted as reflecting an intention that the claimed embodiments have more features than are expressly recited in each claim. Thus the following claims are hereby incorporated into the Detailed Description, with each claim standing on its own as a separate embodiment.

[00118] Thus, embodiments describe a method and a system to identify and/or authenticate an entity. Although embodiments of the present invention have been described with reference to specific example embodiments, it will be evident that various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of embodiments as expressed in the subjoined claims.

APPLE EXHIBIT 1102, Page 537 of 1048

CLAIMS

What is claimed is:

1. A system comprising:

a server-side executable to execute on a server to generate a first result;

a client-side executable to execute on an entity to generate a second result, the client-side executable being associated with the server-side executable, and to execute a subset of instructions from the server; and

an authenticator to compare the first result and the second result to verify an identity of the entity.

2. The system of claim 1 wherein the server includes the authenticator.

3. The system of claim 1 wherein the server and the authenticator are separate.

4. The system of claim 1 further comprising a server-side database accessible by the server-side executable, wherein the server-side database includes entity information used by the server-side executable in generating the first result.

5. The system of claim 4 wherein the server-side executable is to collect the entity information from the database.

6. The system of claim 5 wherein the server-side executable is to process the collected entity information from the database to generate the first result.

7. The system of claim 1 wherein the client-side executable is to collect entity information.

APPLE EXHIBIT 1102, Page 538 of 1048

8. The system of claim 7 wherein the collected information is selected from a group including hardware-related information and software-related information associated with the entity.

9. The system of claim 7 wherein the second result includes a fixed-length value generated from the collected information.

10. The system of claim 7 wherein the client-side executable is to process the collected entity information to generate the second result.

11. The system of claim 1 wherein the authenticator includes a time limit expiration to render the entity as not identified if the second result is not received by the authenticator during a limited time frame.

12. The system of claim 1 wherein the client side executable includes one-way function executable software.

13. The system of claim 1 wherein the client-side executable includes at least one dynamically generated instruction.

14. The system of claim 1 wherein the server-side executable includes at least one dynamically generated instruction.

15. The system of claim 1 wherein the client-side executable includes at least one previously generated instruction and the server-side executable includes at least one previously generated instruction.

16. The system of claim 15 wherein the at least one previously generated instruction includes a dynamic selection from a plurality of previously generated instructions on the server.

17. The system of claim 1 wherein the entity includes a device, wherein the device has an associated authentication score to be added towards a threshold level.

APPLE EXHIBIT 1102, Page 539 of 1048

18. The system of claim 17 wherein the authentication score is higher for associated devices that are less likely to be exchanged or adjusted.

19. The system of claim 1 wherein the entity includes a plurality of components, each component having a level of identification strength.

20. A method comprising:

dynamically generating a client-side executable and a corresponding server-side executable;

executing the server-side executable on a server to generate a first result;

executing the client-side executable on an entity to generate a second result; and

comparing the first result and the second result to verify an identity of the entity.

21. The method of claim 20 further comprising collecting information, selected from a group including a dynamic information type of the entity, a dynamic information order of the entity and a dynamic amount of information of the entity, for the client-side executable.

22. The method of claim 20 further comprising collecting information in the entity by the client-side executable; and generating a fixed-length value from the information collected.

23. The method of claim 20 further comprising collecting information about the entity by the client-side executable; and processing the information collected to generate the second result, wherein the client-side executable includes at least one dynamic encryption instruction.

24. The method of claim 20 further comprising collecting information about the entity by the client-side executable and obfuscating the information collected.

APPLE EXHIBIT 1102, Page 540 of 1048

25. The method of claim 20 further comprising collecting, from a server database, entity information for the server-side executable.

26. The method of claim 20 wherein the server includes an authenticator to compare the first result and the second result.

27. The method of claim 26 further comprising imposing a time limit on receipt of the second result by the authenticator, wherein upon expiration of the time limit, the entity is considered as not authenticated.

28. The method of claim 20 further comprising generating a plurality of instructions on the server; and dynamically selecting a subset of the plurality of instructions as the client-side executable.

29. The method of claim 20 wherein the client-side executable and the server-side executable each include a dynamic sending instruction to send the first result and the second result, respectively.

30. The method of claim 20 wherein the entity includes a plurality of components, each component having a level of identification strength.

31. The method of claim 20 wherein the entity includes a device, wherein the device has an associated authentication score to be added towards a threshold level.

32. The method of claim 31 wherein the authentication score is higher for associated devices that are less likely to be exchanged or adjusted.

33. The method of claim 31 further comprising:

collecting information from a server database, wherein the information collected includes a corresponding image of the device; and

adding the authentication score towards a threshold level to verify the entity.

APPLE EXHIBIT 1102, Page 541 of 1048

34. The method of claim 33 further comprising:

updating a hardware signature used to validate the entity to include a signature associated with a new component to the entity, when the threshold level is met.

35. The method of claim 33 further comprising refusing access to the entity when the threshold level is not met.

36. The method of claim 33 further comprising denying authentication to the entity upon detection of an unauthorized device or of unauthorized software.

37. A machine-readable medium storing a sequence of instructions that, when executed by a computer, cause the computer to perform the method of claim 20.

38. A system comprising:

means for dynamically generating a client-side executable and a corresponding server-side executable;

means for executing the server-side executable on a server to generate a first result and for executing the client-side executable on an entity to generate a second result; and

means for comparing the first result and the second result to verify an identity of the entity.

APPLE EXHIBIT 1102, Page 542 of 1048

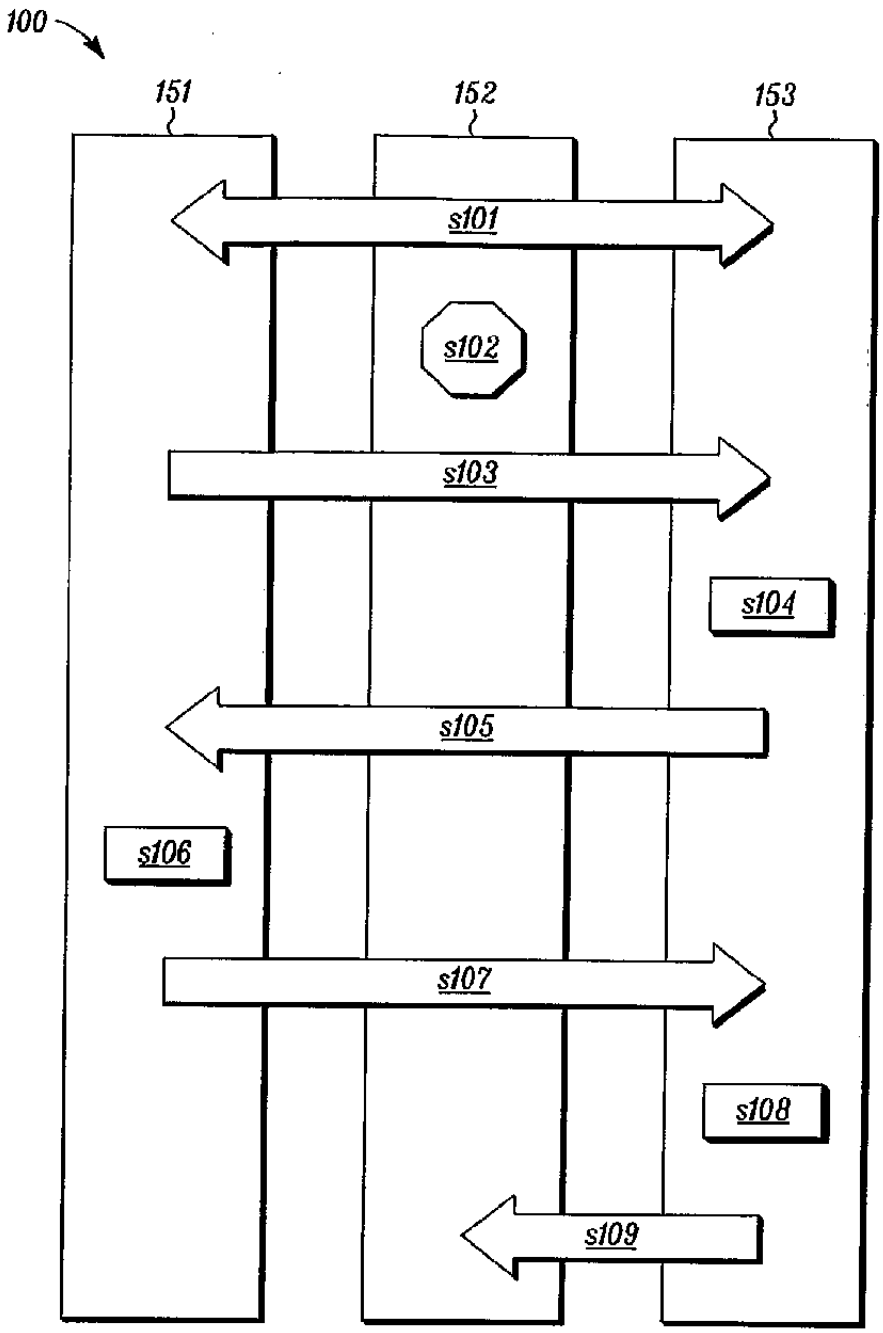


FIG. 1A

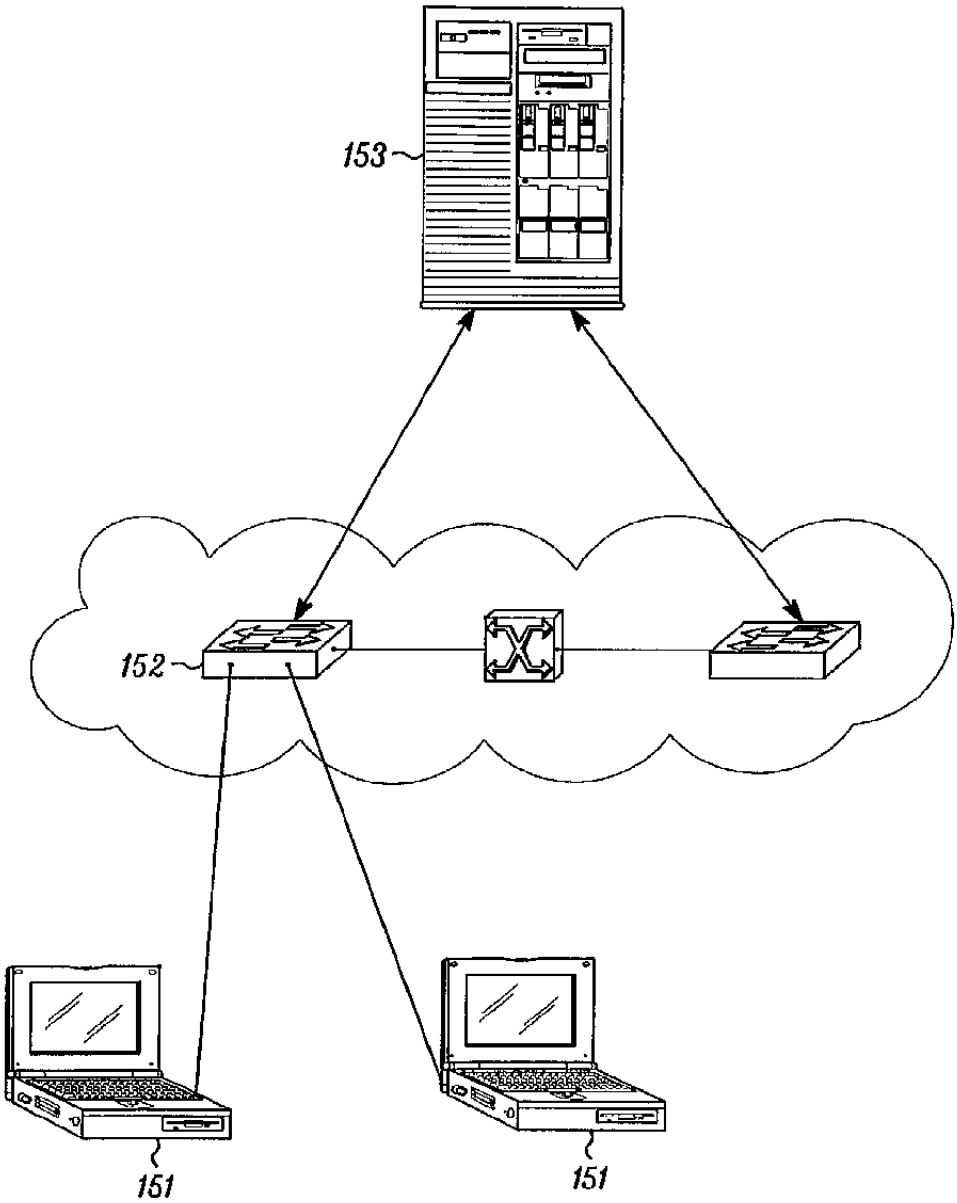


FIG. 1B

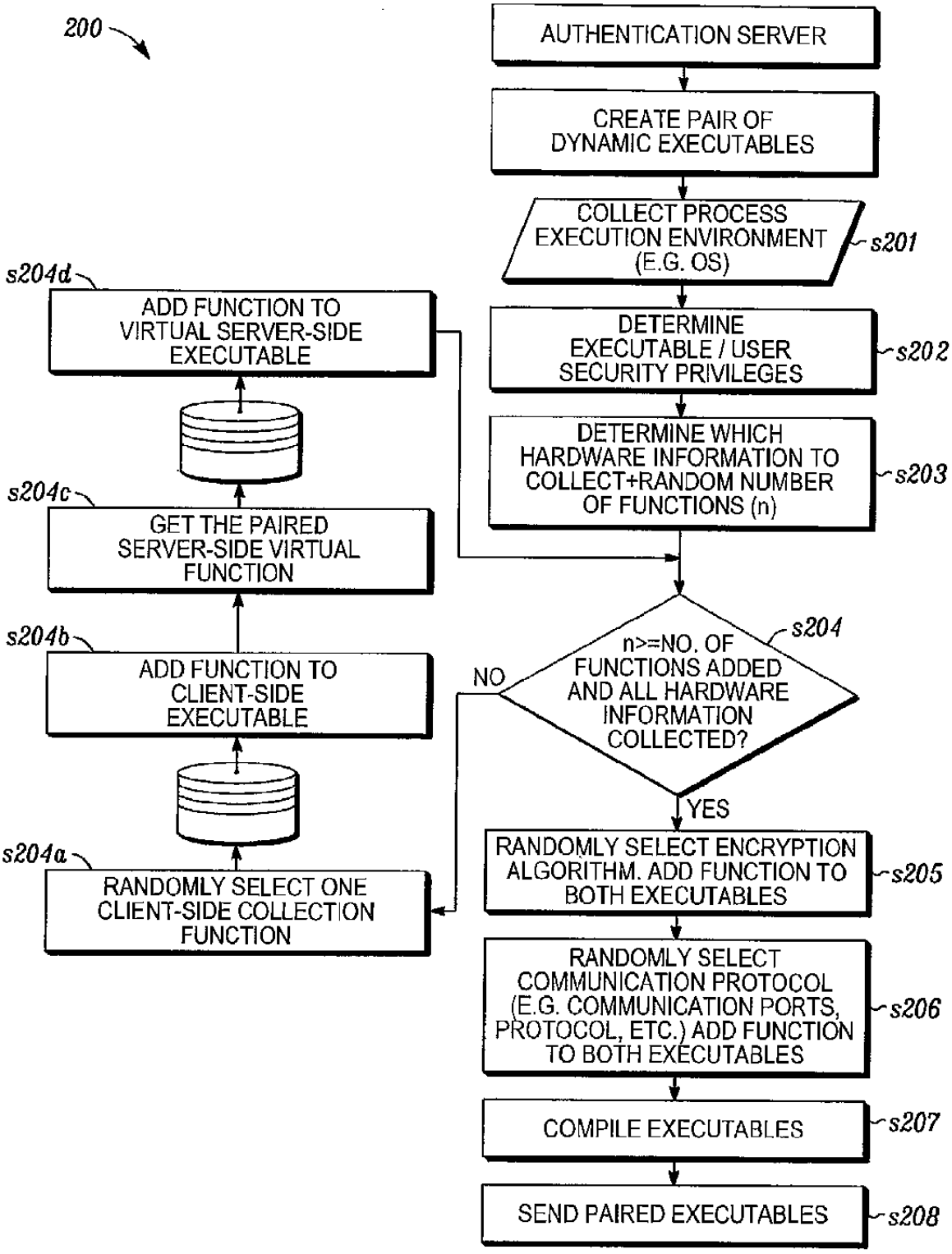

200
AUTHENTICATION SERVER
CREATE PAIR OF DYNAMIC EXECUTABLES
COLLECT PROCESS EXECUTION ENVIRONMENT (E.G. OS)
s201
DETERMINE EXECUTABLE / USER SECURITY PRIVILEGES
s202
DETERMINE WHICH HARDWARE INFORMATION TO COLLECT+RANDOM NUMBER OF FUNCTIONS (n)
s203
n>=NO. OF FUNCTIONS ADDED AND ALL HARDWARE INFORMATION COLLECTED?
s204
NO
YES
RANDOMLY SELECT ENCRYPTION ALGORITHM. ADD FUNCTION TO BOTH EXECUTABLES
s205
RANDOMLY SELECT COMMUNICATION PROTOCOL (E.G. COMMUNICATION PORTS, PROTOCOL, ETC.) ADD FUNCTION TO BOTH EXECUTABLES
s206
COMPILE EXECUTABLES
s207
SEND PAIRED EXECUTABLES
s208
s204a
RANDOMLY SELECT ONE CLIENT-SIDE COLLECTION FUNCTION
s204b
ADD FUNCTION TO CLIENT-SIDE EXECUTABLE
s204c
GET THE PAIRED SERVER-SIDE VIRTUAL FUNCTION
s204d
ADD FUNCTION TO VIRTUAL SERVER-SIDE EXECUTABLE

FIG. 2A

253

```
004010CC  mov     esi, esp
004010CE  push    0
004010D0  push    0
004010D2  push    0
004010D4  lea     eax, [ebp-1Ch]
004010D7  push    eax
004010D8  call    dword ptr [ _imp_GetMessageA@16 (0042a350)]
004010DE  cmp     esi, esp
004010E0  call    _chkesp (00401710)
004010E5  test    eax, eax
004010E7  je      WinMain+0F0h (00401130)
44:       {
45:           if(!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
004010E9  mov     esi, esp
004010EB  lea     ecx, [ebp-1Ch]
004010EE  push    ecx
004010EF  mov     edx, dword ptr [ebp-20h]
004010F2  push    edx
004010F3  mov     eax, dword ptr [ebp-20h]
004010F6  push    eax
004010F7  call    dword ptr [ _imp_TranslateAcceleratorA@12 (0042a354) ]
004010FD  cmp     esi, esp
004010FF  call    _chkesp (00401710)
00401104  test    eax, eax
00401106  jne     WinMain+0EEh (0040112e)
46:           {
47:               TranslateMessage(&msg);
00401108  mov     esi, esp
0040110A  lea     ecx, [ebp-1Ch]
0040110D  push    ecx
0040110E  call    dword ptr [ _imp_TranslateMessage@4 (0042a358)]
00401114  cmp     esi, esp
00401116  call    _chkesp (00401710)
48:           DispatchMessage(&msg);
0040111B  mov     esi, esp
0040111D  lea     edx, [ebp-1Ch]
00401120  push    edx
00401121  call    dword ptr [ _imp_DispatchMessageA@4 (0042a35c)]
```

FIG. 2B

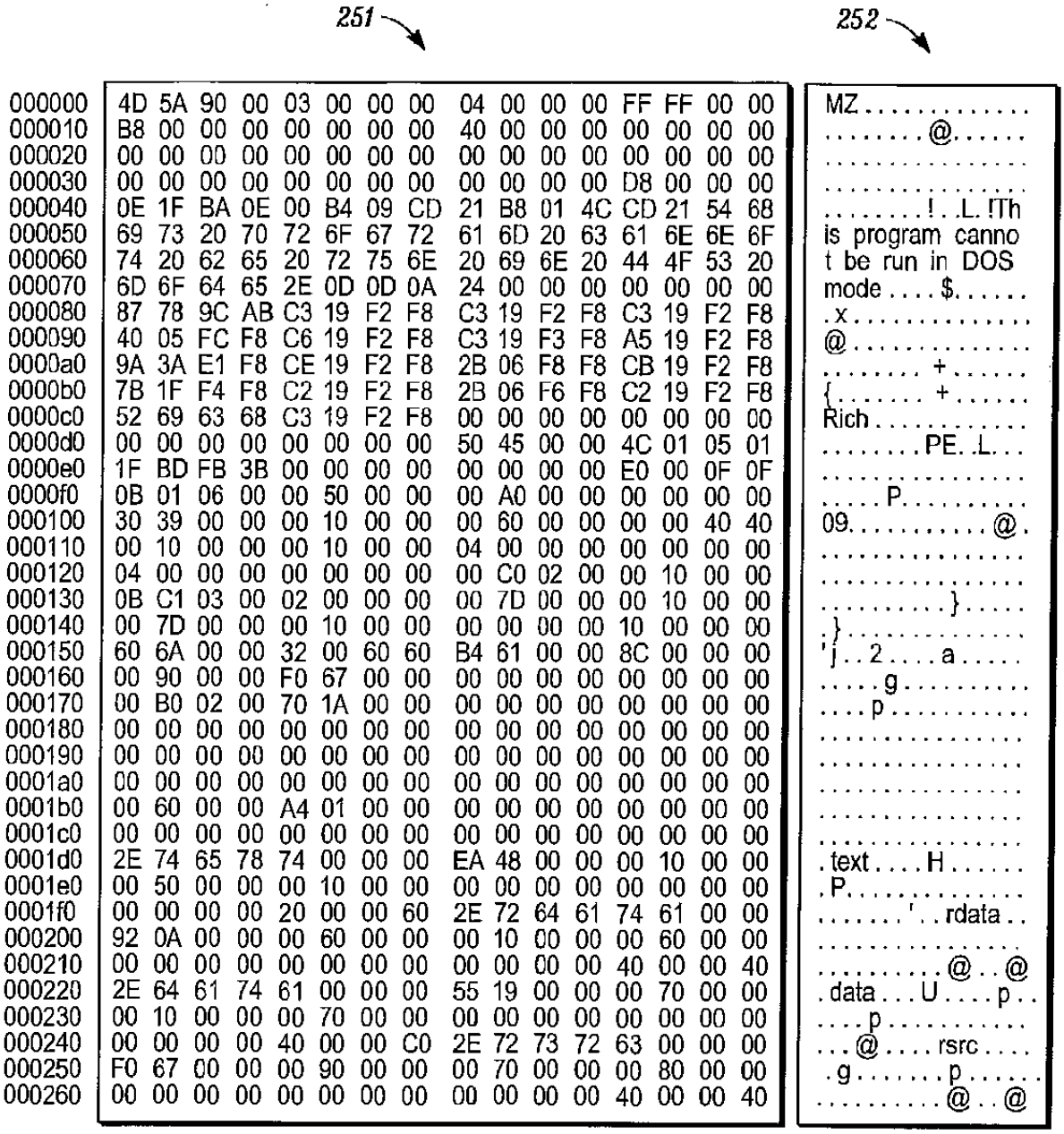

251
252
000000 4D 5A 90 00 03 00 00 00 04 00 00 00 FF FF 00 00 MZ
000010 B8 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 @
000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000030 00 00 00 00 00 00 00 00 00 00 00 00 D8 00 00 00
000040 0E 1F BA 0E 00 B4 09 CD 21 B8 01 4C CD 21 54 68 !..L.!Th
000050 69 73 20 70 72 6F 67 72 61 6D 20 63 61 6E 6E 6F is program canno
000060 74 20 62 65 20 72 75 6E 20 69 6E 20 44 4F 53 20 t be run in DOS
000070 6D 6F 64 65 2E 0D 0D 0A 24 00 00 00 00 00 00 00 mode....$
000080 87 78 9C AB C3 19 F2 F8 C3 19 F2 F8 C3 19 F2 F8 .x
000090 40 05 FC F8 C6 19 F2 F8 C3 19 F3 F8 A5 19 F2 F8 @
0000a0 9A 3A E1 F8 CE 19 F2 F8 2B 06 F8 F8 CB 19 F2 F8 +
0000b0 7B 1F F4 F8 C2 19 F2 F8 2B 06 F6 F8 C2 19 F2 F8 {+
0000c0 52 69 63 68 C3 19 F2 F8 00 00 00 00 00 00 00 00 Rich
0000d0 00 00 00 00 00 00 00 00 50 45 00 00 4C 01 05 01 PE..L
0000e0 1F BD FB 3B 00 00 00 00 00 00 00 00 E0 00 0F 0F
0000f0 0B 01 06 00 00 50 00 00 00 A0 00 00 00 00 00 00 P
000100 30 39 00 00 00 10 00 00 00 60 00 00 00 00 40 40 09@@
000110 00 10 00 00 00 10 00 00 04 00 00 00 00 00 00 00
000120 04 00 00 00 00 00 00 00 00 C0 02 00 00 10 00 00
000130 0B C1 03 00 02 00 00 00 00 7D 00 00 00 10 00 00 }
000140 00 7D 00 00 00 10 00 00 00 00 00 00 10 00 00 00 }
000150 60 6A 00 00 32 00 60 60 B4 61 00 00 8C 00 00 00 `j2`a
000160 00 90 00 00 F0 67 00 00 00 00 00 00 00 00 00 00 g
000170 00 B0 02 00 70 1A 00 00 00 00 00 00 00 00 00 00 p
000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001a0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001b0 00 60 00 00 A4 01 00 00 00 00 00 00 00 00 00 00
0001c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0001d0 2E 74 65 78 74 00 00 00 EA 48 00 00 00 10 00 00 .text....H
0001e0 00 50 00 00 00 10 00 00 00 00 00 00 00 00 00 00 P
0001f0 00 00 00 00 20 00 00 60 2E 72 64 61 74 61 00 00 `.rdata
000200 92 0A 00 00 00 60 00 00 00 10 00 00 00 60 00 00
000210 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 @@
000220 2E 64 61 74 61 00 00 00 55 19 00 00 00 70 00 00 .data...U....p
000230 00 10 00 00 00 70 00 00 00 00 00 00 00 00 00 00 p
000240 00 00 00 00 40 00 00 C0 2E 72 73 72 63 00 00 00 @....rsrc
000250 F0 67 00 00 00 90 00 00 00 70 00 00 00 80 00 00 g.....p
000260 00 00 00 00 00 00 00 00 00 00 00 00 40 00 00 40 @@

FIG. 2C

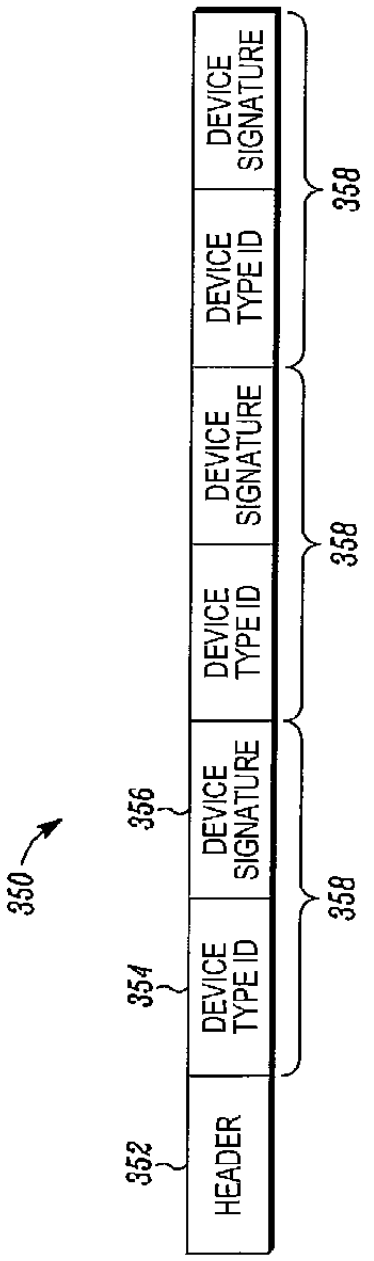

350
352
HEADER
354
DEVICE TYPE ID
356
DEVICE SIGNATURE
DEVICE TYPE ID
DEVICE SIGNATURE
DEVICE TYPE ID
DEVICE SIGNATURE
358
358
358

FIG. 3A

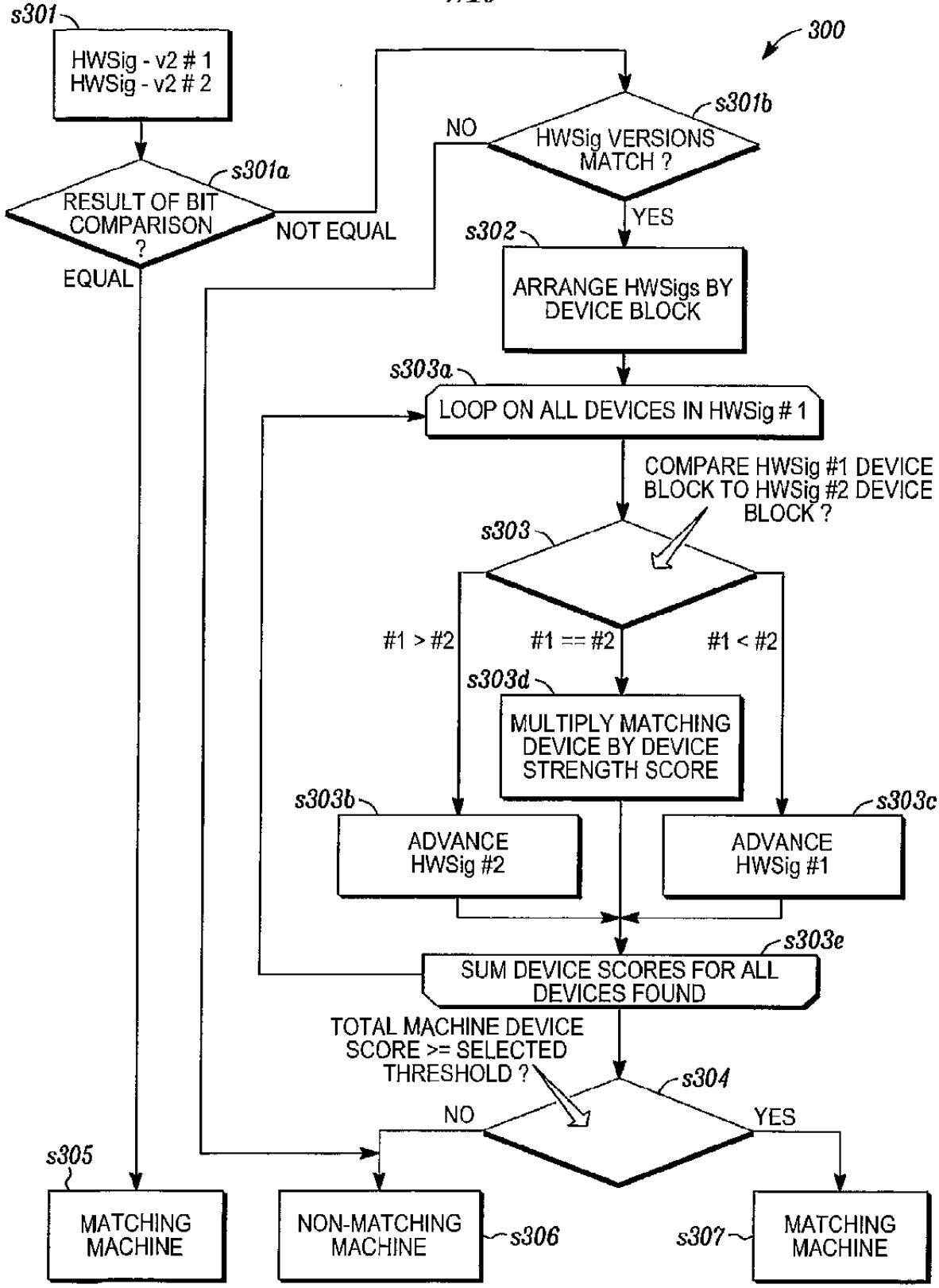

s301
HWSig - v2 # 1
HWSig - v2 # 2
300
s301b
HWSig VERSIONS MATCH ?
NO
s301a
RESULT OF BIT COMPARISON ?
NOT EQUAL
YES
EQUAL
s302
ARRANGE HWSigs BY DEVICE BLOCK
s303a
LOOP ON ALL DEVICES IN HWSig # 1
COMPARE HWSig #1 DEVICE BLOCK TO HWSig #2 DEVICE BLOCK ?
s303
#1 > #2
#1 == #2
#1 < #2
s303d
MULTIPLY MATCHING DEVICE BY DEVICE STRENGTH SCORE
s303b
ADVANCE HWSig #2
s303c
ADVANCE HWSig #1
s303e
SUM DEVICE SCORES FOR ALL DEVICES FOUND
TOTAL MACHINE DEVICE SCORE >= SELECTED THRESHOLD ?
s304
NO
YES
s305
MATCHING MACHINE
NON-MATCHING MACHINE
s306
s307
MATCHING MACHINE

FIG. 3B

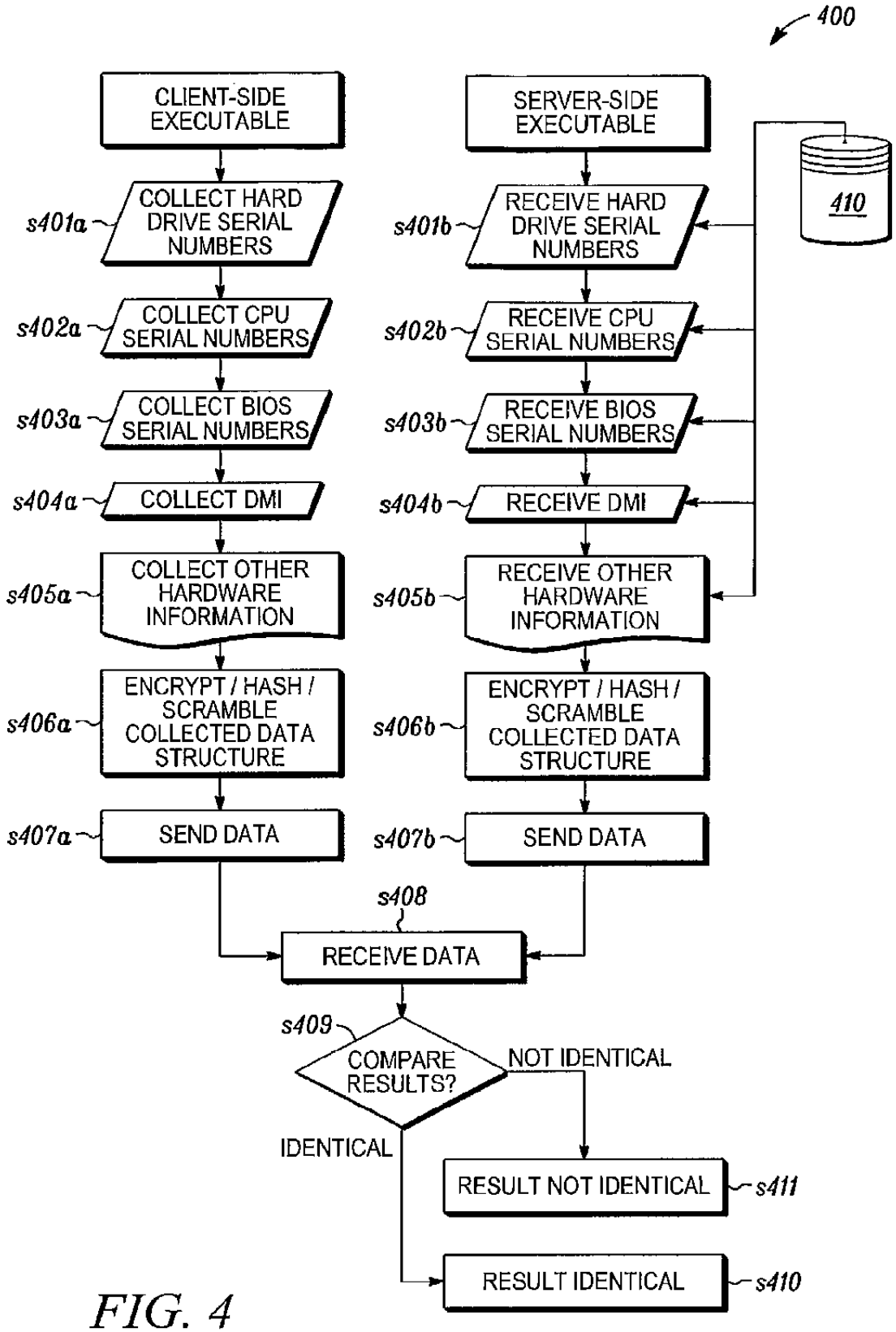

400
CLIENT-SIDE EXECUTABLE
SERVER-SIDE EXECUTABLE
410
s401a
COLLECT HARD DRIVE SERIAL NUMBERS
s401b
RECEIVE HARD DRIVE SERIAL NUMBERS
s402a
COLLECT CPU SERIAL NUMBERS
s402b
RECEIVE CPU SERIAL NUMBERS
s403a
COLLECT BIOS SERIAL NUMBERS
s403b
RECEIVE BIOS SERIAL NUMBERS
s404a
COLLECT DMI
s404b
RECEIVE DMI
s405a
COLLECT OTHER HARDWARE INFORMATION
s405b
RECEIVE OTHER HARDWARE INFORMATION
s406a
ENCRYPT / HASH / SCRAMBLE COLLECTED DATA STRUCTURE
s406b
ENCRYPT / HASH / SCRAMBLE COLLECTED DATA STRUCTURE
s407a
SEND DATA
s407b
SEND DATA
s408
RECEIVE DATA
s409
COMPARE RESULTS?
NOT IDENTICAL
IDENTICAL
RESULT NOT IDENTICAL
s411
RESULT IDENTICAL
s410

FIG. 4

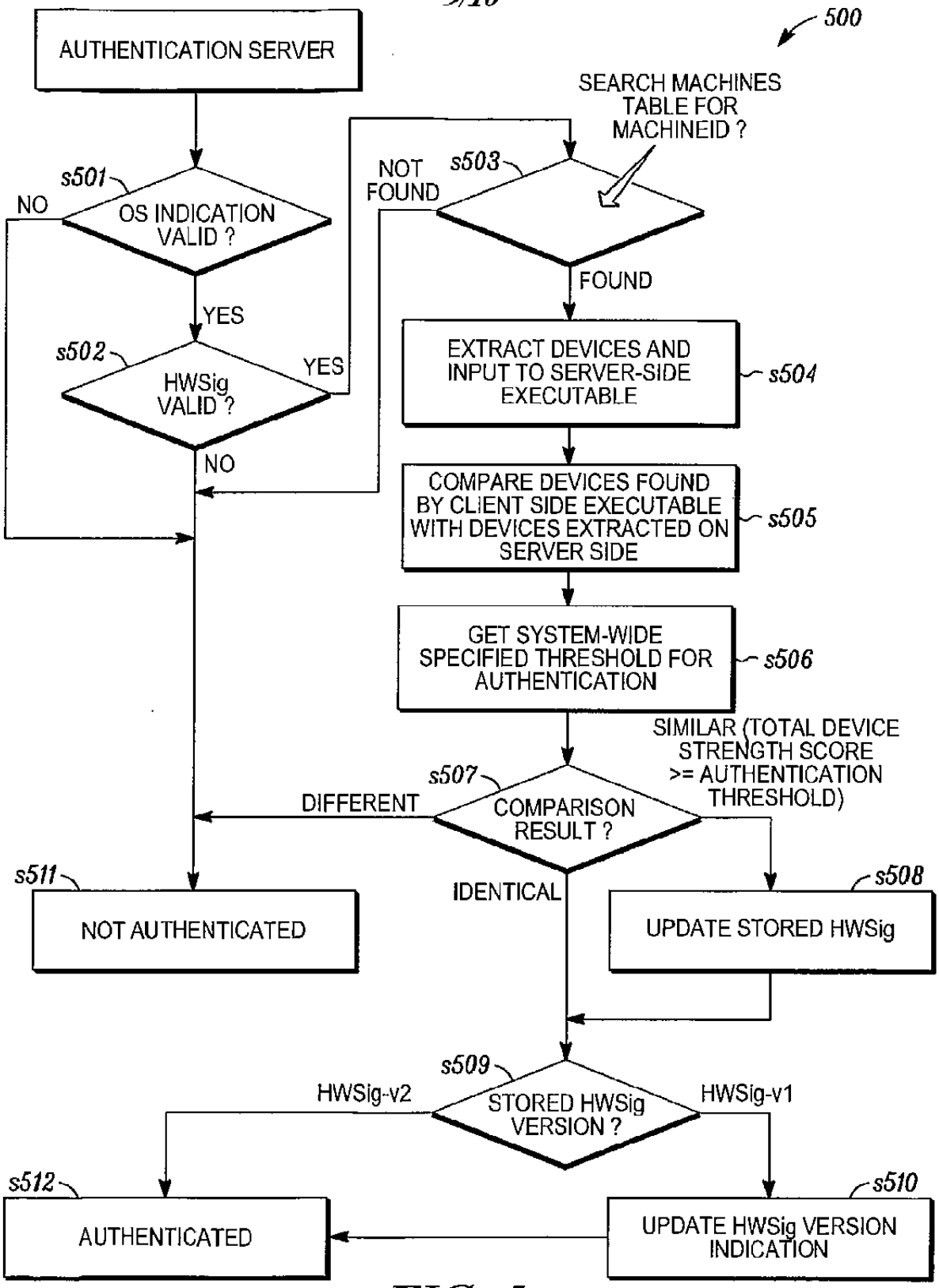

500
AUTHENTICATION SERVER
SEARCH MACHINES TABLE FOR MACHINEID ?
s501
OS INDICATION VALID ?
NO
YES
s502
HWSig VALID ?
YES
NO
s503
NOT FOUND
FOUND
EXTRACT DEVICES AND INPUT TO SERVER-SIDE EXECUTABLE
s504
COMPARE DEVICES FOUND BY CLIENT SIDE EXECUTABLE WITH DEVICES EXTRACTED ON SERVER SIDE
s505
GET SYSTEM-WIDE SPECIFIED THRESHOLD FOR AUTHENTICATION
s506
s507
COMPARISON RESULT ?
DIFFERENT
SIMILAR (TOTAL DEVICE STRENGTH SCORE >= AUTHENTICATION THRESHOLD)
IDENTICAL
s511
NOT AUTHENTICATED
s508
UPDATE STORED HWSig
s509
STORED HWSig VERSION ?
HWSig-v2
HWSig-v1
s512
AUTHENTICATED
s510
UPDATE HWSig VERSION INDICATION

FIG. 5

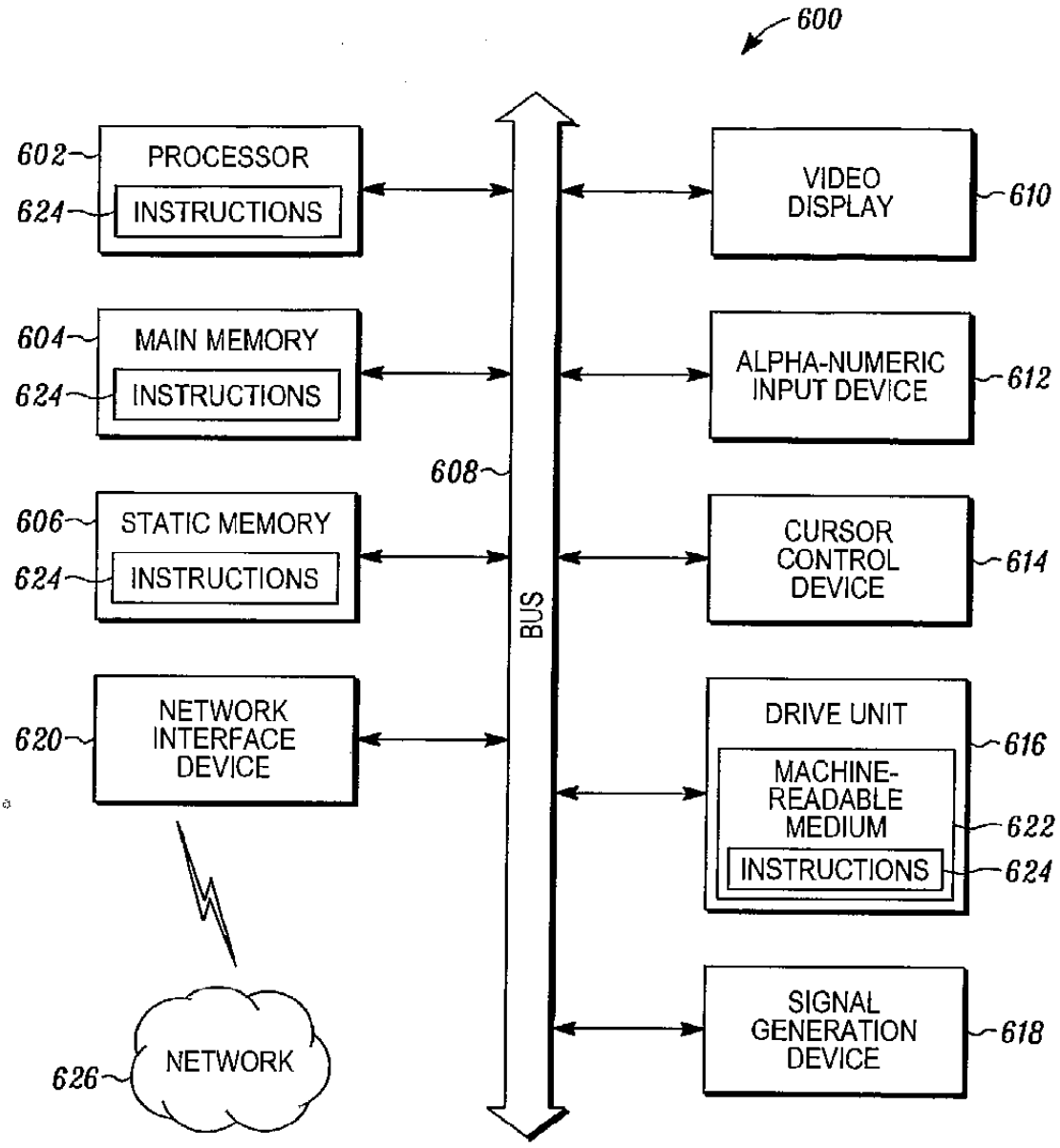

600
602
PROCESSOR
624
INSTRUCTIONS
604
MAIN MEMORY
624
INSTRUCTIONS
606
STATIC MEMORY
624
INSTRUCTIONS
620
NETWORK INTERFACE DEVICE
626
NETWORK
608
BUS
VIDEO DISPLAY
610
ALPHA-NUMERIC INPUT DEVICE
612
CURSOR CONTROL DEVICE
614
DRIVE UNIT
616
MACHINE-READABLE MEDIUM
622
INSTRUCTIONS
624
SIGNAL GENERATION DEVICE
618

FIG. 6

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 May 2007 (31.05.2007)

PCT

(10) International Publication Number
WO 2007/060516 A2

(51) International Patent Classification: Not classified

(21) International Application Number:
PCT/IB2006/003275

(22) International Filing Date: 25 August 2006 (25.08.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/212,312 26 August 2005 (26.08.2005) US

(71) Applicant and
(72) Inventor: LO, Teddy, Yeung, Man [GB/CN]; 19 Kent Road, Kowloon Tong, Hong Kong (CN).

(74) Agent: MAIWALD PATENTANWALTS GMBH; Elisenhof, Elisenstrasse 3, 80335 München (DE).

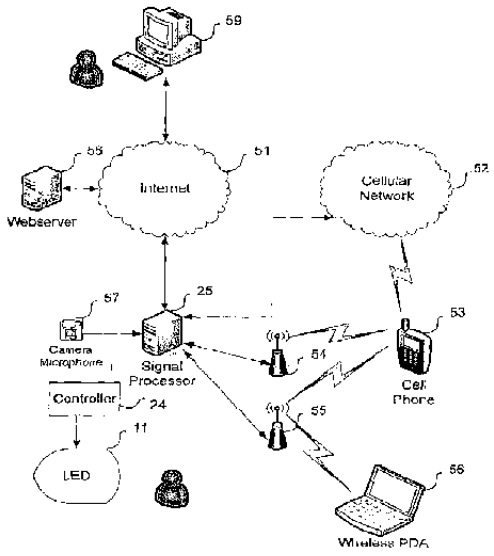
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: INTERACTIVE BULLETIN BOARD SYSTEM AND METHOD



(57) Abstract: A messaging board and a traffic light apparatus comprising a primary traffic panel including a first matrix of multicolored LEDs for directing motor traffic by displaying a plurality of images; a message panel including a second matrix of multicolored LEDs for displaying information unrelated to the directing of traffic; and a personal device interface providing two way communication of information between the traffic light apparatus and at least one local user with a personal portable electronic device is disclosed. The messaging board is capable of being used as a communications station, as well as an information terminal and/or point-of-sales station.

WO 2007/060516 A2

INTERACTIVE BULLETIN BOARD SYSTEM AND METHOD

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This PCT application is a continuation-in-part of U.S. Patent Application Serial No. 11/212,312, titled LED TRAFFIC LIGHT, filed August 26, 2005, the contents of which is incorporated herein by reference into the present application.

[0002] Throughout this application, various references may be cited. Disclosure of these references in their entirety is hereby incorporated by reference into this application to more fully describe the state of the art to which the present invention pertains.

FIELD OF THE INVENTION

[0003] The present invention relates to the field of traffic lights and messaging boards and, more particularly, to an LED messaging board.

BACKGROUND OF THE INVENTION

[0004] Traffic lights are ubiquitous from city streets to country roads. Traffic lights traditionally in the United States consist of three lights: green signifying traffic may pass, yellow signifying that traffic will soon come to a stop, and red signifying traffic must stop. These lights are generally vertically aligned, one on top of the other. In this way, traffic lights control traffic at an intersection. In other locations different conventions may be adopted, such as flashing yellow meaning traffic may pass.

[0005] Some conventional traffic lights are illuminated using incandescent light bulbs. Incandescent bulbs tend to use relatively large amounts of electricity and require periodic replacement as the bulbs burn out. These factors make the operation of incandescent traffic lights relatively expensive.

[0006] More recently, traffic lights are being illuminated using light emitting diodes (LEDs). LEDs provide a source of light that has relatively low energy consumption, and the do not burn out as easily as light bulbs.

[0007] While conventional LED traffic lights are less expensive to operate, however,

APPLE EXHIBIT 1102, Page 554 of 1048

they maintain the conventional approach of three lights, vertically aligned, one on top of the other. While this approach has been used for a very long time, the use of a yellow light to signify that traffic will soon come to a stop has inherent problems. Specifically, motorists approaching a yellow traffic light do not know how much longer the traffic light will continue to stay yellow. These motorists may then unnecessarily speed up creating a safety risk for other motorists and pedestrians or come to a stop unnecessarily abruptly, thereby, creating a safety risk for other motorists.

[0008] Additionally, there is a need for quick dissemination of important information to motorists and pedestrians alike. This information may concern traffic conditions ahead or it may concern instructions disseminated in the event of an emergency. Because of the ubiquity of traffic lights, they are well suited for the conveyance of important information. Conventional traffic lights, however, fail to live up to fulfilling this objective.

[0009] Furthermore, the ubiquity of traffic lights make them well suited for the conveyance of advertisements that can be tailored to the particular community in which the traffic light is located. The use of advertisements on traffic lights can deliver an aesthetically pleasing futuristic look and be an important source of additional revenue for town and city governments that are increasingly under financial pressure. Conventional traffic lights fail to live up to fulfilling these objectives as well.

[0010] Conventional bulletin boards may statically advertise products or services. Traditionally, contact information, including street addresses where such products or services could be obtained or telephone numbers where transactions could be conducted, were displayed. At present, advertisements have moved to an electronic venue, where large, television-like appliances provide information to large groups of persons, including web sites and text message numbers and codes. These device are in wide public view, however, the public has not been provided the means to interact with such display devices. Thus, there is a need for an interactive messaging board.

SUMMARY OF THE INVENTION

[0011] A messaging board and a traffic light apparatus comprising a primary traffic

APPLE EXHIBIT 1102, Page 555 of 1048

panel including a first matrix of multicolored LEDs for directing motor traffic by displaying a plurality of images; a message panel including a second matrix of multicolored LEDs for displaying information unrelated to the directing of traffic; and a personal device interface providing two-way communication of information between the traffic light apparatus and at least one local user with a personal portable electronic device is disclosed. The messaging board is capable of being used as a communications station, as well as an information terminal and/or point-of-sales station.

[0012] In one aspect, the present invention is directed to an interactive bulletin board system, comprising: a signal processor coupled to a computer communications network; a messaging board coupled to said signal processor, said messaging board providing visual indicia responsive to said signal processor; and a personal portable electronic device wirelessly coupled to said computer communications network; wherein said signal processor updates said visual indicia displayed on said messaging board in response to a message sent from said personal portable electronic device to said signal processor.

[0013] In another aspect of the present invention, said messaging board comprises a matrix of multicolored LEDs.

[0014] In another aspect of the present invention, the system further comprises a traffic panel including a first matrix of multicolored LEDS for directing motor traffic by displaying a plurality of images.

[0015] In another aspect of the present invention, said visual indicia represents information requested through the personal portable electronic device.

[0016] In another aspect of the present invention, said information is obtained from the Internet.

[0017] In another aspect of the present invention, said signal processor authenticates said personal portable electronic device before updating said visual indicia.

[0018] In another aspect of the present invention, said personal portable electronic device comprises a device selected from the group comprising a cellular phone, a

APPLE EXHIBIT 1102, Page 556 of 1048

wireless laptop, a handheld email browser, an MP3 player and a digital camera.

[0019] In another aspect of the present invention, the system further comprises a camera coupled to said signal processor, wherein said visual indicia comprises images provided by said camera.

[0020] In another aspect of the present invention, said signal processor responds to messages sent from more than one personal portable electronic device.

[0021] In another aspect of the present invention, said signal processor responds sequentially to messages received from more than one personal portable electronic device.

[0022] In another aspect of the present invention, said signal processor responds to messages received from only a single personal portable electronic device until an ending condition is met.

[0023] In another aspect, the present invention is directed to a method of displaying visual indicia on a public display, the method comprising: a processor receiving a message via a computer communications network from a personal portable electronic device wirelessly coupled to said computer communications network; and the processor updating visual indicia displayed on said display in response to said message.

[0024] In another aspect of the present invention, the method further comprises authenticating said personal portable electronic device.

[0025] In another aspect of the present invention, the method further comprises the processor queuing messages received from said computer communications network.

[0026] In another aspect of the present invention, the method further comprises sending information to said personal portable electronic device in response to said message.

[0027] In another aspect of the present invention, said information is one or more selected from the group comprising a digital image, a series of digitized images and digitized audio.

APPLE EXHIBIT 1102, Page 557 of 1048

[0028] In another aspect of the present invention, said message is one or more selected from the group comprising a text message and an email.

[0029] In another aspect of the present invention, the method further comprises sending information to a web server.

[0030] In another aspect of the present invention, said information is one or more selected from the group comprising a digital image, a series of digitized images and digitized audio.

[0031] In another aspect of the present invention, the method further comprises the processor updating visual indicia in response to one or more messages received only from a single personal portable electronic device.

[0032] In another aspect of the present invention, the method further comprises the processor updating said visual indicia in response to a queued message sent from a second personal portable electronic device after an ending condition is met.

BRIEF DESCRIPTION OF THE DRAWINGS

[0033] FIG. 1 shows an LED traffic light according to an embodiment of the present invention;

[0034] FIG. 2 is a block diagram of an LED control apparatus according to an embodiment of the present invention;

[0035] FIGs. 3A, 3B and 3C show primary LED traffic panel schemes according to an embodiment of the present invention;

[0036] FIG. 4 shows another embodiment of LED traffic light according to the present invention;

[0037] FIG. 5 is a schematic diagram that illustrates an embodiment of the invention for additional uses of a messaging board; and

[0038] FIG. 6 is a flow chart that illustrates an method for interacting with a messaging board.

APPLE EXHIBIT 1102, Page 558 of 1048

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

[0039] The LED traffic light according to the present invention is an LED illuminated traffic light that is capable communicating to motorists approximately how much time remains before the traffic light changes from a signal directing motorists to proceed to a signal directing motorists to stop. Additionally, the LED traffic light according to the present disclosure is capable of disseminating important information such as traffic conditions and emergency instructions. Furthermore, the LED traffic light according to the present invention is also capable of displaying animated or still advertisements.

[0040] The LED traffic light according to the present disclosure can be vertically or horizontally aligned.

[0041] FIG. 1 illustrates a vertically aligned LED traffic light, in which the body 10 of the LED traffic light is a column or pillar, preferably a rectangular prism, and more preferably a rectangular prism with a square base and top and rectangular lateral sides. Preferably, the lateral sides are approximately 15 feet tall and the sides of the square top and bottom are approximately 15 inches wide. A base 16 can be attached to the bottom of the column 10 to increase the stability of the column. The base may be attached to the ground or floor, for example, by bolts, not shown. The base may also be attached to a foundation, for example, a cement foundation set below ground level.

[0042] The column 10 preferably has four lateral sides, but other configurations are possible such as a cylindrical prism having one continuous side or a triangular prism having three. The column must have one or more active surfaces. Active surfaces are lateral sides that contain LED panels. The column 10 has four active surfaces allowing the LED traffic light to control traffic in four directions at a four-way intersection.

[0043] According to one preferred embodiment of the present invention, mounted to each active surface of the column 10 is one primary LED traffic panel 14, one street sign receiving aperture 13, one pedestrian LED traffic panel 12 and one message LED panel 11. The invention is not limited to having one of each element, and various combinations are possible depending on the needs of the environment where the column 10 is installed.

APPLE EXHIBIT 1102, Page 559 of 1048

[0044] The primary LED traffic panel 14 is preferably a low-resolution multicolored LED panel. This panel may also be formed from multiple smaller low-resolution multi-colored LED panels assembled together. For example, the primary LED traffic panel 14 can be constructed from combining 18 (9 rows of 2) light emitting diode dot matrix modules known as the 833 Intelligent Module (IM) manufactured by Desay Optotech Ltd. of Huizhou, China. The primary LED traffic panel 14 is responsible for controlling motor traffic and functions as a replacement for the conventional traffic light. The primary LED traffic panel 14 directs traffic by displaying an image on the panel's surface. The specific images that are displayed depend on the primary traffic panel display scheme used. Examples of primary traffic panel display schemes are described in detail below.

[0045] The street sign receiving aperture 13 allows for the attachment of a street sign 15. When a street sign 15 is attached, the LED traffic light also functions as a street sign. The street sign 15 may be a conventional pre-printed sign board or the street sign 15 may be an LED panel capable of displaying any programmed street name or other information like cross streets or building numbers. Even in embodiments of the present invention where the column 10 has four active surfaces, two street signs orthogonally oriented may be sufficient to convey street names.

[0046] The pedestrian LED traffic panel 12 is preferably a low-resolution LED panel, formed as a multicolored LED panel. This panel may also be formed from multiple smaller low-resolution multicolored LED panels combined in a single unit. For example, the pedestrian LED traffic panel 14 can be constructed by combining 6 (3 rows of 2) light emitting diode dot matrix modules known as the 0833IM manufactured by Desay Optotech Ltd. of Huizhou, China. The pedestrian LED traffic panel 12 is responsible for directing pedestrian traffic and functions as a replacement for the conventional "walk/don't walk" pedestrian traffic signal. The pedestrian LED traffic panel 12 directs pedestrian traffic by displaying an image on the panel's surface. The specific images that are displayed depend on the pedestrian traffic panel display scheme used. Examples of pedestrian traffic panel display schemes are described in detail below.

[0047] The message LED panel 11 is preferably a high-resolution LED panel, such as a multicolored LED panel. This panel may also be formed from multiple smaller

APPLE EXHIBIT 1102, Page 560 of 1048

high-resolution multicolored LED panels combined as a single unit. For example, the message LED traffic panel 11 can be constructed from combining 12 (6 rows of 2) light emitting diode dot matrix modules known as the 0630IM manufactured by Desay Optotech Ltd. Of Huizhou, China. The message LED panel 11 is capable of displaying important information such as traffic conditions and emergency instructions. Additionally, the message LED panel 11 is capable of displaying full color high-resolution advertisements that may be animated or still.

[0048] Furthermore, the message LED panel 11, the pedestrian traffic panel 12, and the primary traffic panel 14 can all have the same resolution, as provided by the LED modules. Also, the locations on the body 10 of the message LED panel 11, the pedestrian traffic panel 12, and the primary traffic panel 14 are interchangeable. This can be done physically or by software. Alternatively, all three panels can be used as message boards.

[0049] As shown in FIG. 2, the primary LED traffic panel 14, the pedestrian LED traffic panel 12 and the message LED panel 11 are all controlled by an LED panel controller 24. Additionally, when the street sign 15 contains an LED panel or other lighted element, the street sign 15 is also controlled by the panel controller 24. The panel controller 24 controls each LED on the LED panels according to instructions provided by a signal processor 25. The signal processor 25 may be a computer with a microprocessor, memory, storage device, such as a hard disk, and an interface for sending instructions to the panel controller 24. The signal processor 25 may also include other hardware necessary for controlling the LED panels 11, 12, 14, 15 and the controller 24. The signal processor may be located within of the column 10 or at a remote location.

[0050] The signal processor 25 runs a program for controlling the primary LED traffic panel 14 according to the primary LED traffic panel scheme, controlling the pedestrian LED traffic panel 12 according to the pedestrian LED traffic panel scheme and controlling the message LED panel 11 according to a message program as described below.

[0051] The signal processor 25 controls the primary LED traffic panel 14 according to the primary LED traffic panel scheme (primary scheme). This primary scheme

APPLE EXHIBIT 1102, Page 561 of 1048

defines what images the primary LED traffic panel 14 displays to direct traffic and the logic used to determine when to display the various defined images.

[0052] FIGs. 3A – 3C illustrate several examples of primary schemes that can be used according to the current specification. The most basic primary scheme is illustrated in FIG. 3A, this is the conventional primary scheme for a traffic control light. According to this scheme, three circles are depicted representing the circles of a conventional traffic light. As with a conventional traffic light, when traffic is to be directed to proceed, the top circle 31 glows green. When traffic is to be informed of an impending red light, the center circle 32 glows yellow. When traffic is to be directed to stop, the bottom circle 33 glows red. Additionally, other traffic signals can be created, such as blinking yellow and blinking red signals, as desired. The conventional scheme includes the images to be displayed and the logic for directing traffic according to this scheme.

[0053] FIG. 3B illustrates an example of a modified conventional primary scheme. This scheme may be any scheme that preserves basic characteristics of the conventional scheme. For example, the conventional scheme may be modified to consist of three rectangles, as shown in FIG. 3B. These rectangles would act in the same way the three circles of the conventional primary scheme as illustrated in FIG. 3A and described above. For example, when traffic is to be directed to proceed, the top rectangle 34 glows green. When traffic is to be informed of an impending red light, the center rectangle 35 glows yellow. When traffic is to be directed to stop, the bottom rectangle 36 glows red.

[0054] Other modifications could include causing the entire primary LED panel to turn one solid color such as all red, all yellow, or all green to control traffic. Other modifications could incorporate the use of a count-down timer indicating how much longer the signal will remain the same until it is time to switch. For example, when the light is yellow, a timer in the form of black digital display 37 located in the center of the yellow light 35 can count down the time remaining until the light turns red. Another example would be the use of a stripe (not shown) either vertically or horizontally aligned that shrinks as the time until the next light change approaches. After the stripe has fully disappeared, the light changes. These count-down timers would give motorists greater opportunity to ascertain whether they should come to an

APPLE EXHIBIT 1102, Page 562 of 1048

immediate stop or continue through the intersection, thereby aiding the flow of traffic and increasing the safety of the intersection.

[0055] FIG. 3C illustrates an example of a gradient primary scheme. The gradient primary scheme calls for the illumination of only a horizontal strip 38. The strip begins at the top of the LED panel where the gradient is colored green. The strip then moves downward illuminating a color that is increasingly yellow and decreasingly green until the strip is half-way down the LED panel and fully yellow. At this point the strip continues to move downward illuminating a color that is increasingly red and decreasingly yellow. When the strip has reached the bottom of the LED panel, it is fully red. Then a clear signal is sent indicating that traffic is directed to stop. For example, the strip may disappear and the entire rectangle will turn red. The use of this timed gradient allows motorists greater opportunity to ascertain whether they should come to an immediate stop or continue through the intersection, thereby aiding the flow of traffic and increasing the safety of the intersection.

[0056] The pedestrian LED traffic panel 12 is controlled according to the pedestrian LED traffic panel scheme (pedestrian scheme). This pedestrian scheme defines what images the pedestrian LED traffic panel 12 displays to direct traffic and the logic used to determine when to display the various defined images. The conventional pedestrian scheme illustrates a white stick figure of a person walking indicating that pedestrian traffic may cross the street. A red stick figure of a person standing still indicates that pedestrian traffic may not cross the street. The animated pedestrian scheme uses figures similar to the conventional pedestrian scheme, however, the stick figures are animated to greater clarify the intent of the signal. For example, when pedestrian traffic is directed to proceed with crossing the street, a white stick figure person may have legs that move to indicate walking. Animation need not be limited to the walking signal, for example, when pedestrian traffic is directed to refrain from crossing the street, a red stick figure person might be shown to repeatedly tap one foot to illustrate waiting. Additionally, the timer features discussed above may be incorporated into the pedestrian scheme. For example, a count-down timer might accompany the walking stick figure.

[0057] A message program is used to control the message LED panel 11 according to the present invention. The message program runs on the signal processor 25, and the

APPLE EXHIBIT 1102, Page 563 of 1048

signal processor 25 is connected to a computer communications network such as the Internet or a wide area network, as shown in FIG. 2. The signal processor 25 may be connected to the computer communications network via a standard telephone line, a DSL line, a fiber-optic line, a coaxial cable or any other form of wired connection. Alternatively, the processor 25 may be connected to the computer communications network via a wireless connection, such as a wireless connection over a digital cellular telephone network or a wireless local area network connection, such as a wireless connection conforming to IEEE 802.1b or 802.11 (e.g. 802.11a, 802.11b, 802.11g, and 802.11n). The connection into the computer communications network described above (hereinafter "computer communications network connection") may be over the Internet using a secure method of communication such as encryption and/or a secure virtual private network (VPN). Using the computer communications network connection, the municipality or contracted administrator may modify the message program to update messages or to install animated advertisements. The message program determines what messages are displayed on the message LED panel 11. The message LED panel 11 is also capable of disseminating important information such as traffic conditions and emergency instructions. For example, text can be displayed indicating that poor traffic conditions are ahead and advising motorists of alternative routes or information to motorists to be on the lookout for a particular vehicle suspected of transporting a fugitive. Text can be displayed all at once or text may scroll across the panel. The message program is capable of receiving any manner of message from the computer communications network.

[0058] In order to increase municipal revenue, municipalities may choose to run advertisements on the message LED panel 11. Because the message LED panel 11 is high resolution, television-style advertisements can be displayed. These advertisements can be tailored for the particular community in which the traffic light is installed, thereby increasing the value of the advertisement. The advertisements can be regularly updated over the computer communications network connection.

[0059] FIG. 5 illustrates an embodiment of the invention for additional uses of the message board 11 in the LED traffic light. As shown in FIG. 5 and previously mentioned above in connection with FIG. 2, signal processor 25 is coupled to a computer communications network such as the Internet 51. Signal processor 25 may

APPLE EXHIBIT 1102, Page 564 of 1048

also be coupled to a cellular network 52 and function as a cellular phone transmitter or relay station for a cellular telephone 53 via cellular phone antenna 54. Signal processor 25 may also be adapted to serve as a wireless internet transmitter station or hot spot via access point 55. A portable laptop, personal digital assistant (PDA), or other personal portable electronic devices 56 can connect to the Internet 51 through access point 55. Access point 55 may support 802.11 wireless public networking, Bluetooth, or any other wireless or infrared technologies known in the art. Signal processor 25 can be adapted to include audio and or video surveillance equipment 57 to assist in law enforcement, emergency response or advertising. Thus, the LED traffic light can also be adapted to include a stationary or mobile webcam, live cam, or other digital audio and/or video equipment (represented by camera microphone 57) which records and/or broadcasts real-time audio and/or video of its surrounding location to other LED traffic lights nearby or in other cities, or on a website 58 via Internet 51.

[0060] Additionally, an emergency motif can be used to warn motorists and pedestrians to move to the curb or stay on the sidewalk in the event of an oncoming emergency vehicle.

[0061] The LED traffic light according to the present invention has the added advantage of being easily configurable to display a holiday or festive motif. To implement such a motif, the utilized schemes can be modified to display timely festive accents on one or more LED panels. The use of specialized motifs is not limited to holidays and festivals, motifs can be used to modify the appearance of LED traffic lights in accordance with the cultural or historic significance of the neighborhood or area in which the LED traffic light is located.

[0062] The LED traffic light according to the present invention is not limited to a vertical column configuration. For example, the LED traffic light may be horizontally oriented. FIG. 4 illustrates an embodiment of the present invention where the LED traffic light is horizontally oriented. According to one example of a horizontal orientation, the traffic light 40 is held over the intersection by cables or attached to one or more support columns. The primary LED traffic panel 42 directs traffic while the message LED panel 41 displays a message or advertisement. A pedestrian LED traffic panel (not shown) and/or a street sign (not shown) may also be included in

APPLE EXHIBIT 1102, Page 565 of 1048

horizontal embodiments. In another example, the LED traffic light may be oriented in a L-shaped, a hook-shaped, a chair-shaped, a bench-shaped, a steeple-shaped, or a zig-zag-shaped configuration by arranging the message LED panel 11, the pedestrian LED traffic panel 12, and the primary traffic panel 14 either 45° or 90° relative to each other.

[0063] In another embodiment of the present invention, the LED traffic light allows a user in visual and communications proximity of message LED panel (or LED message board) 11 to interact with the device to obtain information, products and/or other services. Multiple users may interact with the device simultaneously or sequentially, for example, on a first-come, first-served basis. As an example, when the device is being used by a particular user, messages sent by other users may be queued by signal processor 25. The device may respond only to messages sent by a current user, until an ending condition, such as a message transmitted by a user or the mere passage of time, allows the device to respond to the next message received by another user that is stored in the queue. As noted above and illustrated in FIG. 5, signal processor 25 is connected to Internet 51. Either access point 55, cellular antenna 54 or an interface to a cellular telephone network 52 or communications through Internet 51 can provide a proximal user to communicate with signal processor 25 and affect the output of LED message board 11. Proximal users of personal portable electronic devices, such as cellular phone 53, PDA/wireless laptop 56, handheld email browsers, MP3 players, digital cameras or other electronic devices with wireless communications capabilities (not shown) may interface with signal processor 25, either directly or via Internet 51 or cellular network 52, using emails, SMS text messages, or other types or formats of messages as well known in the art. To communicate with the LED traffic light, the proximal user may need to register on a website and/or download a program into the personal portable electronic device, or authenticate via web server 58 as is well known in the art. In a preferred embodiment, signal processor 25 or web server 58 provides two-way communications with the personal portable device. In this way, LED message board 11 can be used as, for example, an information terminal. As an information terminal, using his/her personal portable electronic device (e.g., PDA 56), a proximal user may request information, such as directions to a particular event or address, advertisements, promotions, play games, or request other visual output from LED message board 11 . Alternatively,

APPLE EXHIBIT 1102, Page 566 of 1048

signal processor 25 may receive requests from a proximal user and send back requested information directly to his/her PDA 56 or other personal portable electronic device while providing instructions to obtain information on LED message board 11. Such information may be stored in signal processor 25 or may be obtained from web server 58 or Internet 51.

[0064] As noted above, the LED traffic light may also provide a cellular phone transmitter or relay station or a wireless internet transmitter, thus allowing a user with a wireless laptop computer or a web-enabled cellular phone to directly access the Internet, if desired.

[0065] In addition, the LED traffic light may be used as a self-service electronic point-of-sales station. For example, the local user may use a Bluetooth®-enabled personal portable electronic device, such as a cellular phone, to access the LED traffic light to purchase tickets for a theater show advertised on the LED message board 11.

[0066] As discussed above and illustrated in FIG. 5, a municipality or administrator may modify the message program or install animated advertisements using a computer 59 connected to the Internet 51, for example. In addition, individuals or businesses that advertise using the LED traffic light may also modify or update their advertisements via Internet 51 or by interacting directly with LED traffic light using a personal portable electronic device (e.g., PDA 56; cell phone 53), or through some alternative computer communications network or cellular telephone network 52. This feature may be particularly useful for local advertisers who will be able to modify or update their advertisements based on local pedestrian traffic, for example, or based on pedestrian's digital interaction with the LED traffic light, as instructed by its LED message board 11. Furthermore, LED traffic lights may be designated as digital bulletin boards, enabling users to post information or "digital flyers" regarding upcoming local community events.

[0067] FIG. 6 is a flow chart that illustrates a method for interacting with a messaging board. As shown in FIG. 6, at step 610, the system receives a wireless message from a PDA. Alternatively, the system may receive the message from the Internet. As explained above, the message may be an email, text message, or some other form of digital communication as is well known in the art.

APPLE EXHIBIT 1102, Page 567 of 1048

[0068] In a preferred embodiment, illustrated in step 620, the system authenticates the identity of the PDA user using the identification information contained in the message. In an alternate embodiment, the system will query the user for further information in order to verify that the user has the proper credentials to interact with the system. This step is optional, as the system is designed for unfettered public use. However, different users may be permitted to perform different functions, depending upon their authenticated identity, thus authentication may ensure that unauthorized tampering occurs. If authentication is required, unauthenticated user messages are discarded.

[0069] In step 630, the system examines the message sent to it to determine if the PDA sending the message is at the front of the line for interaction with the message board. If not, in step 635 the user's message is stored in a queue for later processing. However, if the user is the current user interacting with the system, processing proceeds to step 640 where the system updates the visual indicia on LED message board 11 in response to the message. Exemplary updates were illustrated in the preceding paragraphs.

[0070] Optionally, in step 650, the system generates a message responsive to the user's message. The responsive message might contain data from a peripheral device attached to the system, for example, digital audio or video, and send that information back to a web server for viewing through the Internet. The system may alternatively send the information directly to the user's PDA. In this fashion, point-of-sale transactions could be conducted.

[0071] In step 660, the system checks to see if an ending condition has been reached. This check is necessary to ensure that a particular user does not monopolize the resources provided by the messaging board system. Such condition comprises the passage of time or a particular limit on the number of user messages processed, for example. If the end condition has been reached, in step 670 the identity of the current user is updated to the next user waiting in line.

[0072] In step 680, the next message in the queue is retrieved. Processing loops back to step 630, where that message is checked to see if it came from the current user.

[0073] Although the present invention has been described in relation to particular

APPLE EXHIBIT 1102, Page 568 of 1048

embodiments thereof, many other variations and modifications and other uses will become apparent to those skilled in the art. It is preferred, therefore, that the present invention be limited not by the specific disclosure herein, but only by the appended claims.

APPLE EXHIBIT 1102, Page 569 of 1048

WHAT IS CLAIMED IS:

1. An interactive bulletin board system, comprising:

a signal processor coupled to a computer communications network;

a messaging board coupled to said signal processor, said messaging board providing visual indicia responsive to said signal processor; and

a personal portable electronic device wirelessly coupled to said computer communications network;

wherein said signal processor updates said visual indicia displayed on said messaging board in response to a message sent from said personal portable electronic device to said signal processor.

2. The system of claim 1, wherein said messaging board comprises a matrix of multicolored LEDs.

3. The system of claim 2, further comprising a traffic panel including a first matrix of multicolored LEDS for directing motor traffic by displaying a plurality of images.

4. The system of claim 2, wherein said visual indicia represents information requested through the personal portable electronic device.

5. The system of claim 4, wherein said information is obtained from the Internet.

6. The system of claim 5, wherein said signal processor authenticates said personal portable electronic device before updating said visual indicia.

7. The system of claim 6, wherein said personal portable electronic device comprises a device selected from the group comprising a cellular phone, a wireless laptop, a handheld email browser, an MP3 player and a digital camera.

8. The system of claim 7, further comprising a camera coupled to said signal processor, wherein said visual indicia comprises images provided by said camera.

9. The system of claim 8, wherein said signal processor responds to messages sent from more than one personal portable electronic device.

APPLE EXHIBIT 1102, Page 570 of 1048

10. The system of claim 9, wherein said signal processor responds sequentially to messages received from more than one personal portable electronic device.

11. The system of claim 10, wherein said signal processor responds to messages received from only a single personal portable electronic device until an ending condition is met.

12. A method of displaying visual indicia on a public display, the method comprising:

a processor receiving a message via a computer communications network from a personal portable electronic device wirelessly coupled to said computer communications network; and

the processor updating visual indicia displayed on said display in response to said message.

13. The method of claim 12, further comprising authenticating said personal portable electronic device.

14. The method of claim 13, further comprising the processor queuing messages received from said computer communications network.

15. The method of claim 14, further comprising sending information to said personal portable electronic device in response to said message.

16. The method of claim 15, wherein said information is one or more selected from the group comprising a digital image, a series of digitized images and digitized audio.

17. The method of claim 16, wherein said message is one or more selected from the group comprising a text message and an email.

18. The method of claim 14, further comprising sending information to a web server.

19. The method of claim 18, wherein said information is one or more selected from the group comprising a digital image, a series of digitized images and digitized audio.

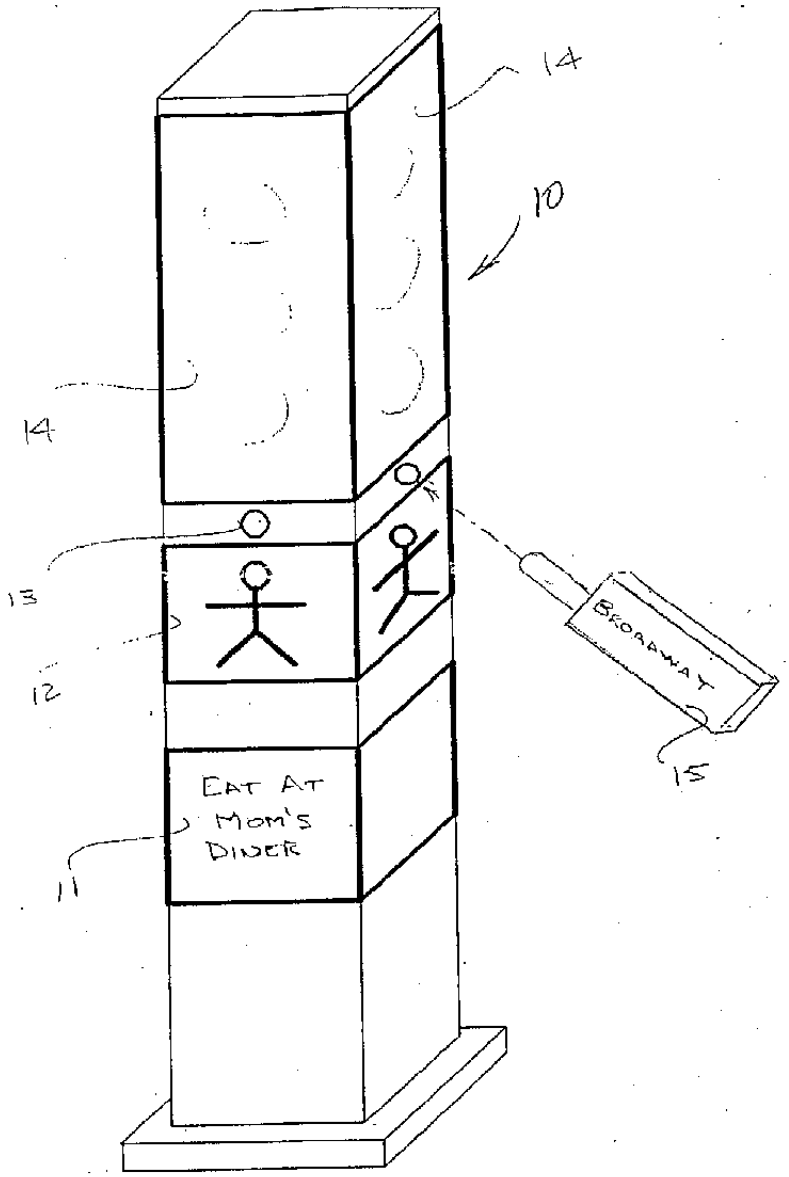
20. The method of claim 14, further comprising the processor updating visual indicia in response to one or more messages received only from a single personal portable electronic device.

APPLE EXHIBIT 1102, Page 571 of 1048

21. The method of claim 20, further comprising the processor updating said visual indicia in response to a queued message sent from a second personal portable electronic device after an ending condition is met.

APPLE EXHIBIT 1102, Page 572 of 1048

FIG. 1



APPLE EXHIBIT 1102, Page 573 of 1048

FIG. 2

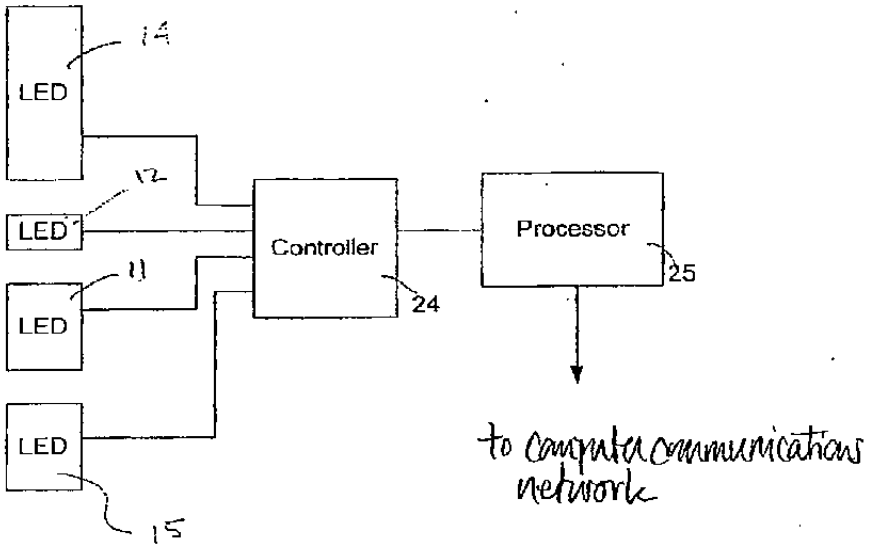

LED
14
LED
12
LED
11
LED
15
Controller
24
Processor
25
to computer communications network

FIG. 3A

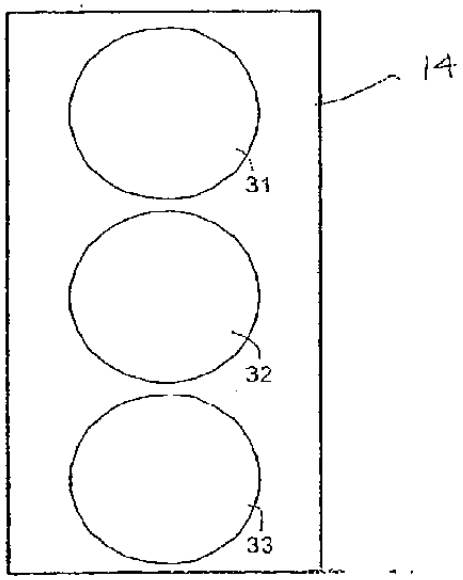


FIG. 3B

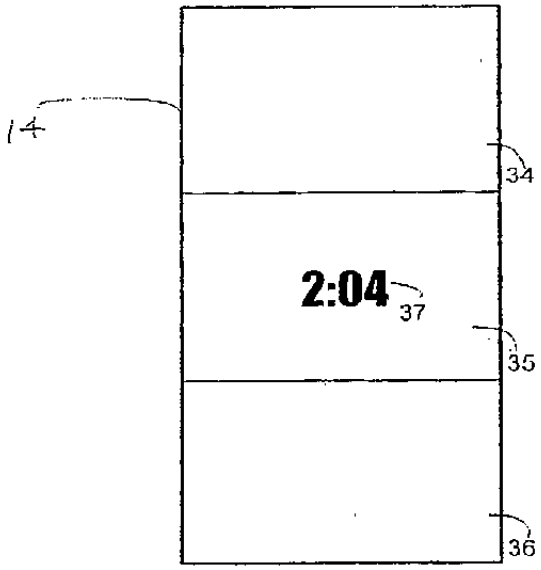


FIG. 3C

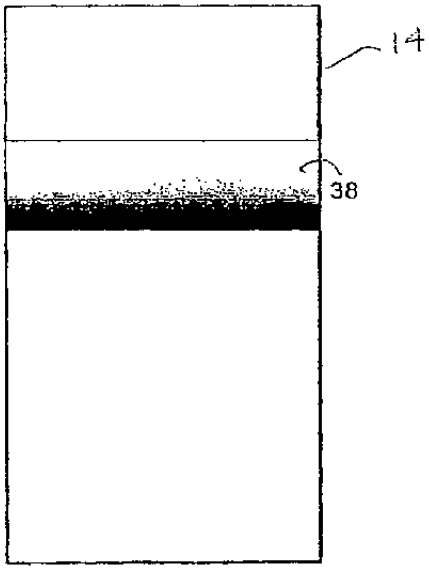
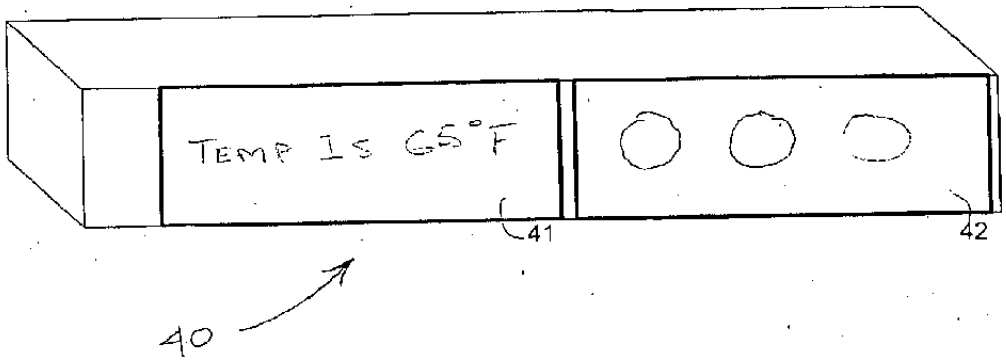


FIG. 4



APPLE EXHIBIT 1102, Page 576 of 1048

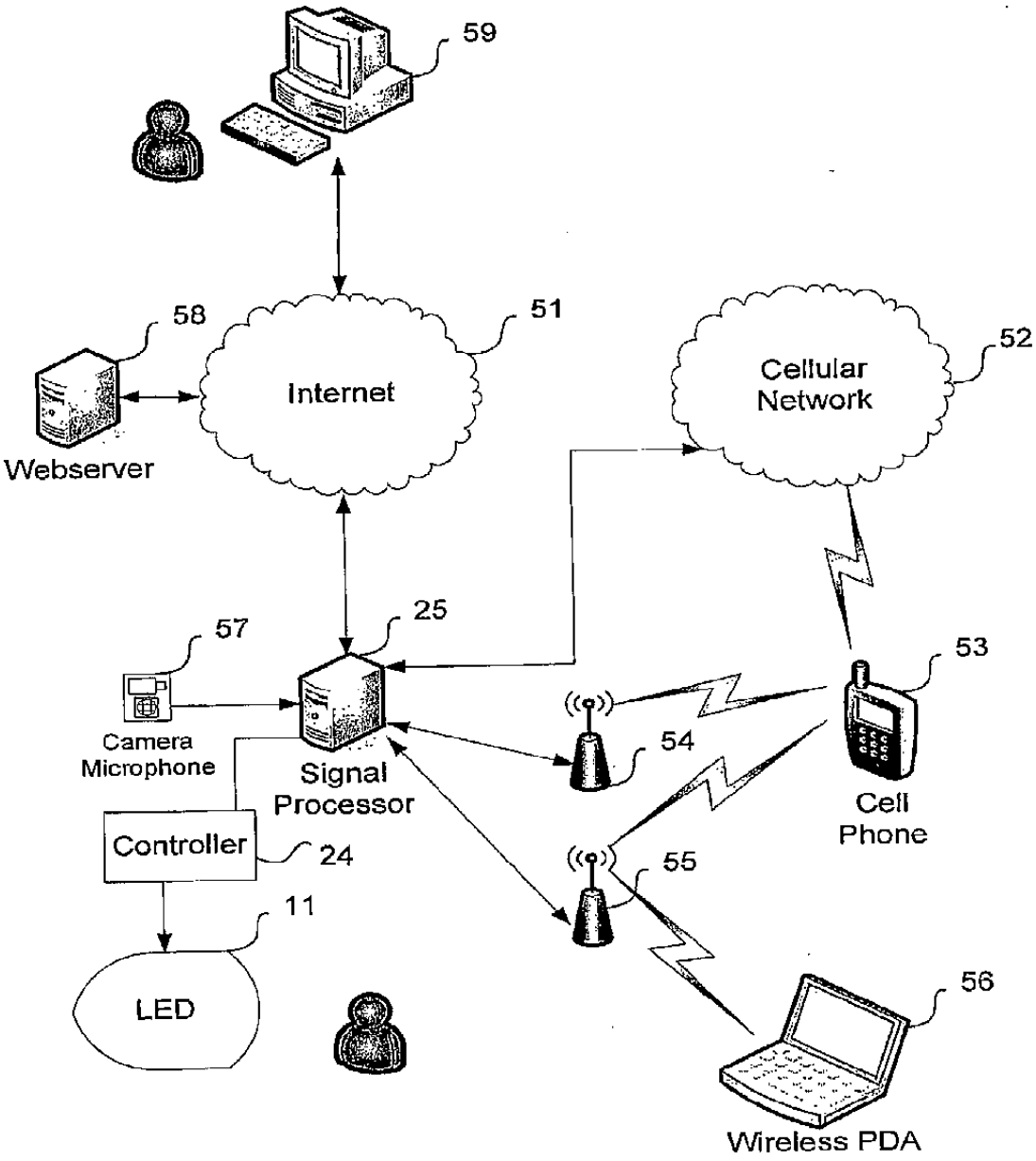


Fig. 5

APPLE EXHIBIT 1102, Page 577 of 1048

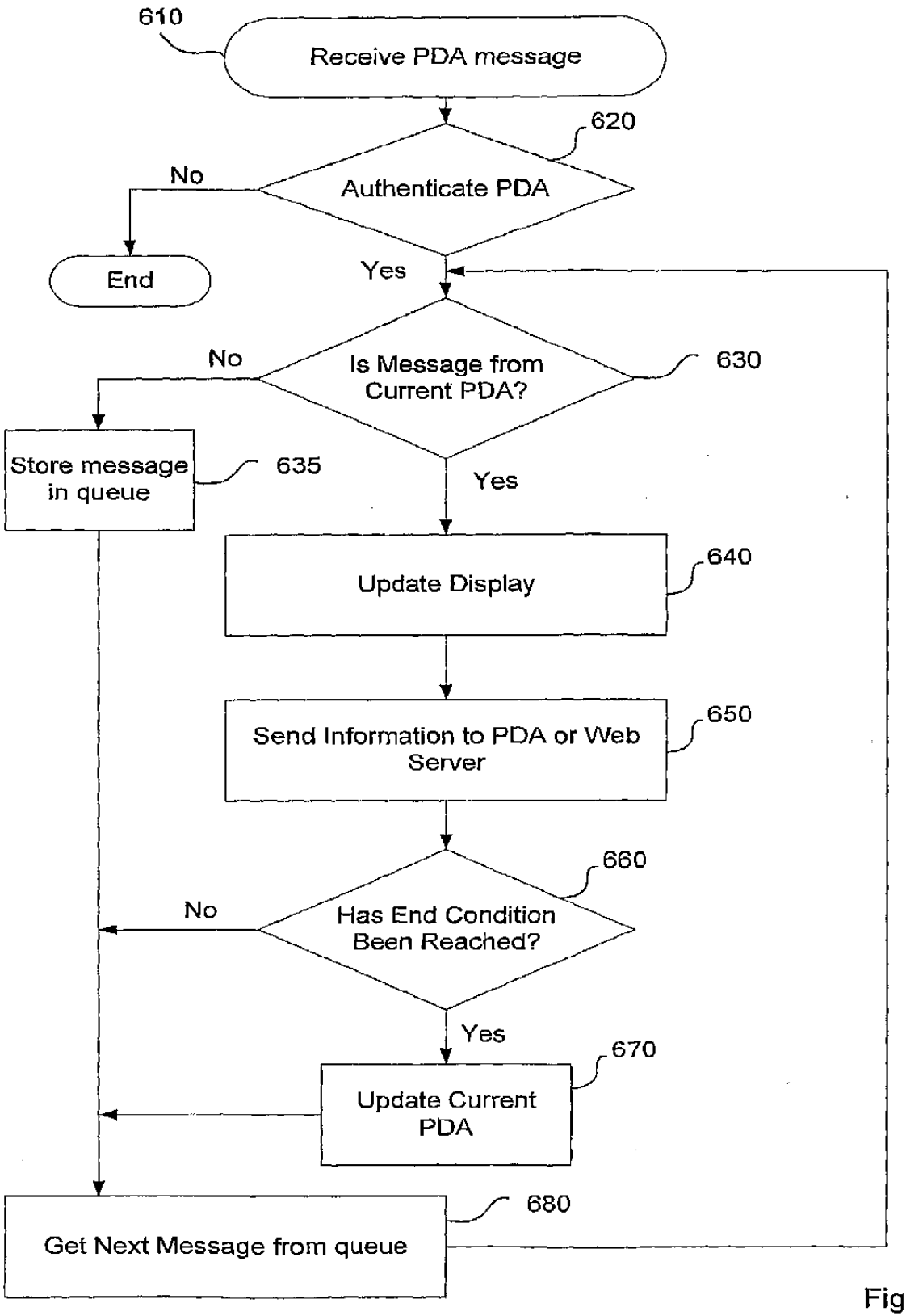


Fig. 6

APPLE EXHIBIT 1102, Page 578 of 1048

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 January 2008 (31.01.2008)

PCT

(10) International Publication Number
WO 2008/013504 A1

(51) International Patent Classification:
H04L 12/14 (2006.01) *G06Q 30/00* (2006.01)
H04L 12/16 (2006.01) *H04L 12/22* (2006.01)
H04L 29/06 (2006.01) *H04L 29/08* (2006.01)

(21) International Application Number:
PCT/SG2006/000212

(22) International Filing Date: 26 July 2006 (26.07.2006)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **STARHUB LTD** [SG/SG]; 51 Cuppage Road, #07-00, Starhub Centre, Singapore 229469 (SG).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **EE, Thomas Chong Gay** [SG/SG]; 511 Woodlands Drive 14, #05-61, Singapore 730511 (SG).

(74) Agent: **DREW & NAPIER LLC**; 20 Raffles Place, #17-00 Ocean Towers, Singapore 048620 (SG).

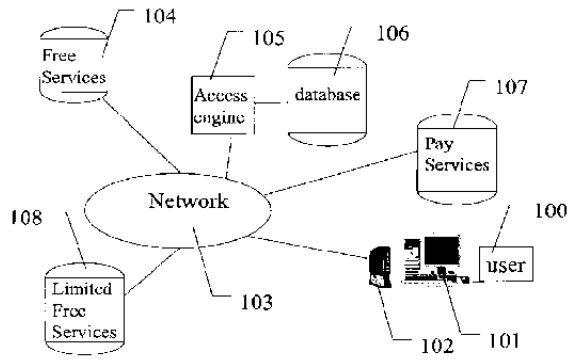
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: NETWORK ACCESS METHOD



(57) Abstract: A network access system and method are described the system comprising the steps of: identifying the hardware identifier of the connecting device; comparing the hardware identifier of the network device to previously stored hardware identifiers by providing some free network services for the user to have access to. The network access system then identifies a class to which a known previously stored hardware identifier belongs. The classes comprising: a first class where the hardware identifier has been previously stored and the user of the connecting device has provided details including payment details. A second class where the hardware identifier has been previously stored but the user of the device has not provided details. The network access system then permits access to the first class of hardware identifiers to a first class of network services (and/or web sites) and access to the second class of hardware identifiers to a second class of network services and web sites. The second class of service being a sub set of the first class of services. The system stores unknown hardware identifiers and permits access to unknown hardware identifiers to the second class of services.

WO 2008/013504 A1

NETWORK ACCESS METHOD

TECHNICAL FIELD

The present invention relates to a method of providing network access to unknown users. In particular the present invention relates to a network access method allowing network access to unknown users using a system hardware identifier to distinguish users.

BACKGROUND ART

Network providers such as internet service providers ("ISP"), hotels or cafés in order to operate the network need to collect usage charges. In order to do so, the network providers must be able to identify each user connecting to the network. User identification is commonly achieved via a process of user registration. An example of user registration is the use of a web page in which a new user enters their details and a unique user account is created for the user. The user account will allow the user to log onto the network to gain access. Another example of the registration of a user is in an internet café or hotel where pass codes are given out allowing temporary connect to the network.

The use of the identifier allows the network provider to monitor resources consumed by the individual user and to charge for those resources either on a time basis or on a volume basis.

Such a system makes it difficult for users in a new network environment, either wireless or wired, to connect to a previously unknown system. It also makes it difficult and expensive for network service providers to collect the necessary information in order to be able to bill the user.

SUMMARY OF THE INVENTION

Accordingly, it is an object of the present invention to overcome the above disadvantages or to provide the public or industry with a useful choice.

In a first embodiment the present invention consists in a network access method comprising the steps of:

detecting the hardware identifier of the connecting network device;

comparing said hardware identifier to previously stored hardware identifiers; and

identifying a class to which a known previously stored hardware identifier belongs, said classes comprising:

a. a first class where said hardware identifier has been previously stored and the user of said connecting device has provided details including payment details; or
b. a second class where said hardware identifier has been previously stored but the user of said device has not provided details;

permitting access to said first class of hardware identifiers to a first class of network services;

permitting access to said second class of hardware identifiers to a second class of network services, said second class of service being a sub-set of said first class of services; and

storing unknown hardware identifiers and permitting access to said second class of services.

Preferably said hardware identifier is a hardware address of a network device.

Alternatively said hardware identifier is a hardware address of a wired network card.

Alternatively said hardware identifier is a hardware address of a wireless network card.

Preferably wherein each said class of network services includes network services selected from a plurality of web pages, access to Internet sites and/or access to other network resources, including network file and print services.

Preferably including the step of prompting users outside of class one to enter user and or payment details when said user attempts to access web sites outside of said user's class of service.

Preferably said user of said connecting device has provided details including payment details, said method including recording network usage by said user.

Preferably said step of storing unknown hardware identifiers includes the step of recording the time of first access;

said classes further including:

c. a third class where said hardware identifier has been previously stored but the user of said device has not provided details and a defined period of time since said first access has expired;

said step of identifying said class to which an identifier belongs includes the step of identifying hardware identifiers in said second class where the defined period of time since said first access has expired, and moving said hardware identifiers into a third class;

and said steps include permitting access to said third class of hardware identifiers to a third class of network services, said third class of service being a sub-set of said second class of services.

Preferably said user of said connecting device has provided details including payment details, said method including recording network usage by said user, accounting for said network usage by said user, and wherein when said users balance exceeds a limit set by said network provider moving said user into said third class.

Preferably including the step of prompting said user for a payment before said users balance exceeds said limit.

Preferably including the step of prompting said user for a payment when said users balance exceeds said limit.

Preferably said limit is zero and said payment is a prepayment.

Preferably including the step of attempting deducting an amount from an account provided by said user before moving said user into said third class.

Preferably said classes further including:

d. a fourth class where said hardware identifier has been denied network access by the network service provider;

and said steps include prohibiting network access to said fourth class of hardware identifiers.

Preferably network access to said second class of network services is granted until said hardware identifier is detected.

Preferably network access is charged by network resource consumed.

Preferably network access is charged by data volume.

Alternatively network access is charged by data volume for accessing class one network services.

Alternatively network access is charged by one or a combination of data volume, duration of access or access speed.

In a second embodiment the present invention consists in a method of providing prepaid access to network services comprising the steps of:

allowing access to a sub-set of all network services while the hardware identifier of a connecting network device is detected;

comparing the detected hardware identifier to previously detected and stored hardware identifiers;

storing unknown hardware identifiers and associating said hardware identifiers with an account;

requesting a payment if the balance of said account is below a set amount;

incrementing said account balance when a payment is received;

decrementing said account balance as network services are consumed; and

allowing a network device access to a set of network resources, said account balance determining the set of network services a detected hardware identifier is allowed access to.

Preferably said hardware identifier is a hardware address of a network device.

Alternatively said hardware identifier is a hardware address of a wired network card.

Alternatively said hardware identifier is a hardware address of a wireless network card.

Preferably a network device with an account balance above said set amount is allowed access to the full set of network services.

Preferably said set amount is zero.

Preferably including the step of requesting a payment before said balance is at or below said limit.

Preferably the set of network services includes a plurality of web pages, access to Internet sites and or access to other network resources including network file and print services.

Preferably including the step of requesting a payment when a network device requests network services outside of the set which the device is allowed access to.

Preferably the step of storing unknown hardware identifiers and associating said hardware identifiers with an account includes setting a grace period for free access, and wherein said grace period status and said account balance determine the set of network services a detected hardware identifier is allowed access to.

Preferably said account balance is decremented based on a charge for data volume.

Alternatively said account balance is decremented based on a charge for access to a sub-set of network services.

Alternatively said account balance is decremented based on a charge for a combination of data volume, duration of access or access speed.

Preferably including the step of denying network access to hardware identifiers tagged as prohibited.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described with reference to the accompanying drawings in which:

Figure 1 is a block diagram of the present invention;

Figure 2 is an example illustrating the unique hardware identifier of some network devices;

Figure 3 is a flow diagram of the system of the present invention; and

Figure 4 is an example payment interface of the present invention.

DETAILED DESCRIPTION

Referring to Figure 1 when a new user using a computer 101 or computer like device seeks to establish a connection with a network operator 103 in order to enjoy free network services 104, the underlying network allows the network operator to detect the hardware identifier (hardwareID) of the network device 102 the user is using to connect to the network.

This detection, a process of discovery and collection, of network device hardwareID 102 is part of the functionality within the access engine 105. The hardware identifier is a globally unique hardware address of the network device 102, commonly known as the medium access control (MAC) address. The network device can be a modem (examples, cable modem with a MAC address) or an Ethernet network interface card (NIC) commonly found in personal computers. An example of the Ethernet card with its MAC address identifier 201 can be seen in Figure 2.

The access engine 105 of the present invention is a non-intrusive system that is transparent to the user as it sniffs and process the network information in the background, while the user enjoys the free services 104. When the access engine 105 has identified the hardware identifier 102, it will check in its database 106 to determine whether this hardware identifier 102 has been previously detected.

The hardware identifiers 102 within the database 106 are sorted into four classes. The first class contains those identifiers that have been detected and the user has provided their details including payment details and the users account balance does not exceed a limit set by the network provider. In one embodiment the limit set is zero and the user has to prepay for the network access. The second class contains those identifiers that have been detected but the user has not yet provided their details. The third class contains those identifiers that have been detected; the user has not provided their details and a time period fixed by the network provider for providing details from when access first granted has expired. The third class may also contain hardware identifiers of users who have provided details but where the users account debt balance exceeds the limit set by the network provider. A fourth class of hardware identifiers that are banned from connection to the system may also be kept. Users may be banned from access for various reasons.

Each class of user has a class or sub-set of network services that they can access. In the preferred embodiment the first class 107 contains more services than the second class 104 and the second class contains more services than the third class 108. A user in the first class is allowed access to all the services in the first 107, second 104 and third class 108 of services. A user in the second class is allowed access to the second 104 and third class 108 of services and a user in the third class is allowed access only to the third class 108 of services. Each class of services may

contain a plurality of web pages, Internet access and other network services. Network services include file and print services.

Referring to Figure 3, free network access 301 to the second class of services is granted until the access system 105 detects the hardware identifier 302. If the hardware identifier 201 has not been previously detected 302 the access engine 105 initialises the user/device 304 by adding the network identifier to the database 106 and the access engine 105 may set a grace period for extended access.

During the grace period a user/network device is allowed to connect to the network 103 and access network services as allowed by the network service provider. Typically the user is provided with access to network access class two services, being a sub-set of the network services/resources available. After the grace period has expired the user is allowed access to only a limited number of network services, typically only to class three 108 services.

During the grace period the user may be allowed to access all or the vast majority of network services. After the grace period ends a user may only have access to a limited number of network services. These accessible network services can be any form of network resources in general, examples such as network file storage, network printing, email or access to Internet websites as selected by the network provider. The network provider my select services depending upon the services from which the network provider receives revenue.

If a user tries to access a network service or site that is part of the payable services 107, the user will be requested to provide user details to the network provider. The user may also be requested for their details after the initial grace period has expired if the user tries to access a network service (and/or web sites) that are allowed during the grace period but charged for after the grace period has expired. The user details may be requested via an online form, via a software application that is provided by the network provider that the user downloads (or is prompted to download to their system), or via a webpage.

In another embodiment the user may only need to make a payment and need not provide user details.

Referring to Figure 4 an example payment interface is shown. The payment interface may include the hardware identifier 401, the current limited associated with the hardware identifier account 402 and the current balance 403. The account status/balance may be shown on a separate

form from the payment interface, as seen in Figure 4 both the current balance and the payment interface are shown on the same form.

To make a payment a user enters the payment amount 404, the payment method 405, then enters details associated with the payment method 406. In the example illustrated credit card details are entered. The user can then choose to proceed 407 with the transaction or may alternatively cancel the transaction 408. The illustrated payment method is not to be understood as limiting the invention as many known online payment methods can be utilised without departing from the spirit of the invention.

Various network protocols for example, voice over IP, virtual private networks or instant messaging may be outside the pool of free services 104 and a user/device attempting to connect to any of these protocols would be prevented from doing so. During the grace period the user may be allowed access to some of these services.

Once the network operator has collected information from the user and stored against the user's hardware identifier within database 106, the user will have unlimited access provided that they have either prepaid for the access or provided that their account is within the limit set by the network provider.

Referring again to Figure 3 if the hardware identifier is known 303 the system checks for user and or payment details and an account balance 306. If the balance is acceptable to the network provider the system 105 grants open access 307 to the class one network services 107. The access system 103 records the user's access to or usage of at least chargeable network services and record the usage against the hardware identifier.

If the balance is unacceptable to the network provider the system 103 then checks whether the grace period has expired 308 if the period has not expired the user is provided with access to class two services and is prompted to provide details or make a payment 310.

If the grace period has expired the user is restricted to the third class of services and is prompted to provide details and or make a payment 311. This class of services is usually more limited.

In one embodiment a user who has previously made a payment is granted a grace period when their payment has been used. The system 103 may set a new grace period when the user's funds are used up.

The user may also provide bank account or credit card details and request that the access engine 105 automatically top the payment up once the user has used their prepaid amount.

In the preferred embodiment the users is charged by the network resources that the user accesses and not by the time the user is accessing services. The user may however be charged for any or a combination of access speed, access time or network resources consumed. The most preferred option is to charge for network resources consumed given that the user need not take a positive action to connect to the network.

Payments made by a user increment the users account balance and usage charges decrement the users balance. If the system is operated as a prepayment system balances below a defined negative limit prevent a user accessing the full array of network resources. In most cases the limit will be zero.

When access is from behind a router or any other device that substitutes its own hardware address then the system will identify only the last substituted hardware address. Accordingly that hardware address will be associated with the first user to provide details. Subsequent users will be able to access the same services as the first user is allocated and will be able to consume the first user's resources.

The foregoing description of the preferred embodiment of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise forms disclosed, since many modifications or variations thereof are possible in the light of the above teaching. All such modifications and variations are within the scope of the invention. The embodiments described herein were chosen and described in order best to explain the principles of the invention and its practical application, thereby to enable others skilled in the art to utilise the invention in various embodiments and with various modifications as are suited to the particular use contemplated thereof. It is intended that the scope of the invention be defined by the claims appended hereto, when interpreted in accordance with the full breadth to which they are legally and equitably suited.

CLAIMS

1. A network access method comprising the steps of:

detecting the hardware identifier of the connecting network device;

comparing said hardware identifier to previously stored hardware identifiers; and

identifying a class to which a known previously stored hardware identifier belongs, said classes comprising:

a. a first class where said hardware identifier has been previously stored and the user of said connecting device has provided user details including payment details; or
b. a second class where said hardware identifier has been previously stored but the user of said device has not provided details;

permitting access to said first class of hardware identifiers to a first class of network services;

permitting access to said second class of hardware identifiers to a second class of network services, said second class of service being a sub-set of said first class of services; and

storing unknown hardware identifiers and permitting access to said second class of services.

2. A network access method as claimed in claim 1 wherein said hardware identifier is a hardware address of a network device.

3. A network access method as claimed in claim 1 wherein said hardware identifier is a hardware address of a wired network card.

4. A network access method as claimed in claim 1 wherein said hardware identifier is a hardware address of a wireless network card.

5. A network access method as claimed in any one of claims 1 to 4 wherein each said class of network services includes network services selected a plurality of web pages, access to Internet site and/or access to other network resources, including network file and print services.

6. A network access method as claimed in any one of claims 1 to 5 including the step of prompting users outside of class one to enter user and or payment details when said user attempts to access web sites outside of said users class of service.

7. A network access method as claimed in any one of claims 1 to 6 wherein said user of said connecting device has provided details including payment details, said method including recording network usage by said user.

8. A network access method as claimed in any one of claims 1 to 7 wherein

said step of storing unknown hardware identifiers includes the step of recording the time of first access;

said classes further including:

c. a third class where said hardware identifier has been previously stored but the user of said device has not provided details and a defined period of time since said first access has expired;

said step of identifying said class to which an identifier belongs includes the step of identifying hardware identifiers in said second class where the defined period of time since said first access has expired, and moving said hardware identifiers into a third class;

and said steps include permitting access to said third class of hardware identifiers to a third class of network services and web sites, said third class of service being a sub-set of said second class of services.

9. A network access method as claimed in claim 8 wherein said user of said connecting device has provided details including payment details, said method including recording network usage by said user, accounting for said network usage by said user, and wherein when said users balance exceeds a limit set by said network provider moving said user into said third class.

10. A network access method as claimed in claim 9 including the step of prompting said user for a payment before said users balance exceeds said limit.

11. A network access method as claimed in claim 10 or claim 11 including the step of prompting said user for a payment when said users balance exceeds said limit.

12. A network access method as claimed in any one of claims 9 to 11 wherein said limit is zero and said payment is a prepayment.

13. A network access method as claimed in claim 9 including the step of attempting deducting an amount from an account provided by said user before moving said user into said third class.

14. A network access method as claimed in any one of claims 1 to 13 wherein

said classes further including:

d. an access denied class where said hardware identifier has been denied network access by the network service provider;

and said steps include prohibiting network access to said access denied class of hardware identifiers.

15. A network access method as claimed in any one of claims 1 to 14 wherein network access to said second class of network services is granted until said hardware identifier is detected.

16. A network access method as claimed in any one of claims 1 to 15 wherein network access is charged by network resource consumed.

17. A network access method as claimed in any one of claims 1 to 15 network access is charged by data volume.

18. A network access method as claimed in any one of claims 1 to 15 network access is charged by data volume for accessing class one network services.

19. A network access method as claimed in any one of claims 1 to 15 wherein network access is charged by one or a combination of data volume, duration of access or access speed.

20. A method of providing prepaid access to network services comprising the steps of:

allowing access to a sub-set of all network services while the hardware address of a connecting network device is detected;

comparing the detected hardware identifier to previously detected and stored hardware identifiers;

storing unknown hardware identifiers and associating said hardware identifiers with an account;

requesting a payment if the balance of said account is below a set amount;

incrementing said account balance when a payment is received;

decrementing said account balance as network services are consumed; and

allowing a network device access to a set of network resources, said account balance determining the set of network services a detected hardware identifier is allowed access to.

21. A method of providing prepaid access to network services as claimed in claim 20 wherein said hardware identifier is a hardware address of a network device.

22. A method of providing prepaid access to network services as claimed in claim 20 wherein said hardware identifier is a hardware address of a wired network card.

23. A method of providing prepaid access to network services as claimed in claim 20 wherein said hardware identifier is a hardware address of a wireless network card.

24. A method of providing prepaid access to network services as claimed in any one of claims 20 to 23 wherein a network device with an account balance above said set amount is allowed access to the full set of network services.

25. A method of providing prepaid access to network services as claimed in any claim 24 wherein said set amount is zero.

26. A method of providing prepaid access to network services as claimed in claim 24 or claim 25 including the step of requesting a payment before said balance is at or below said limit.

27. A method of providing prepaid access to network services as claimed in any one of claims 20 to 26 wherein the set of network services includes a plurality of web pages, access to Internet sites and or access to other network resources including network file and print services.

28. A method of providing prepaid access to network services as claimed in any one of claims 20 to 27 including the step of requesting a payment when a network device requests network services outside of the set which the device is allowed access to.

29. A method of providing prepaid access to network services as claimed in any one of claims 20 to 28 wherein the step of storing unknown hardware identifiers and associating said hardware identifiers with an account includes setting a grace period for free access, and wherein said grace period status and said account balance determine the set of network services a detected hardware identifier is allowed access to.

30. A method of providing prepaid access to network services as claimed in any one of claims 20 to 29 wherein said account balance is decremented based on a charge for data volume.

31. A method of providing prepaid access to network services as claimed in any one of claims 20 to 29 wherein said account balance is decremented based on a charge for access to a sub-set of network services.

32. A method of providing prepaid access to network services as claimed in any one of claims 20 to 29 wherein said account balance is decremented based on a charge for a combination of data volume, duration of access or access speed.

33. A method of providing prepaid access to network services as claimed in any one of claims 20 to 32 including the step of denying network access to hardware identifiers tagged as prohibited.

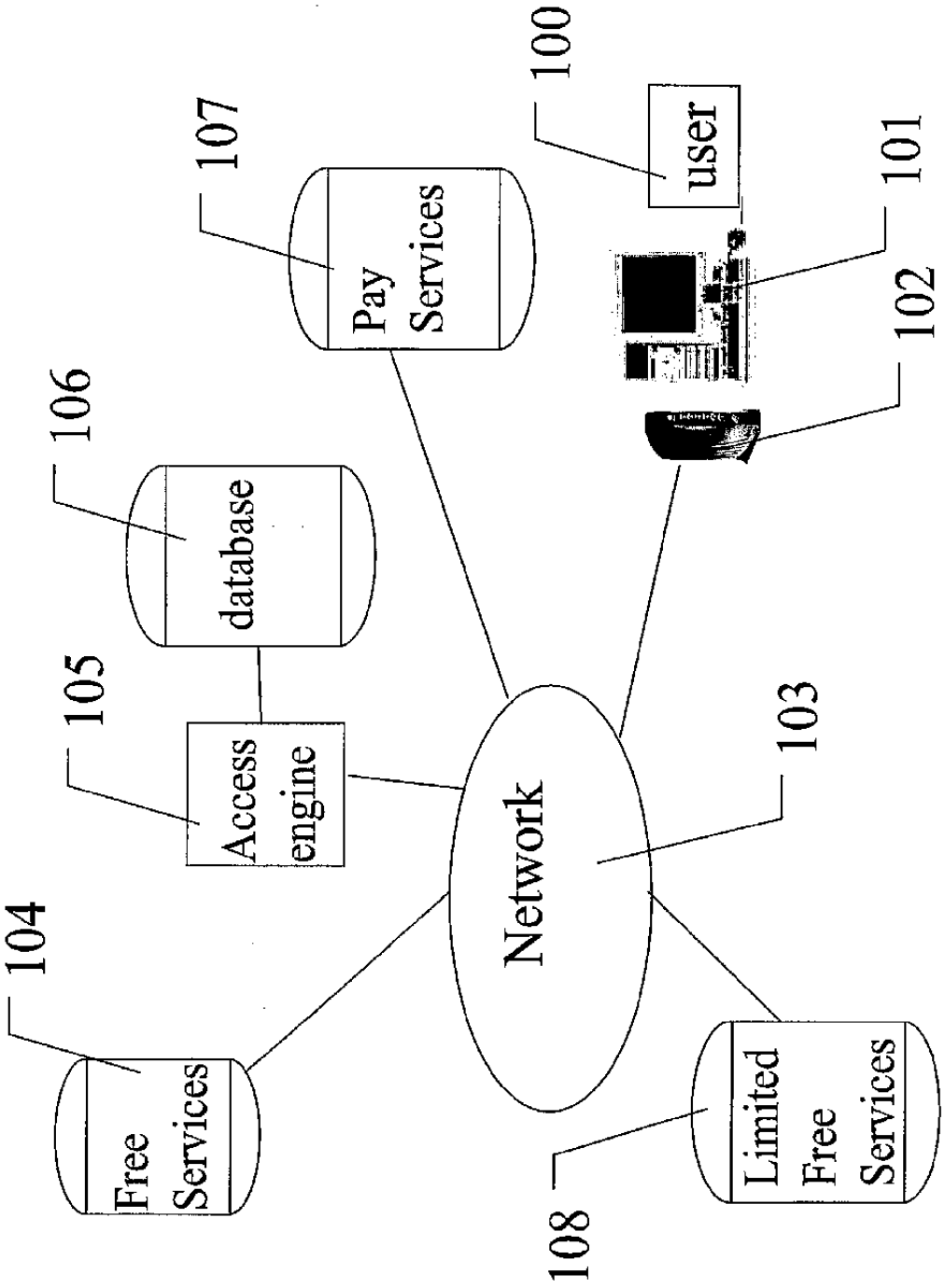
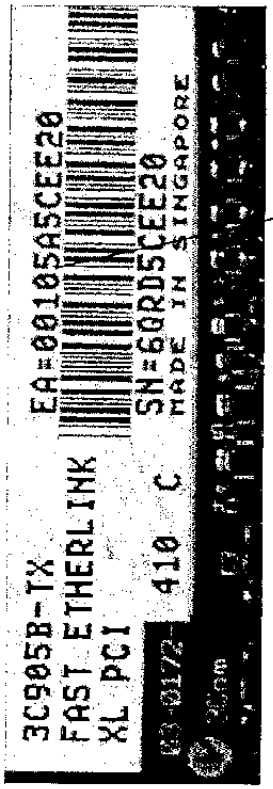

100
user
101
102
107
Pay Services
106
database
105
Access engine
103
Network
104
Free Services
108
Limited Free Services

Figure 1


3C905B-TX
FAST ETHERLINK
XL PCI
EA=0010A5CEE20
SN=60RD5CEE20
MADE IN SINGAPORE
03-0172 410 C

201

Figure 2

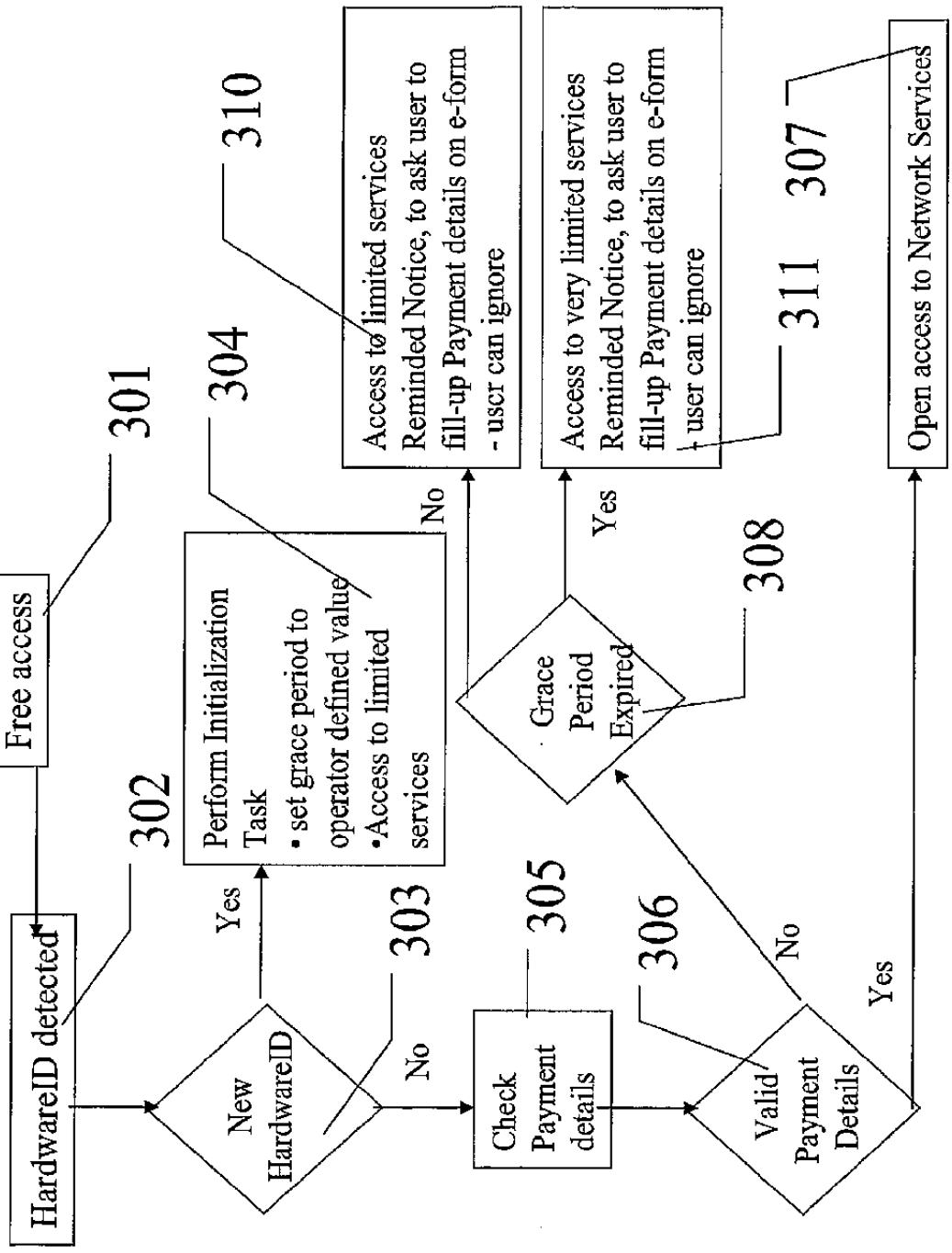

Free access
301
HardwareID detected
302
New HardwareID
303
Yes
No
Perform Initialization Task
• set grace period to operator defined value
•Access to limited services
304
Access to limited services
Reminded Notice, to ask user to fill-up Payment details on e-form
- user can ignore
310
Check Payment details
305
Valid Payment Details
306
No
Yes
Grace Period Expired
308
Yes
No
Access to very limited services
Reminded Notice, to ask user to fill-up Payment details on e-form
- user can ignore
311
Open access to Network Services
307

Figure 3

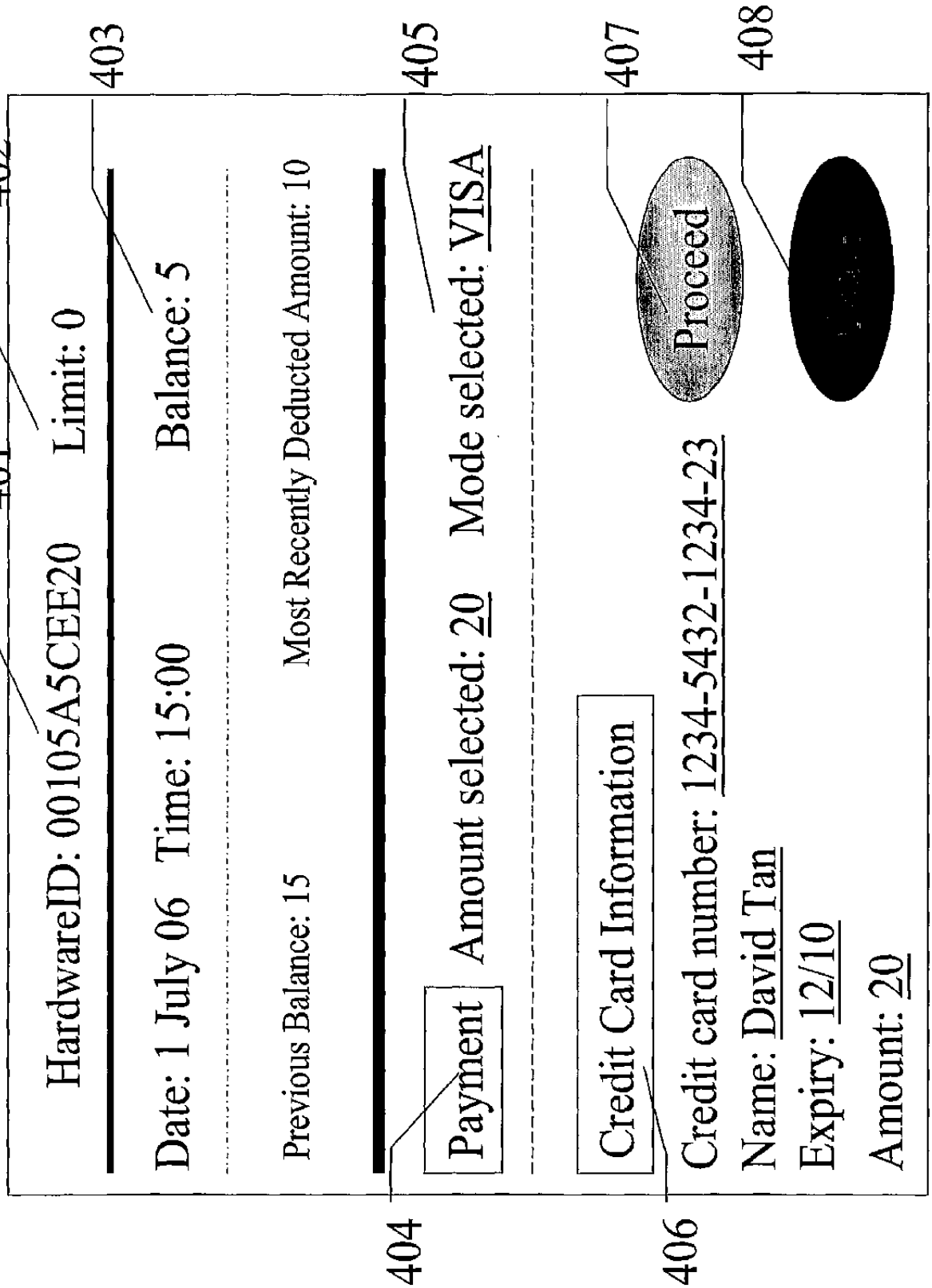

401
402
403
405
407
408
HardwareID: 00105A5CEE20 Limit: 0
Date: 1 July 06 Time: 15:00 Balance: 5
Previous Balance: 15 Most Recently Deducted Amount: 10
Payment Amount selected: 20 Mode selected: VISA
Credit Card Information
Credit card number: 1234-5432-1234-23
Name: David Tan
Expiry: 12/10
Amount: 20
Proceed
404
406

Figure 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG2006/000212

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl.

H04L 12/14 (2006.01) ***H04L 12/16*** (2006.01) ***H04L 29/06*** (2006.01)
G06Q 30/00 (2006.01) ***H04L 12/22*** (2006.01) ***H04L 29/08*** (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
DWPI & USPTO & Google: Keywords: (Network, access, address, payment, balance, services, hardware identifier and similar terms).

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CHOI et al. Enhancement of a WLAN-based internet service. ACM Mobile Networks and Applications; Special Issue: Wireless mobile wireless applications and services on WLAN hotspots, vol 10, no 3, pp 303-314, 3 June 2005. < retrieved at http://portal.acm.org/citation.cfm?id=1145911.1145916&coll=&dl=ACM&idx=J547&part=&WantType=&title=&CFID=15151515&CFTOKEN=6184618 on 7 September 2006 > See the entire document.	1-33
X	XIA et al. Secure and flexible support for visitors in enterprise Wi-Fi networks. Proceedings of the IEEE Global Telecommunications Conference, 2005 (GLOBECOM '05), vol 5, 28 November – 2 December 2005. See for example the abstract and sections 2 and 3.	1-33

[X] Further documents are listed in the continuation of Box C [X] See patent family annex

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
07 September 2006	15 SEP 2006
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **DALE E. SIVER** Telephone No : (02) 6283 2196

Form PCT/ISA/210 (second sheet) (April 2005)

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CUSHNIE, John. QoS Charging For Internet Access Networks: The Wireless QoS Gateway. Lancaster University, Computing Department, April 2003. < retrieved at http://www.comp.lancs.ac.uk/computing/users/cushniej/papers/jc_phd_latest.pdf on 7 September 2006 > See in particular sections 2.8 and 4.2.1.	1-33
X	US 2006/0153122 A1 (HINMAN et al.) 13 July 2006. See for example the abstract, fig 6 and paras [0006]-[0011].	1-33
X	US 2005/0177515 A1 (KALAVADE et al.) 11 August 2005. See for example the abstract, paras [0055] and [0293]-[0315].	1-33
X	US 2004/0102182 A1 (REITH et al.) 27 May 2004. See for example the abstract, paras [0002]-[0004], [0008]-[0010] and [0015]-[0021].	1-33

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG2006/000212

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member					
US	2006153122	WO	2006076626				
US	2005177515	WO	2005076884				
US	2004102182	AU	2002249268	EP	1246445	EP	1371220
		WO	02078316				

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.

END OF ANNEX

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 December 2008 (24.12.2008)

PCT

(10) International Publication Number
WO 2008/157639 A1

(51) International Patent Classification:
G06F 21/22 (2006.01)

(21) International Application Number:
PCT/US2008/067400

(22) International Filing Date: 18 June 2008 (18.06.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/945,359 21 June 2007 (21.06.2007) US

(71) Applicant *(for all designated States except US)*: **UNILOC CORPORATION** [US/US]; 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **RICHARDSON, Ric, B.** [AU/US]; 19200 Von Karman, Suite 400, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

(54) Title: SYSTEM AND METHOD FOR AUDITING SOFTWARE USAGE

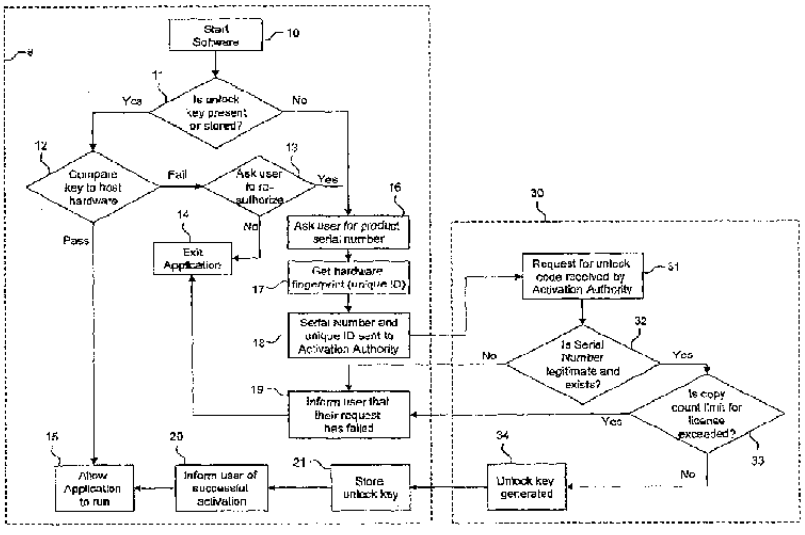


Figure 1 (Prior Art)

(57) Abstract: Systems and methods are provided for auditing and selectively restricting software usage based on, for example, software copy counts or execution counts. In one embodiment, the method comprises verifying whether the serial number for a software installed on a computing device corresponds to one of recognized serial numbers, and calculating a copy count (or software execution count) for the serial number. In response to the copy count exceeding a defined upper limit, a limited unlock key may be sent to the device. The limited unlock key may allow the software to be executed on the device for a defined time period, a defined number of executions, and/or with at least one feature of the software disabled.

WO 2008/157639 A1

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 March 2009 (26.03.2009)

PCT

(10) International Publication Number
WO 2009/039504 A1

(51) International Patent Classification:
G06F 9/445 (2006.01)

(21) International Application Number:
PCT/US2008/077245

(22) International Filing Date:
22 September 2008 (22.09.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/973,781 20 September 2007 (20.09.2007) US

(71) Applicant *(for all designated States except US)*: **UNILOC CORPORATION** [US/US]; 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **RICHARDSON, Ric, B.** [AU/US]; 19200 Von Karman, Suite 400, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(54) Title: INSTALLING PROTECTED SOFTWARE PRODUCT USING UNPROTECTED INSTALLATION IMAGE

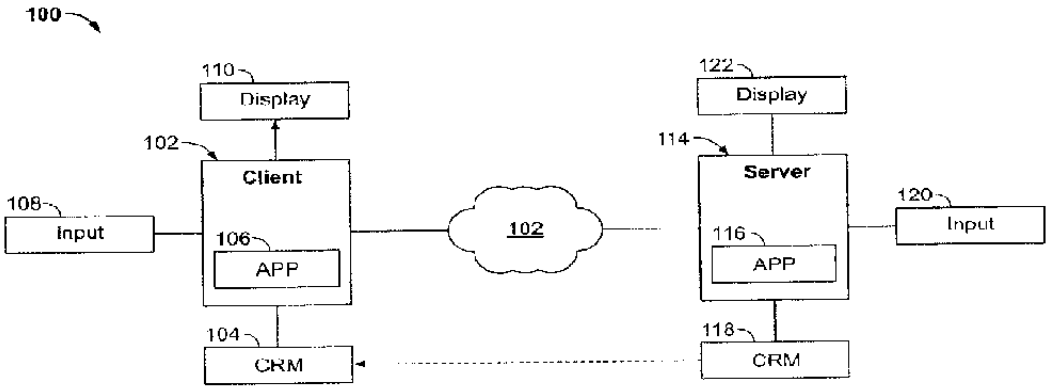


FIG. 1

(57) Abstract: An installation image for installing an unprotected software product is used to install a protected version of the same product. A protected version of the executable file is embedded in a new installation image with the original installation image, in which the unprotected version of the executable file is damaged so as to be unusable and unreadable. The new installation image causes the original installation image to operate, installing the damaged installation file and other data files. The new installation image then replaces the damaged installation file with the protected installation file.

WO 2009/039504 A1

INSTALLING PROTECTED SOFTWARE PRODUCT USING UNPROTECTED INSTALLATION IMAGE

CROSS-REFERENCE TO RELATED APPLICATION

This application claims priority pursuant to 35 U.S.C. § 119(e) to U.S. provisional application Serial No. 60/973,781, filed September 20, 2007, which application is specifically incorporated herein, in its entirety, by reference.

BACKGROUND

1. Field

The present disclosure relates to methods and systems for protecting software from unauthorized copying and reverse-engineering.

2. Description of Related Art

Many different methods of installing software and protecting installed software exist in the art. As used herein, software protection includes adding copy control and anti reverse engineering functionality to an existing application, either before or after it is compiled. Such software protection operates to prevent unauthorized copying or reverse engineering of the software that is being protected. As such, software protection does not provide core functionality to the end user or drive distribution of software products, and may require programming expertise different from what is needed for the core software to develop. Therefore, protection may often be added to a software product after the product's core functionality has been designed, implemented, and tested. It is often desired to add software protection after a software product has completed testing and has been implemented as an application installation image configured for installation on client computers. The original installation image, which

does not include the desired software protection features, must therefore be altered to include them.

As used herein, an "installation image" refers to all executable and non-executable data making up a software product application and all of its supporting data or applications, configured for encoding on a computer-readable medium or for transmission to a client for storage on or in a computer-readable medium. As used herein, "executable data," "executable code," "software executable," or generally, "executable" means that the executable data, code or software is capable of being executed by a computer processor after being installed in the computer system. Prior to being installed, the data, code or software may, or may not, be in executable form. Executable software that becomes executable after being installed may nonetheless not be executable until installation on is completed. For example, the software executable may need to be first decrypted, assembled or decoded during an installation process. The installation image makes up the encoded data package that computer users obtain, for example, when they purchase a software product encoded on an optical disc, or download a software application. In each case the installation image includes all of the encoded information required to install the software product on a client computer.

To provide software protection for an existing software product having a completed installation image, developers of software protection often rebuild the installation image of an application to include protected versions of the main software components that make up the software product to be protected. This process requires a complete rebuild of the target product installation image, and the additional testing, quality assurance and time that this entails. Thus, software protection costs may be exacerbated by the practice of adding software protection after the application has been compiled, tested and placed into production. In addition, in such cases the original software developers, testers and quality assurance personnel may have become unavailable, causing delay in adding the software protection capabilities to the target product. For example, addition or software protection may be delayed until the release of the next version of the product, resulting in considerable exposure of the unprotected

APPLE EXHIBIT 1102, Page 604 of 1048

version of the software to the risk of copyright piracy or unauthorized reverse engineering.

It is desirable, therefore, to provide a method or system for adding software protection to an existing application installation image, that overcomes the limitations of the prior art as exemplified above.

SUMMARY

The present technology enables addition of a protected edition of target software after the tested installation image of the unprotected software has been installed on a computer client, without requiring rebuilding of the installation image for the protected product.

An installation image for a software product comprises executable files and non-executable data used by the one or more executable files during operation. One of the executable files operates as an installer application for installing the remainder of the executable files and data contained in the installation image. In the alternative, or in addition, one or more applications that are not included in the installation image may participate in some part of the installation process. For example, the computer system may include a generic installer application that operates to install some or all files and data included in the installation image. Installation may include decompressing compressed data or executable files, copying decompressed data or executable files to a computer memory for storage (such as to a hard drive, for example), writing information to specific system locations, such as uninstall files or shortcuts, updating system registry files, and other operations required to install the application within the client computer system as known in the art. It may be desired to provide software protection to one or more of the executable files included within the installation image. These executable files are sometimes referred to herein as "target files" or in the singular as a "target file."

Software protection techniques as known in the art may be applied to the target files to produce a corresponding number of protected executable files. The protected executable files should operate in the same way as the target files with respect to use of

APPLE EXHIBIT 1102, Page 605 of 1048

input data, user interface, and interacting with other executable files within the installation image. However, the protected executable files may include ancillary features that prevent unauthorized copying or reverse engineering, without interfering with functionality of the target files. These ancillary features are not present in the original target files.

To provide a protected installation image, the present technology embeds or adds an existing installation image, including the target files, inside a master installation image that also includes the protected executable files. The master installation image contains master executable code for controlling and managing the installation process as well as a protected version of at least one target software executable. These two components work with the components of the original installation image to facilitate the installation of a protected version of the target software product.

Before the addition of the original installation image to the master installation image, one or more unprotected target software executables (normally all of the target software executables for which a protected executable file has been developed) are damaged so as to become inoperable. For example, some non-zero number of bits in the files may be flipped, set to zero, or set to one, such that the binary code will not operate and cannot be reverse engineered. This may be done to ensure that an unprotected version of the target software executable (or target software executables) is never installed in the event that the installation process of the master installation image is intentionally or unintentionally interrupted. Conversely, failing to damage the target files in the master installation image may create a risk of unauthorized access to a fully working unprotected version of the product, such as may occur if the master installation process is interrupted.

The master installation image may be configured with an installation executable that operates as follows. As used herein, "installation executable" refers to an executable application, software, or code configured to perform an installation process for other software and/or data. During installation of the master installation image for the protected version of the software product, the master installation image may invoke

APPLE EXHIBIT 1102, Page 606 of 1048

the embedded original installation executable, which, in turn, installs a complete copy of the damaged target software executable or executables. In the alternative, or in addition, the master installation executable may invoke an installation executable not included in the installation image, for example an installation utility application include in the computer's operating system, to install the complete copy of the damaged target software executable. Therefore, a damaged version of the original target software executable is first installed. After the original installation executable has completed execution, the master installation executable code replaces the damaged version of the original target software executable with the protected version of the target software executable, thereby completing the installation of a protected version of the software product, installed on the client computer and ready for use by the end user of the software.

A more complete understanding of the system and method for installing a protected software product will be afforded to those skilled in the art, as well as a realization of additional advantages and objects thereof, by a consideration of the following detailed description. Reference will be made to the appended sheets of drawings which will first be described briefly.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 shows an exemplary computer system in which various aspects of the present technology may be implemented.

Fig. 2A is a block diagram showing exemplary elements of an installation image and related elements prior to being embedded inside a master installation image.

Fig. 2B is a block diagram showing exemplary elements of a master installation image with the embedded original installation image shown in Fig. 1A.

Fig. 3 is a flow chart showing an exemplary process flow implemented by operation of a master installation image on a client computer.

Fig. 4 is a flow chart showing an exemplary method for preparing a master installation image.

DETAILED DESCRIPTION

The novel technical solution disclosed herein overcomes the limitation of the prior art to regarding installation of newly protected software products. In the detailed description that follows, like element numerals are used to indicate like elements appearing in one or more of the figures.

Fig. 1 shows an exemplary client-server system 100 in which various aspects of the present technology may be implemented. A client computer 102 may include processor, memory, I/O, and other components as known in the art, and one or more devices for reading and/or writing to a computer-readable medium 104. For example, the client 102 may comprise an internal magnetic media hard drive, a DVD or CD-ROM optical drive, and a USB port for connecting to a flash memory drive. The client may also include processor memory 106 used for operating one or more software applications as described herein. In general, software applications are designed to receive some form of input from an external device or process, and process the input to provide some tangible output to an output device or to another process. For example, an application may be designed to receive keyboard input from a keyboard 108, process the input, and provide output to a display device 110, such as a computer monitor. Optionally, the client 102 may be adapted to connect to a wide area network 112 or other computer network for communicating with one or more other computers, for example a server 114. Server 114 may comprise a computer with components as known in the art, including, for example, a device for reading and/or writing to a computer-readable medium 118, processor memory 116 for operating one or more applications, at least one input device 120 and a display device 122. In application of the present technology, a server such as server 114 may be used to prepare a master installation image, which may be transmitted to one or more clients such as client 102 via a wide area network or other communications network, or physically transported to the client on a tangible computer-readable medium.

An exemplary configuration and operation of a master installation image is shown in Figs. 2A, 2B and 3. Figs. 2A-2B show exemplary components of a protected

software installation image. An original compiled installer component may be produced in the form of an installation image file 200, sometimes referred to herein as an original installation image, that contains at least one target software executable 206 and a combination of additional files and data 208. The target software executable 206 may be in the form of a binary-coded file or file portion configured to be loaded into a processor memory and thereby cause a client computer to carry out the processes for any desired application, as known in the art. Data, in contrast, is not executable and may be used as input for an application-driven computer process, and/or may represent output from the computer process. Some additional executable code 210 may placed in the installation image so as to load prior to loading the target software executable 206 to control and manage an installation process. In the alternative, or in addition, the installation image may be configured for installation by a pre-existing installer application on the target computer, which may obviate the need for an installation executable in the installation image itself. An installation process may include, for example, decompressing compressed data or executable files from the installation image 200, copying decompressed data or executable files to a computer memory for storage, writing information to specific system locations, such as uninstall files or shortcuts, updating system registry files, and other operations required to install the application within the client computer system, as known in the art. The installation executable 210 and target software executable may comprise two or more separate files, or in the alternative, may be integrated into a single file or portion of the installation image 200. After installation, the target software executable 206 may comprise one or more separate files within a file system of the target computer.

The target software executable 206 included in the original installation image may lack certain desired software protection features, and therefore it is desired to update it with a more thoroughly protected version. The original executable 206 may have some protection features; it need not be entirely devoid of such features.

To update the installation image 200 with improved protection features present in a protected version 215 of the target software executable, a new master installation

APPLE EXHIBIT 1102, Page 609 of 1048

executable 202 may be created. The protected software executable 215 should not include any changes that will affect operation of the executable in relation to the data 208 or installation configuration implemented by the installer 210, or in the alternative, by an external installation utility, if used. If such changes are necessary, the technical solution described herein may have unpredictable results, and is not recommended. The present solution addresses circumstances wherein the addition of desired protection features does not change operational features of the target software executable with respect to the original installation image configuration.

The new master installation image 202 may be assembled by embedding or integrating the original compiled installation image 200 that includes all the components of the compiled installer in their compiled state for installation in addition to a protected version of the target software executable 215, which may operate in the same way as the target software executable 206 with the addition of protection capabilities. The master installation file 202 may also comprise additional master installation executable code 214 configured to load and execute with the highest priority within the master installation image 202, to initiate, control and manage the installation process as explained in more detail below in connection with Fig. 3. Initially, the master installation executable code 214 may be configured to trigger the original compiled executable file 200 which, in turn, controls and manages its own installation process.

Before the original compiled installation image 200 is integrated with the master installation image 202, the target software executable 206 contained in the installation executable 200 may be damaged to transform it into a damaged executable file 213. This may be done to ensure that after installation, the target software executable in its damaged state 213 will not run and cannot be decompiled, in case the installation process carried out by the master installation image 202 is somehow interrupted. Damage may be done in a random, irreversible manner so that the original file cannot be restored by someone who possesses or guesses the algorithm used to damage the target software executable. For example, a damage algorithm may randomly select and flip the value of bits within the target software executable, thereby corrupting it,

rendering it inoperable and impossible to decompile. As such, damaging the file, which irreversible corrupts the executable file, is distinct from encryption, which places the file in a temporarily inoperable state that may be reversed by anyone possessing a decryption key. Of course, instead of damaging the target software executable 206 to produce a damaged executable 213, the target software executable may be left undamaged or merely encrypted. These alternatives, however, may entail increased risk of unauthorized use. The master installation image may be used to delete or overwrite the damaged executable 213, so there should be no reason to preserve its operability.

After being assembled, the master installation image 202 may comprise one or more files made up of component parts, as follows. The installation image 202 may comprise a master installation executable file or file portion, which is configured to initiate and control the master installation process. The installation image 202 may further comprise the protected version 215 of the target software executable, the original installation executable 210, the damaged original (unprotected) target software executable 213, and the original data 208.

Fig. 3 shows exemplary steps of an installation method 300 such as may be implemented by a master installation image operating on a client computer. Method 300 may be initiated by running the master installation image, which may be configured to operate first when after the master installation image is copied to the client computer and selected for installation by a computer user, or automatically run. Upon initializing, the master installer application may execute the original installer executable at 304 or external installation application if applicable, and also initiate a wait process 310 that waits for the original installer application to complete execution. At 306, the original installer application may install all files from the original installation image, which may include a damaged version of the target application main executable, or in the alternative, an undamaged or encrypted version. When the original installer application completes execution at 308, the master installer monitoring process 310 may trigger a replacement process 312, that replaces the damaged version of the target software

executable (or in the alternative, the undamaged or encrypted original target software executable) with the protected version of the original target software executable. For example, the replacement process may overwrite the damaged target software executable with the protected version of the target software executable having an identical file name in the client file system, or may delete the original target software executable from the file system and then write the protected version of the target software executable, having a filename identical to the deleted file, to the client file system. The target software executable may comprise more than one file, in which case each file may be replaced by a protected version of the file under the same file name. Subsequently the master installer application completes execution 314 and terminates. The result of method 300 should be that the protected application version is installed on the client system with all necessary data, registry entries, DLL files, and other system files necessary to facilitate proper operation of the application on the client, and the original target software executable no longer exists in the client computer system as an operable file. It may still exist on the client as part of the master installation image, if the master installation image is not deleted. However, if damaged as disclosed herein, it cannot be extracted for unauthorized use.

Fig. 4 shows exemplary steps 400 for preparing a master installation image, in accordance with the foregoing. The master installation image may be prepared, for example, using a server for distribution to multiple target clients. At 402, the server may receive a copy of the original installation image. This may be the original image as prepared for distribution by the original software developers. A protected version of the target software executable may be received by the server at 404. This should possess the attributes described above, and may be prepared as known in the art of software protection. Methods for coding software protection features are not the concern of the present disclosure, and are known in the art. At 406, a master installation executable for the master installation image may be prepared and received for use by the server. The master installation executable should be programmed to incorporate the essential features diagrammed in Fig. 3 and otherwise disclosed herein. At 408, the original

APPLE EXHIBIT 1102, Page 612 of 1048

target software executable included in the original installation image may be disabled, that is, damage as described above. For example, randomly selected bits in the executable may be flipped. An algorithm may be employed to randomly select and flip the bits in a pattern sufficiently complex and massive so as to make repair of the target software executable virtually impossible. For example, if the target software executable comprises 10^8 bits of data, the algorithm may flip 10 bits at each of, for example, 1000 randomly selected locations. At 410, the server may be used to assemble the master installation image as diagrammed in Fig. 2B. The finished master installation image may be output 412, for example, by transmitting to a target client or by encoding on a computer-readable medium. In general, the server may transmit multiple copies of the master installation image to a corresponding number of target clients via a network connection. In the alternative, or in addition, duplicate copies of the master installation image may be encoded on optical discs or in flash memory devices for physical distribution to target clients.

Having thus described exemplary embodiments for installing a protected software product using an unprotected installation image, it should be apparent to those skilled in the art that certain advantages of the within system have been achieved. It should also be appreciated that various modifications, adaptations, and alternative embodiments thereof may be made without departing from the scope and spirit of the present technology. For example, methods and systems for installing a single target software executable have been illustrated, but it should be apparent that the novel concepts described above may be applied by one of ordinary skill to multiple executables within an installation image to thereby realize the benefits described herein. The following claims define the scope of what is claimed.

APPLE EXHIBIT 1102, Page 613 of 1048

CLAIMS

What is Claimed is:

1. A method for installing a protected software product using an unprotected installation image, comprising:

initiating an installation process on a client computer system for installation of a protected software product;

in response to initiating the installation process, installing an unprotected software executable file included in a first installation image into the client computer system;

deleting the unprotected software executable from the client computer system after installing the unprotected software and before terminating the installation process; and

writing a protected version of the software executable to a memory of the client computer in place of the unprotected software executable that is deleted from the client computer system, to complete installation of the protected software product.

2. The method of claim 1, wherein installing the unprotected software executable file includes installing only a damaged version of the unprotected software executable that is deliberately damaged so as to be inoperable.

3. The method of claim 1, wherein installing the unprotected software executable file is performed using an installation executable included in the first installation image.

4. The method of claim 1, wherein writing the protected software executable to the memory comprises saving the protected software executable as an executable file in a file system of the client computer.

5. The method of claim 4, wherein saving the protected software executable as an executable file comprises assigning a file name to the executable file that is identical to a prior file name previously used to save the unprotected software executable in the file system.

6. The method of claim 5, wherein writing the protected software executable as an executable file causes the unprotected software executable to be overwritten and deleted from the file system.

7. The method of claim 1, wherein initiating the installation process is performed using a second installation image comprising the first installation image, the protected software executable, a first executable for initiating the installation process, and second executable for writing the protected version of the software executable to the client memory.

8. The method of claim 1, further comprising reading the installation image from a computer-readable medium.

9. The method of claim 1, further comprising receiving the installation image at the client computer from a remote server.

10. A computer-readable medium encoded with instructions comprising:

initiating an installation process on a client system for installation of a protected software executable;

in response to initiating the installation process, installing an unprotected software executable file included in a first installation image into the client computer system;

deleting the unprotected software executable from the client system after installing the unprotected software and before terminating the installation process; and

writing the protected software executable to a memory of the client computer in place of the unprotected software executable that is deleted from the client computer system, to complete installation of the protected software product.

11. The computer-readable medium of claim 10, further encoded with instructions for installing only a damaged version of the unprotected software executable that is deliberately damaged so as to be inoperable.

12. The computer-readable medium of claim 10, further encoded with instructions for installing the unprotected software executable file using an installation executable included in the first installation image.

13. The computer-readable medium of claim 10, further encoded with instructions for saving the protected software executable as an executable file in a file system of the client.

14. The computer-readable medium of claim 13, further encoded with instructions for assigning a file name to the executable file that is identical to a prior file name previously used to save the unprotected software executable in the file system.

15. The computer-readable medium of claim 13, further encoded with instructions for writing the protected software executable as an executable file, thereby causing the unprotected software executable to be overwritten and deleted from the file system.

16. The computer-readable medium of claim 13, further encoded with instructions for initiating the installation process using a second installation image comprising the first installation image, the protected software executable, a first executable for initiating the installation process, and second executable for writing the protected software executable to the client memory.

17. A method for altering an unprotected software installation image to install a protected software version of an unprotected executable file included in the installation image, instead of installing the unprotected executable file, the method comprising:

receiving a first installation image configured for installing a first executable file in a target computer;

receiving a second executable file that is identical in function to the first software executable file except for incorporating additional code configured to protect the second executable file from unauthorized use;

receiving a third executable file configured for replacing the first executable file with the second executable file in a computer-readable medium, in response to completion of installation of the first executable file in the computer-readable medium; and

integrating the first installation image, the second executable file, and the third executable file to produce a second installation image configured for installing the second executable file instead of the first executable file.

18. The method of claim 17, further comprising receiving a fourth executable file configured for initiating operation of the first installation image on the target computer.

19. The method of claim 18, further comprising integrating the fourth executable file into the second installation image.

20. The method of claim 17, further comprising disabling the first executable file included in the second installation image so as to render the first executable file inoperable.

21. The method of claim 20, further comprising irreversibly altering information in the first executable file to disable the first executable file.

22. The method of claim 20, further comprising randomly altering information in the first executable file to disable the first executable file.

23. The method of claim 17, further comprising transmitting the second installation image to a target computer.

24. The method of claim 17, further comprising writing the second installation image to a computer-readable medium.

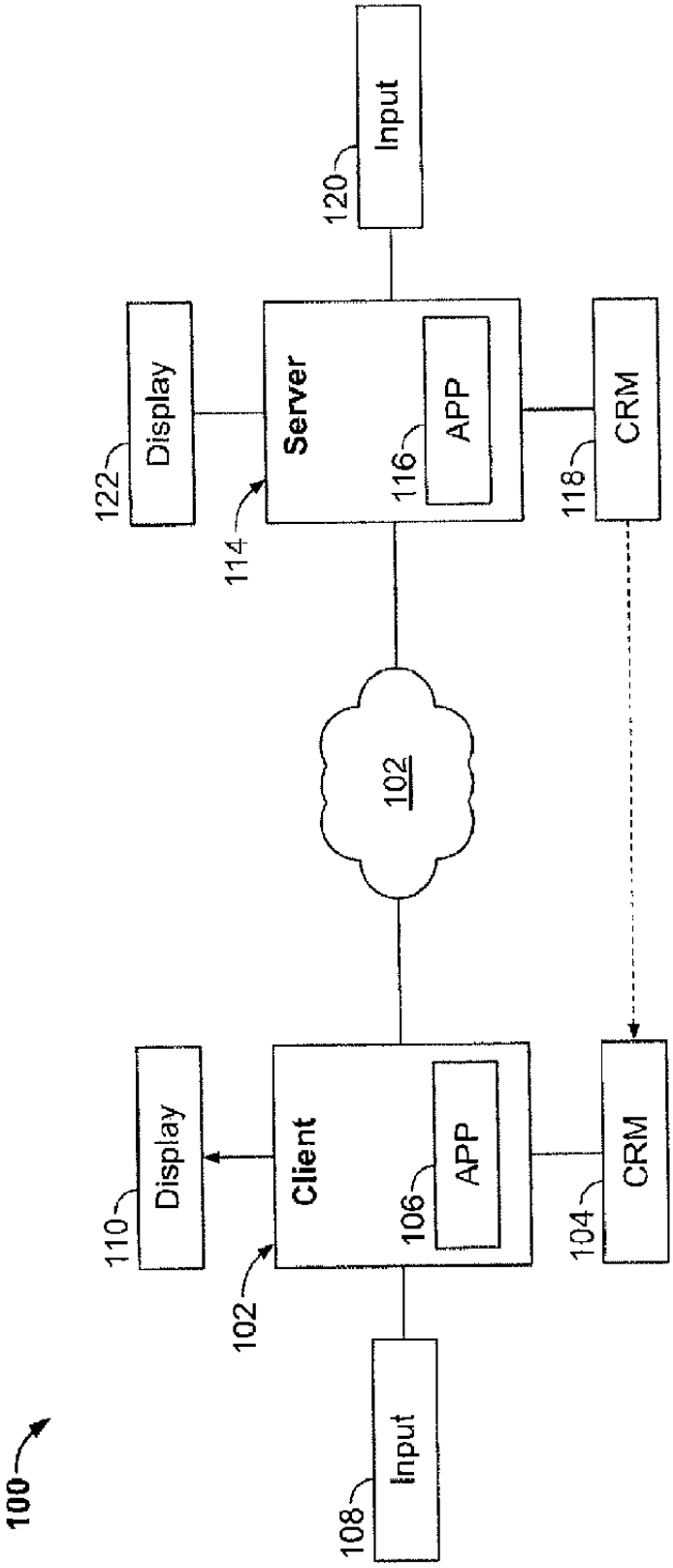

100
Input
120
Display
122
Server
APP
116
CRM
118
114
102
Display
110
Client
APP
106
CRM
104
102
Input
108

FIG. 1

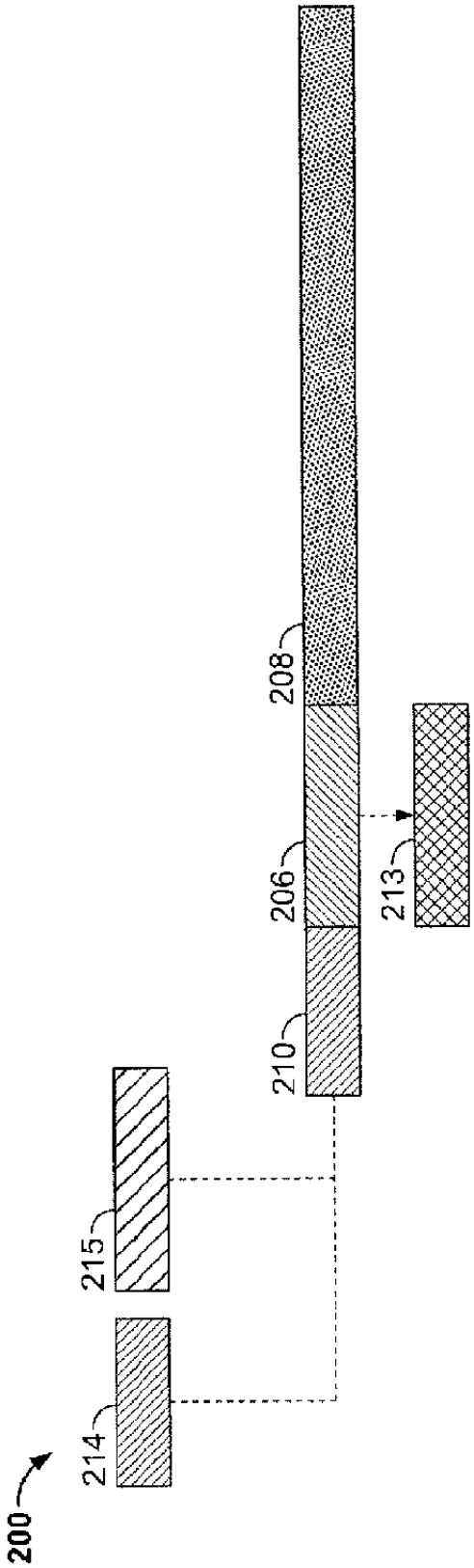


FIG. 2A

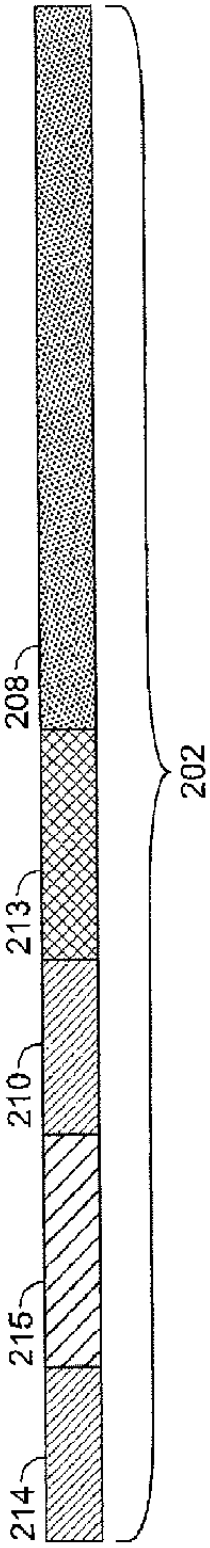


FIG. 2B

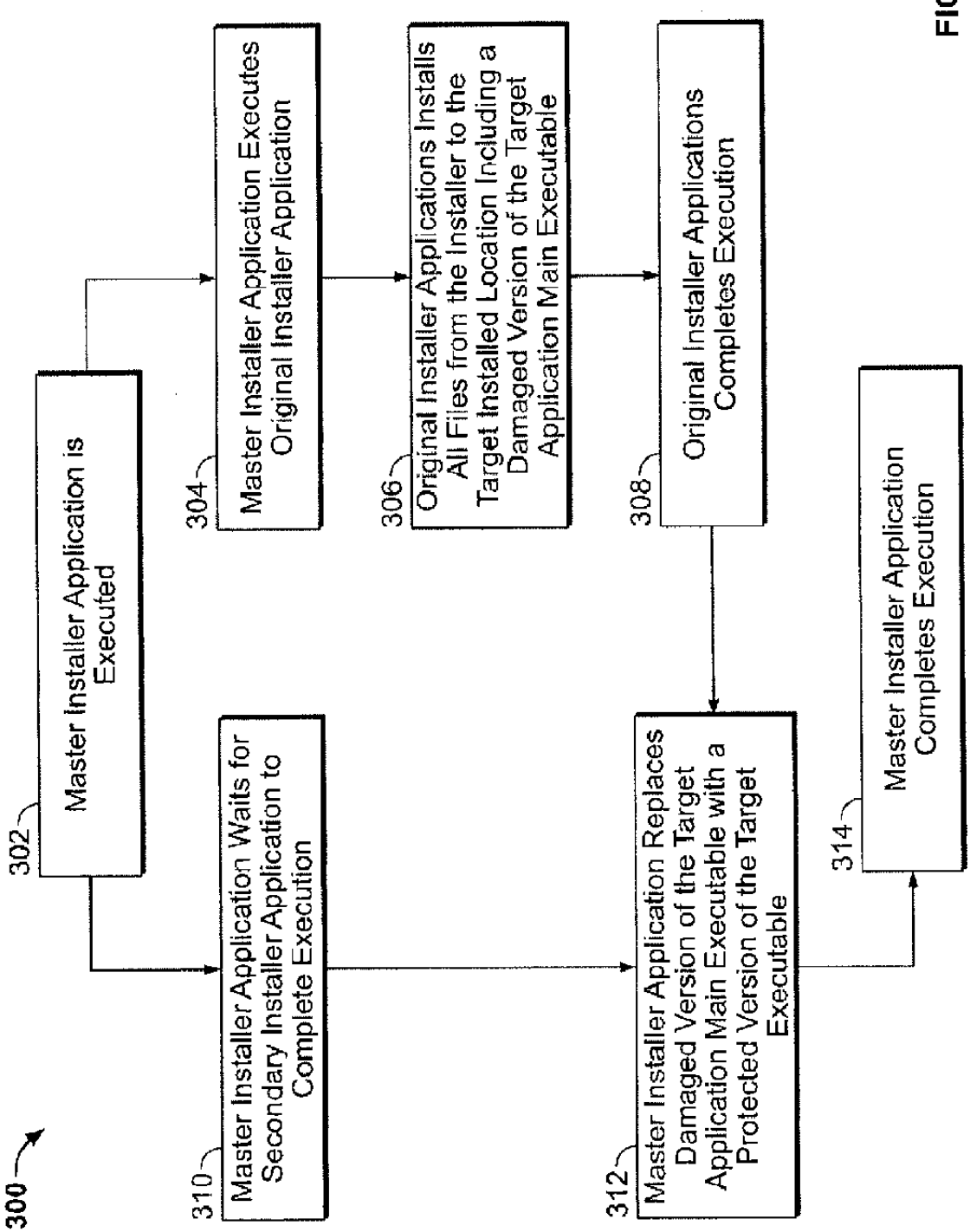

300
302
Master Installer Application is Executed
304
Master Installer Application Executes Original Installer Application
306
Original Installer Applications Installs All Files from the Installer to the Target Installed Location Including a Damaged Version of the Target Application Main Executable
308
Original Installer Applications Completes Execution
310
Master Installer Application Waits for Secondary Installer Application to Complete Execution
312
Master Installer Application Replaces Damaged Version of the Target Application Main Executable with a Protected Version of the Target Executable
314
Master Installer Application Completes Execution

FIG. 3

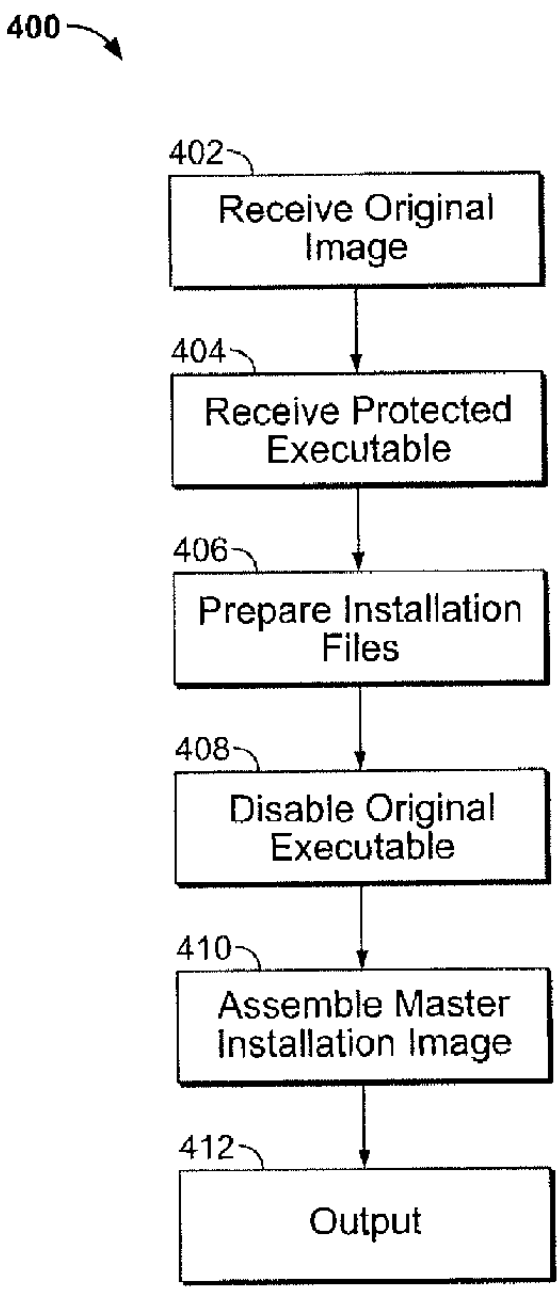


FIG. 4

APPLE EXHIBIT 1102, Page 622 of 1048

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2008/077245

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/445

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2005/172280 A1 (ZIEGLER JEREMY R [US] ET AL) 4 August 2005 (2005-08-04) abstract; figures paragraphs [0002], [0008], [0009], [0016] - [0018]; claims	1-24
A	EP 1 637 961 A (MICROSOFT CORP [US]) 22 March 2006 (2006-03-22) the whole document	
A	US 2006/161914 A1 (MORRISON SHANE A [US] ET AL) 20 July 2006 (2006-07-20) the whole document	

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 December 2008	10/12/2008

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Wiltink, Jan Gerhard

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2008/077245

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005172280 A1	04-08-2005	NONE	
EP 1637961 A	22-03-2006	AU 2005203664 A1	30-03-2006
		BR PI0503691 A	25-04-2006
		CA 2515711 A1	15-03-2006
		CN 1758609 A	12-04-2006
		JP 2006085714 A	30-03-2006
		KR 20060050436 A	19-05-2006
		MX PA05008665 A	22-05-2006
		US 2006059541 A1	16-03-2006
		US 2006059542 A1	16-03-2006
		US 2006059555 A1	16-03-2006
US 2006161914 A1	20-07-2006	NONE	

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2009 (22.05.2009)

PCT

(10) International Publication Number
WO 2009/065135 A1

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/US2008/083809

(22) International Filing Date:
17 November 2008 (17.11.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/988,778 17 November 2007 (17.11.2007) US

(71) Applicant *(for all designated States except US)*: **UNILOC CORPORATION** [US/US]; 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **RICHARDSON, Ric, B.** [AU/US]; 19200 Von Karman, Suite 400, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR ADJUSTABLE LICENSING OF DIGITAL PRODUCTS

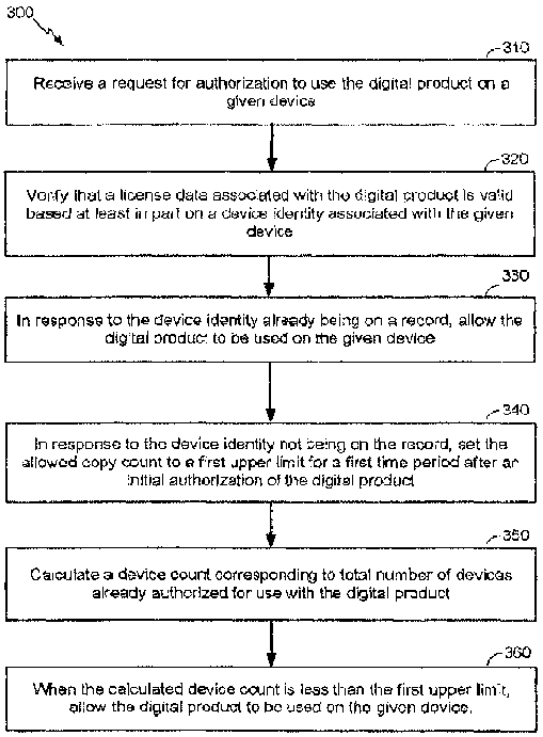


Figure 3A

(57) Abstract: Techniques are provided for adjusting the number of devices allowed to use a digital product (e.g., software) under a license. In one embodiment, the technique may involve setting the allowed number of devices to a first upper/lower limit for a first time period, and, after the first time period has expired, increasing/lowering the allowed number of devices to a second upper/lower limit for a second time period. The technique may involve, readjusting the allowed number for a third time period, thereby allowing for a changing number of device installations of the digital product.

WO 2009/065135 A1

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

SYSTEM AND METHOD FOR ADJUSTABLE LICENSING OF DIGITAL PRODUCTS

Cross-Reference to Related Application(s)

This application claims priority pursuant to 35 U.S.C. §119(e) to U.S. Provisional Application No. 60/ 988,778, entitled "SYSTEM FOR ADJUSTABLE DIGITAL LICENSING OVER TIME," filed November 17, 2007, which application is specifically incorporated herein, in its entirety, by reference.

Background of the Invention

Field of the Invention

The present application relates generally to managing software use, and more specifically to systems and methods to enable the monitoring and adjusting software usage under a software license.

Description of the Related Art

A common capability of digital product license systems is the ability to control how many devices are allowed to be used with each product license which is usually sold to an individual customer or company. For example U.S. Patent No. 5,490,216 refers to a system where a license is given to an individual, but in turn that license is linked to a specific personal computer thereby limiting the copyright holders exposure to copyright abuse if the user decided to share their license with other unauthorized users.

A problem that has arisen over time is the fact that consumers of software have normal patterns of use that include the installation and use of digital products on multiple devices. For example a person may wish to buy software and use it on three computers at their home, a computer at work, a mobile computer and the computers at their holiday home and their parent's house. In addition to these uses, computers are also bought, sold and replaced so over time maybe two or three times this number of

computers may be used by the user over time with a legitimate need to install and use the software on every computer.

Publishers of digital products have a dilemma in that they may want their customers to receive the normal freedom to use the software that they have purchased but they also do not want the software licenses to be freely shared amongst end users or even in worst case shared anonymously over the Internet resulting in massive piracy and copyright abuse of the product.

To solve this problem some publishers have set a relatively high device to license ratio in their control systems in the hope that customers will not exceed the maximum number of devices per license. An example of this is Apple iTunes which enables customers to play a purchased music file on up to a preset number (e.g., five) of devices (e.g., PCs) per license before being requested to buy an additional license. They have also implemented a system that allows customers to turn off the license rights of individual devices with regard to a specific music file license and therefore free up that device installation so that the music file can be used on one additional device.

While this method does go some way to appeasing the problem of a normal customers usage expectations, it does not take into consideration the normal attrition that occurs with the purchase and upgrade of personal computing devices or the like and places an expectation on the user to go through a number of involved steps to retain their rights to use the software. Accordingly, there is a need for an improved technique for allowing for a changing number of device installations on a per license basis over time.

Summary of the Invention

The following presents a simplified summary of one or more embodiments in order to provide a basic understanding of such embodiments. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later.

In accordance with one or more embodiments and corresponding disclosure thereof, various aspects are described in connection with adjusting a license for a digital product over time. The license may comprise at least one allowed copy count corresponding to a maximum number of devices authorized for use with the digital product. In one embodiment, a system for adjustable licensing includes: a communication module for receiving a request for authorization to use the digital product from a given device; a processor module in operative communication with the communication module; and a memory module in operative communication with the processor module.

The memory module may include executable code for the processor module to: (a) verify that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device; and (b) in response to the device identity already being on a record, allow the digital product to be used on the given device.

The memory module may further include executable code for the processor module to: (c) in response to the device identity not being on the record, set the allowed copy count to a first upper limit for a first time period; (d) calculate a device count corresponding to total number of devices already authorized for use with the digital product; and (e) when the calculated device count is less than the first upper limit, allow the digital product to be used on the given device.

In related aspects, the processor module may be adapted to: (a) in response to the device identity not being on the record, after the first time period has expired, set the allowed copy count to a second upper limit for a second time period; (b) recalculate the device count; and/or (c) when the recalculated device count is less than the second upper limit, allow the digital product to be used on the given device. For example, the second time period may comprise a defined number of days since the initial authorization. The processor module may be adapted to, in response to the calculated device count equaling the second upper limit, send a warning regarding the allowed copy count to the given device. The processor module may be adapted to, in response to the calculated device count exceeding the second upper limit, deny the request for authorization.

APPLE EXHIBIT 1102, Page 629 of 1048

In further related aspects, the processor module may be adapted to: (a) in response to the device identity not being on the record, after the second time period has expired, set the allowed copy count to a third upper limit; (b) recalculate the device count; and (c) when the recalculated device count is less than the third upper limit, allow the digital product to be used on the given device. The processor module may be adapted to, in response to the calculated device count equaling the third upper limit, send a warning regarding the allowed copy count to the given device. The processor module may be adapted to, in response to the calculated device count exceeding the third upper limit, deny the request for authorization.

To the accomplishment of the foregoing and related ends, the one or more embodiments comprise the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects of the one or more embodiments. These aspects are indicative, however, of but a few of the various ways in which the principles of various embodiments may be employed and the described embodiments are intended to include all such aspects and their equivalents.

Brief Description of the Drawings

Figure 1 is an exemplary set of license rules that may be implemented to adjust the number of device installations on a per license basis over time.

Figure 2 shows an exemplary approach for adjusting a license for a digital product.

Figure 3A shows one embodiment for a method for adjusting a license for a digital product.

Figure 3B shows several sample aspects of the method shown in Figure 3A.

Figure 4 shows one embodiment for a system for adjusting a license for a digital product.

Detailed Description

Various embodiments are now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following

APPLE EXHIBIT 1102, Page 630 of 1048

description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of one or more embodiments. It may be evident, however, that such embodiment(s) can be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing one or more embodiments.

The techniques described herein allow for a changing number of device installations on a per license basis over time. Aspects of the techniques may include a customer feedback system that warns a user when they are nearing the limit of their device installation ceiling for their license. An example scenario could be as follows. A software publisher wants to commence distribution of a software product and to minimize unauthorized copying of their software. Their license may state that the publisher authorizes the user to use their software on up to, for example, five devices, but that the publisher reserves the right to increase this limit at their own discretion. The customer installs the software on the three computers they have at home. Each time the software connects to a license management server controlled by the publisher over the Internet to ensure that the device limit for the individual license has not been exceeded.

The customer may choose to install the same software on their personal computer (PC) at work. Upon contacting the publishers license management server over the Internet a message is displayed to the user warning them that they are nearing the limit of their device count for their license.

Two weeks later the user wishes to install their software on the two computers they own at the customers holiday home. If the publisher uses the proposed invention the maximum number of devices for the license may have been adjusted to accommodate a reasonable small increase in the number of devices linked to a specific license and both PCs may be allowed to install and run even though the publishers stated device limit per license is five.

Then three months later, the user experiences water damage from a flood in their house and a new PC is purchased. Upon installation of the protected software the invention will allow the user to obtain additional device installations from the publishers license management server for the same license (e.g., up to a total of seven devices) even though the device limit is initially set to five. However, if that user shares

their license with all the computer users in a college dormitory, the invention can be set to stop wholesale abuse of the license terms, as described in further detail herein.

In accordance with one or more aspects of the embodiments described herein, there is provided a system for adjustable digital licensing over time allows a software user to increase the number of devices they can use with a particular software license over the period of ownership of that license. The terms or rules 60 of an exemplary software license are shown in Figure 1. For example, initially, the publisher or distributor of the software sets rules 60 that govern the use of the software on a specific number of devices. The number of devices allowed to run the software in an authorized or enabled state may increase over time to reflect the normal usage pattern of software users where the user adds devices, replaces or upgrades devices over time. The rules 60 may reflect this pattern of an increasing number of devices authorized over time. For the first five days of the users use of the software a total of five devices can be authorized on new devices. For the next twenty-five days until the thirtieth day after first authorization, the user is allowed to authorize a total of seven new devices. After the first thirty days an additional four devices can be authorized, delivering the maximum number of copies on separate devices under the license which, in this example embodiment, is eleven.

It is noted that the various numbers used to describe the embodiments herein, such as, for example, the allowed copy counts, the maximum number of devices authorized for use, the upper limit on the number of devices for a given time period, or the like, are purely exemplary, and that other numbers, data, values, or algorithms may be used in lieu of the exemplary numbers herein.

In related aspects, Figure 2 shows an example embodiment of a software system that is designed to manage and implement the rules under a license, such as, for example, the licensing terms 60 described in Figure 1. Device locked license systems such as described in U.S. Patent No. 5,490,216, entitled "SYSTEM FOR SOFTWARE REGISTRATION," which is specifically incorporated herein, in its entirety, by reference, allow a software license to be locked to a license agreement and specific authorized devices. With continued reference to Figure 2, there is shown a system comprising a device 50 that requests authorization via a software process, and a

licensing authority 55 that may be a software system that represents the publisher or distributors interests and regulates the number of devices that can be used with each license.

Typically the device 50 requesting authorization collects license related information 10 and unique device identifying information 11, compiles the collected information into a communication and sends it to the authorization authority 55. Upon receipt of this communication from the device 50, the license authority 55 checks that the license information is valid (step 13). If the request fails, an authorization is disallowed (step 14) and the device based software is sent a message to this effect. In practice this may involve further action by the device based software to notify the user of the failure to authorize and then either terminate the software or allow the software to continue in some form of trial mode or the like.

If the request for authorization 12 includes license information/data that is valid, the license information checking process (at step 13) will pass and the requesting devices unique identity information 11 is checked to see if it exists in the database of prior authorizations 15. If the device identity exists (step 16), meaning that the software has been successfully registered on the same device in the past, then according to the license terms 60 for the software a re-authorization is automatically allowed (step 17). A communication allowing the software to continue in an authorized state is passed to the requesting device software 50 and the software on the device is subsequently authorized (step 18) and allowed to run.

If the unique identity of the device 11 is not in the authorization database 15 of previous device requests, then the licensing authority 55 checks to see if the new authorization request is the first request or is a subsequent request that has occurred in the first five days from the date of the first successful authorization (step 19).

At step 19, if the request is within the first five day period, the authorization database 15 is consulted for a count of how many successful authorizations for new devices have been allowed. Under the license rules 60, if the device count is less than five then a message is sent to the request device that allows the software to continue in an authorized state (step 18). If the device count is equal to five then the licensing authority 55 may send a message to the requesting device 50 allowing the device to run

in an authorized state (step 18), but also may optionally inform the user that the limit of the number of devices available to run under this license has been reached and that subsequent requests for authorization may be denied in the short term (step 22).

If the count of devices authorized for use with the specific license 10 is greater than five (step 23), then the licensing authority 55 sends a message denying authorization (step 25) and the user is optionally notified that the limit of devices that can be authorized with their license terms has been exceeded (step 24). In practice, the software on the requesting device 50 may subsequently terminate the software or may allow the software to run in a limited trial mode if this is available.

If the number of days since the first authorization of a device for the license 10 is not less than six (step 19), then the licensing authority tests the time elapsed from the first successful authorization to see if it is less than thirty-one days since the date and time of the first successful authorization (step 26). If this test at step 26 is successful (i.e., if the time elapsed since the first successful authorization is less than thirty-one days), then a test is made to see if the count of successful new device authorizations is less than seven (step 27). If this is so, a communication is made to the requesting device 50 authorizing the device 50 to run the software (step 28). If the new device count is equal to seven (step 29), then the user is warned that their device limit has been reached (step 30) and the device 50 is subsequently authorized to run (step 28).

However, if the new device count is greater than seven (step 31), a communication is made to the requesting device 50 that the authorization is denied (step 33) and optionally the user is notified that their license device count has been exceeded (step 32).

If the number of days since the first successful authorization is greater than thirty days (step 34), the device count for the license 10 is checked in the authorization database 15 and the device count for the license 10 retrieved. If the number of successful new device authorizations is ten or less (step 35), then the device authorization is allowed (step 36). If the device count is equal to eleven (step 37), then the user is optionally warned that they have reached the limit (step 38) and the device 50 is authorized to run (step 36).

APPLE EXHIBIT 1102, Page 634 of 1048

However if the device count is greater than eleven (step 39), then a communication is made to the requesting device 50 that the user be optionally notified that the maximum number of allowed devices under terms of the license has been exceeded (step 40) and the authorization is denied (step 41).

The result is a license system that allows consumers of software to load their software on new or replacement devices as they are purchased over time without exposing the publisher to copying abuses that is common amongst software pirates and casual software copiers.

In one alternative embodiment, there is provided a license management system that is linked to a fixed calendar date rather than the date of first successful authorization. This approach can be used for marketing and distribution purposes such as specifying specific periods of high copy counts to encourage word of mouth and user to user sharing but later restricting the device count to encourage people to begin paying for copies that have been intentionally shared.

It is noted that the example embodiment of Figures 1 and 2 is simple for the purposes of understanding but can include any number of evaluation periods, not just the five, thirty and unlimited day periods described in the example. Also the number of notification stages can be indefinitely expanded, for example the user could be given a polite message encouraging them to be careful with making copies when they are two copies away from their count limit and a stronger message when it is their last copy before being denied authorizations. Messages could also optionally tell the user how many days they have to wait before additional device authorizations will be available.

It is further noted that in Figure 1 and 2 the allowed copy count increases over time. An alternative embodiment could be used where the allowed copy count decreases over time. This may be useful in a situation, for example, where the publisher supplies their software with a fairly open device count license rule but discovers individual instances of copy abuse and decides to limit the license terms of those specific licenses.

The described system could also be used with authorizations for software that is rented or otherwise allowed to be used for a specific period of time or number of uses rather than indefinitely as in the example embodiment of Figures 1 and 2.

APPLE EXHIBIT 1102, Page 635 of 1048

Another alternative embodiment of the above scenarios could include an algorithm rather than an arbitrary value in calculating both the time period for the calculation of the device count, and the device count related to that specific measured time period. For example, the algorithm for the available device count could be equal to the number of elapsed days since the first successful activation divided by five in brackets plus five. Using the example algorithm a device count of five would be available from day one, and a device count of eleven at day thirty and so on.

In yet another alternative embodiment, the techniques described herein may be used for security applications where access is granted to data or some other valuable or important item as a result of a successful authorization rather than in the example of Figures 1 and 2 where it is a license that is being granted.

In accordance with one or more aspects of the embodiments described herein, there is provided a method for adjusting a license for a digital product over time. The license rules may comprise at least one allowed copy count corresponding to a maximum number of devices authorized for use with the digital product. With reference to the flow chart shown in Figure 3A, there is provided a method 300 that may involve receiving a request for authorization to use the digital product on a given device (step 310). The method 300 may further involve verifying that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device (step 320).

In response to the device identity already being on a record, the method 300 may involve allowing the digital product to be used on the given device (step 330). In response to the device identity not being on the record, the method 300 may involve setting the allowed copy count to a first upper limit for a first time period after an initial authorization of the digital product (step 340). The method 300 may further involve calculating a device count corresponding to total number of devices already authorized for use with the digital product (step 350), and when the calculated device count is less than the first upper limit, allowing the digital product to be used on the given device (step 360).

With reference to Figure 3B, in one embodiment, the method 300 may also involve, in response to the device identity not being on the record, after the first time

APPLE EXHIBIT 1102, Page 636 of 1048

period has expired, setting the allowed copy count to a second upper limit for a second time period (step 370). The method 300 may further involve recalculating the device count (step 372), and when the recalculated device count is less than the second upper limit, allowing the digital product to be used on the given device (step 374).

With continued reference to Figure 3B, at step 380, the method 300 may also involve, in response to the device identity not being on the record, after the second time period has expired, setting the allowed copy count to a third upper limit. The method 300 may further involve recalculating the device count (step 382), and when the recalculated device count is less than the third upper limit, allowing the digital product to be used on the given device (step 384).

In accordance with one or more aspects of the embodiments described herein, there is provided a system for adjusting a license for a digital product over time. For example, the license rules may comprise at least one allowed copy count corresponding to a maximum number of devices authorized for use with the digital product. With reference to the flow chart shown in Figure 4, there is provided a system 400 that may include: a communication module 410 for receiving a request for authorization to use the digital product from a given device; a processor module 420 in operative communication with the communication module; and a memory module 430 in operative communication with the processor module.

The memory module 430 may include executable code for the processor module to: (a) verify that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device; and (b) in response to the device identity already being on a record, allow the digital product to be used on the given device. The memory module 430 may further include executable code for the processor module to: (c) in response to the device identity not being on the record, set the allowed copy count to a first upper limit for a first time period (e.g., a time period after an initial authorization of the digital product); (d) calculate a device count corresponding to total number of devices already authorized for use with the digital product; and (e) when the calculated device count is less than the first upper limit, allow the digital product to be used on the given device. While the various steps or tasks described herein, e.g., steps (a) through (e) above, sometimes involve having executable

APPLE EXHIBIT 1102, Page 637 of 1048

code stored in the memory module 430, it is noted that the processor module 420 may otherwise be adapted to perform such steps/tasks.

In related aspects, the digital product may comprise software, and/or the given device may comprise a PC or the like. The license data may comprises information that may be used to verify whether the license for the digital product is valid. The record may comprise an authorization database. In further related aspects, the first time period may comprises a defined number of days since the initial authorization. For example, the defined number of days may comprise six days since the initial authorization, and the first upper limit may comprise five authorized devices. In yet further related aspects, the processor module 420 may comprise one or more processor, and may be adapted to, in response to the calculated device count equaling the first upper limit, send a warning regarding the allowed copy count to the given device. The processor module 420 may be adapted to, in response to the calculated device count exceeding the first upper limit, deny the request for authorization.

In further related aspects, the processor module 420 also be adapted to: (a) in response to the device identity not being on the record, after the first time period has expired, set the allowed copy count to a second upper limit for a second time period; (b) recalculate the device count; and/or (c) when the recalculated device count is less than the second upper limit, allow the digital product to be used on the given device. The second time period may comprise a defined number of days since the initial authorization. For example, the defined number of days may comprise thirty-one days since the initial authorization, and the second upper limit may comprise seven authorized devices. The processor module 420 may be adapted to, in response to the calculated device count equaling the second upper limit, send a warning regarding the allowed copy count to the given device. The processor module 420 may be adapted to, in response to the calculated device count exceeding the second upper limit, deny the request for authorization.

In yet further related aspects, the processor module 420 also be adapted to: (a) in response to the device identity not being on the record, after the second time period has expired, set the allowed copy count to a third upper limit; (b) recalculate the device count; and (c) when the recalculated device count is less than the third upper limit,

APPLE EXHIBIT 1102, Page 638 of 1048

allow the digital product to be used on the given device. The third upper limit comprises eleven authorized devices. The processor module 420 may be adapted to, in response to the calculated device count equaling the third upper limit, send a warning regarding the allowed copy count to the given device. The processor module 420 may be adapted to, in response to the calculated device count exceeding the third upper limit, deny the request for authorization.

It is noted that the system 400 may optionally include: a means 450 for verifying that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device; a means 460 for, in response to the device identity already being on a record, allowing the digital product to be used on the given device; a means 470 for, in response to the device identity not being on the record, setting the allowed copy count to a first upper limit for a first time period (e.g. a time period after an initial authorization of the digital product); a means 480 for calculating a device count corresponding to total number of devices already authorized for use with the digital product; and/or a means 490 for, when the calculated device count is less than the first upper limit, allowing the digital product to be used on the given device.

It is also noted that the system 400 may optionally include: a means for, in response to the device identity not being on the record, after the first time period has expired, setting the allowed copy count to a second upper limit for a second time period; a means for recalculating the device count; and/or a means for, when the recalculated device count is less than the second upper limit, allowing the digital product to be used on the given device. It is further noted that the system 400 may optionally include: a means for, in response to the device identity not being on the record, after the second time period has expired, setting the allowed copy count to a third upper limit; a means for recalculating the device count; and/or a means for, when the recalculated device count is less than the third upper limit, allowing the digital product to be used on the given device. The at least one processor of processor module 420, in such case, may be in operative communication with the means 450, 460, 470, 480, and 490 via a bus 440 or similar communication coupling. The processor module 420 may effect initiation and scheduling of the processes or functions performed by the means 450, 460, 470, 480, and 490, and any components thereof.

APPLE EXHIBIT 1102, Page 639 of 1048

In still further related aspects, the device identity may comprise unique device identifying information, wherein the unique device identifying information may comprise at least one user-configurable parameter and/or at least one non-user-configurable parameter of the given device. The device identity may be generated by utilizing at least one irreversible transformation of the at least one user-configurable and the at least one non-user-configurable parameters of the given device. The device identity may be generated by utilizing a cryptographic hash function on the at least one user-configurable and the at least one non-user-configurable parameters of the given device.

It is noted that generating the device identity may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties as well as a variety of additional parameters such as uniquely generated hashes and time sensitive values. Physical device parameters available for sampling may include, for example, unique manufacturer characteristics, carbon and silicone degradation and small device failures.

The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as heat and additional stresses placed on a chip during the sampling process allows for each and every benchmark to reproduce the expected values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms.

In addition to the chip benchmarking and degradation measurements, the process for generating a device identity may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

APPLE EXHIBIT 1102, Page 640 of 1048

Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further extended by the use of complex encryption algorithms to convolute the device identity values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

The device identity may be generated by utilizing machine or device parameters associated with one or more of the following: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: memory model; memory slots; memory total; and memory details.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: video model; video details; display model; display details; audio model; and audio details.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: network model; network address; Bluetooth address; Blackbox model (including IDE and SCSI); Blackbox serial; Blackbox details; Blackbox damage map; Blackbox volume name; NetStore details; and NetStore volume name.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

APPLE EXHIBIT 1102, Page 641 of 1048

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD.

The device identity may also be generated by utilizing machine parameters associated with one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For

APPLE EXHIBIT 1102, Page 642 of 1048

example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

It is understood that the specific order or hierarchy of steps in the processes disclosed herein in an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 643 of 1048

WHAT IS CLAIMED IS:

1. A system (400) for adjusting a license for a digital product over time, the license comprising at least one allowed copy count corresponding to a maximum number of devices authorized for use with the digital product, comprising:

a communication module (410) for receiving a request for authorization to use the digital product from a given device;

a processor module (420) in operative communication with the communication module;

a memory module (430) in operative communication with the processor module and comprising executable code for the processor module to:

verify that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device (450);

in response to the device identity already being on a record, allow the digital product to be used on the given device (460);

in response to the device identity not being on the record, set the allowed copy count to a first upper limit for a first time period (470);

calculate a device count corresponding to total number of devices already authorized for use with the digital product (480); and

when the calculated device count is less than the first upper limit, allow the digital product to be used on the given device (490).

2. The system of Claim 1, wherein the digital product comprises software.

3. The system of Claim 1, wherein the license data comprises information that may be used to verify whether the license for the digital product is valid.

4. The system of Claim 1, wherein the record comprises an authorization database.

5. The system of Claim 1, wherein the first time period comprises a defined number of days after an initial authorization of the digital product.

6. The system of Claim 5, wherein the defined number of days comprises six days since the initial authorization, and wherein the first upper limit comprises five authorized devices.

7. The system of Claim 1, wherein the processor module is adapted to, in response to the calculated device count equaling the first upper limit, send a warning regarding the allowed copy count to the given device.

8. The system of Claim 1, wherein the processor module is adapted to, in response to the calculated device count exceeding the first upper limit, deny the request for authorization.

9. The system of Claim 1, wherein the processor module is adapted to:

in response to the device identity not being on the record, after the first time period has expired, set the allowed copy count to a second upper limit for a second time period (370);

recalculate the device count (372); and

when the recalculated device count is less than the second upper limit, allow the digital product to be used on the given device (374).

10. The system of Claim 9, wherein the second time period comprises a defined number of days since the initial authorization.

11. The system of Claim 10, wherein the defined number of days comprises thirty-one days since the initial authorization, and wherein the second upper limit comprises seven authorized devices.

12. The system of Claim 9, wherein the processor module is adapted to, in response to the calculated device count equaling the second upper limit, send a warning regarding the allowed copy count to the given device.

APPLE EXHIBIT 1102, Page 645 of 1048

13. The system of Claim 9, wherein the processor module is adapted to, in response to the calculated device count exceeding the second upper limit, deny the request for authorization.

14. The system of Claim 9, wherein the processor module is adapted to:

in response to the device identity not being on the record, after the second time period has expired, set the allowed copy count to a third upper limit (380);

recalculate the device count (382); and

when the recalculated device count is less than the third upper limit, allow the digital product to be used on the given device (384).

15. The system of Claim 14, wherein the third upper limit comprises eleven authorized devices.

16. The system of Claim 14, wherein the processor module is adapted to, in response to the calculated device count equaling the third upper limit, send a warning regarding the allowed copy count to the given device.

17. The system of Claim 14, wherein the processor module is adapted to, in response to the calculated device count exceeding the third upper limit, deny the request for authorization.

18. The system of Claim 1, wherein the device identity comprises unique device identifying information.

19. The system of Claim 18, wherein the unique device identifying information comprises at least one user-configurable parameter and at least one non-user-configurable parameter of the given device.

20. The system of Claim 18, wherein the device identity is generated by utilizing at least one irreversible transformation of the at least one user-configurable and the at least one non-user-configurable parameters of the given device.

APPLE EXHIBIT 1102, Page 646 of 1048

21. The system of Claim 18, wherein the device identity is generated by utilizing a cryptographic hash function on the at least one user-configurable and the at least one non-user-configurable parameters of the given device.

22. A method (300) for adjusting a license for a digital product over time, the license comprising at least one allowed copy count corresponding to a maximum number of devices authorized for use with the digital product, comprising:

receiving a request for authorization to use the digital product on a given device (310);

verifying that a license data associated with the digital product is valid based at least in part on a device identity associated with the given device (320);

in response to the device identity already being on a record, allowing the digital product to be used on the given device (330);

in response to the device identity not being on the record, setting the allowed copy count to a first upper limit for a first time period (340);

calculating a device count corresponding to total number of devices already authorized for use with the digital product (350); and

when the calculated device count is less than the first upper limit, allowing the digital product to be used on the given device (360).

23. The method of Claim 22, further comprising:

in response to the device identity not being on the record, after the first time period has expired, setting the allowed copy count to a second upper limit for a second time period (370);

recalculating the device count (372); and

when the recalculated device count is less than the second upper limit, allowing the digital product to be used on the given device (374).

APPLE EXHIBIT 1102, Page 647 of 1048

24. The method of Claim 23, further comprising:

in response to the device identity not being on the record, after the second time period has expired, setting the allowed copy count to a third upper limit (380);

recalculating the device count (382); and

when the recalculated device count is less than the third upper limit, allowing the digital product to be used on the given device. (384).

25. A computer program product, comprising:

a computer-readable medium comprising:

code for causing a computer to receive a request for authorization to use the digital product (310);

code for causing a computer to verify that a license data associated with the digital product is valid based at least in part on a device identity associated with the computer (320, 450);

code for causing a computer to, in response to the device identity already being on a record, allow the digital product to be used on the computer (330, 460);

code for causing a computer to, in response to the device identity not being on the record, set the allowed copy count to a first upper limit for a first time period after an initial authorization of the digital product (340, 470);

code for causing a computer to calculate a device count corresponding to total number of devices already authorized for use with the digital product (350, 480); and

code for causing a computer to, when the calculated device count is less than the first upper limit, allowing the digital product to be used on the computer (360, 490).

APPLE EXHIBIT 1102, Page 648 of 1048

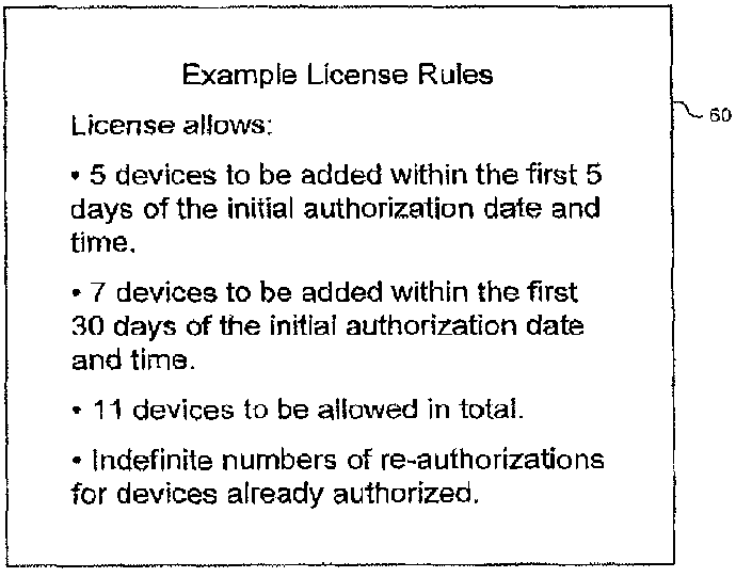


Figure 1

APPLE EXHIBIT 1102, Page 649 of 1048

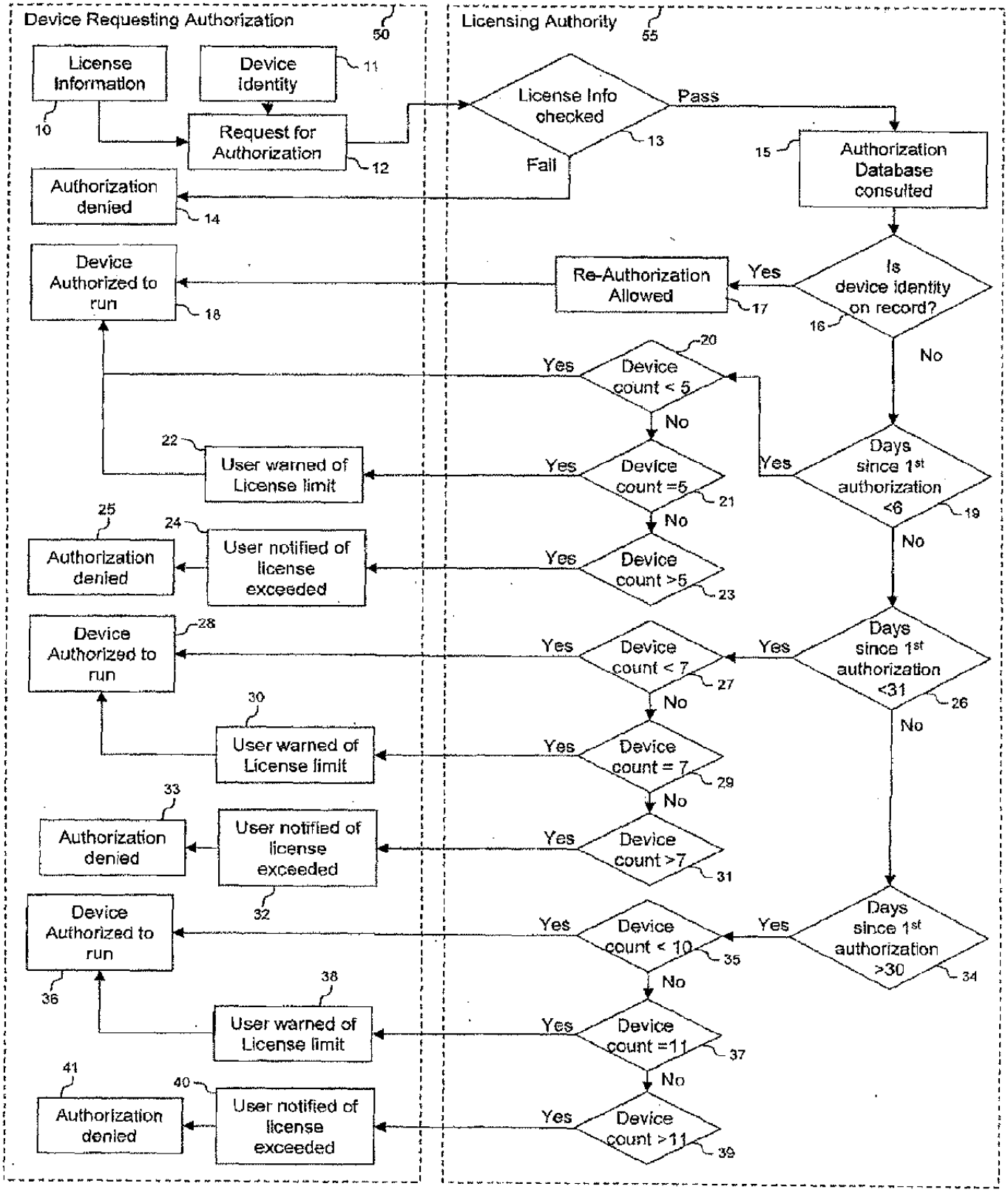


Figure 2

APPLE EXHIBIT 1102, Page 650 of 1048

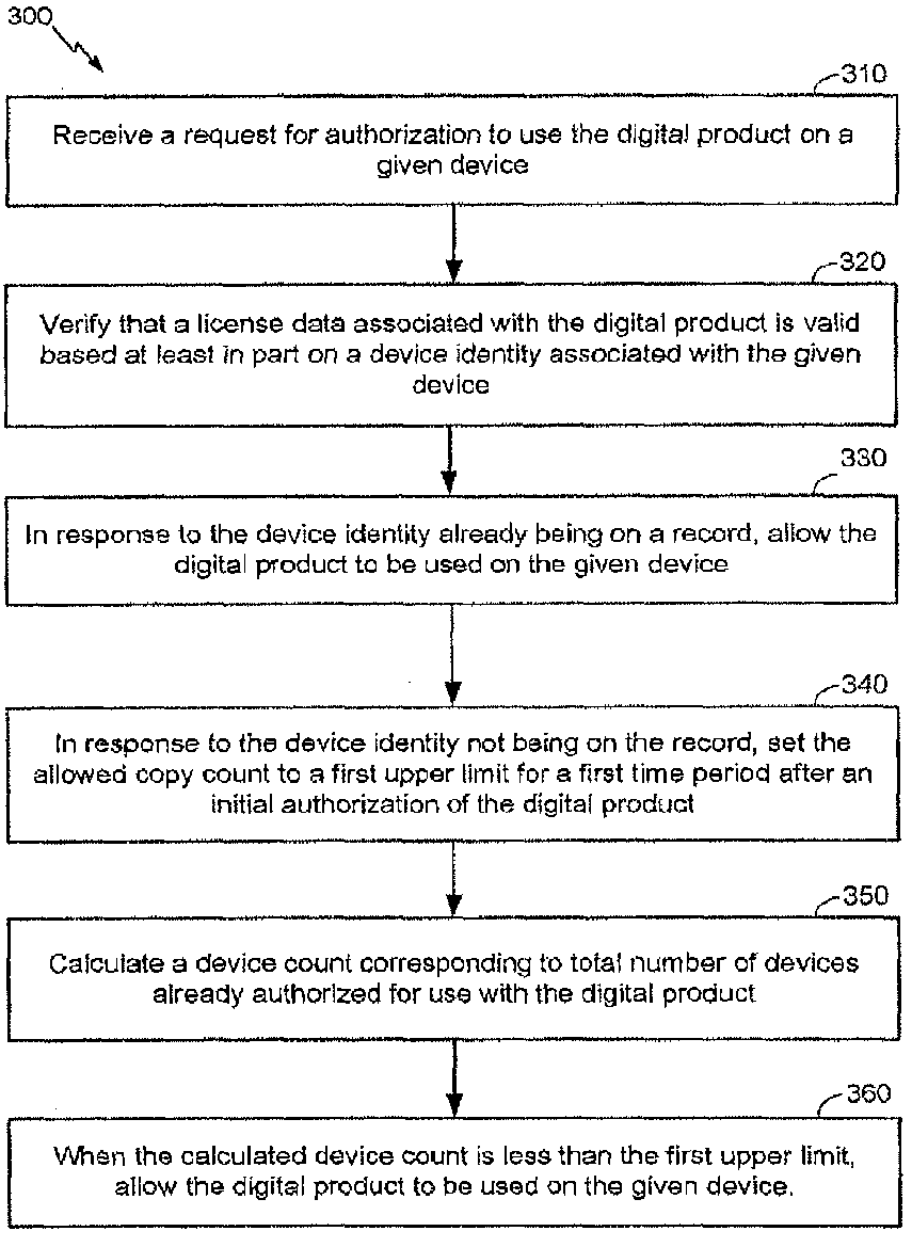


Figure 3A

APPLE EXHIBIT 1102, Page 651 of 1048

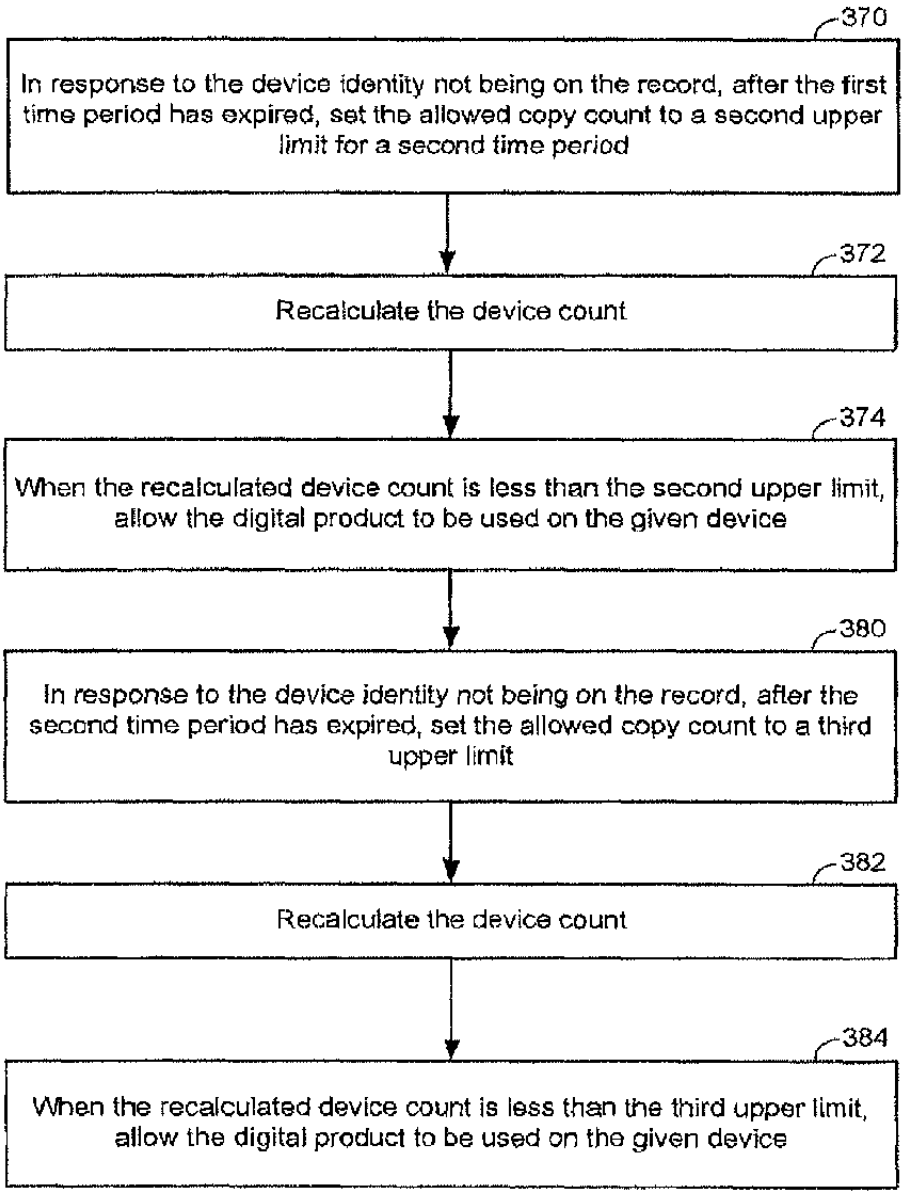


Figure 3B

APPLE EXHIBIT 1102, Page 652 of 1048

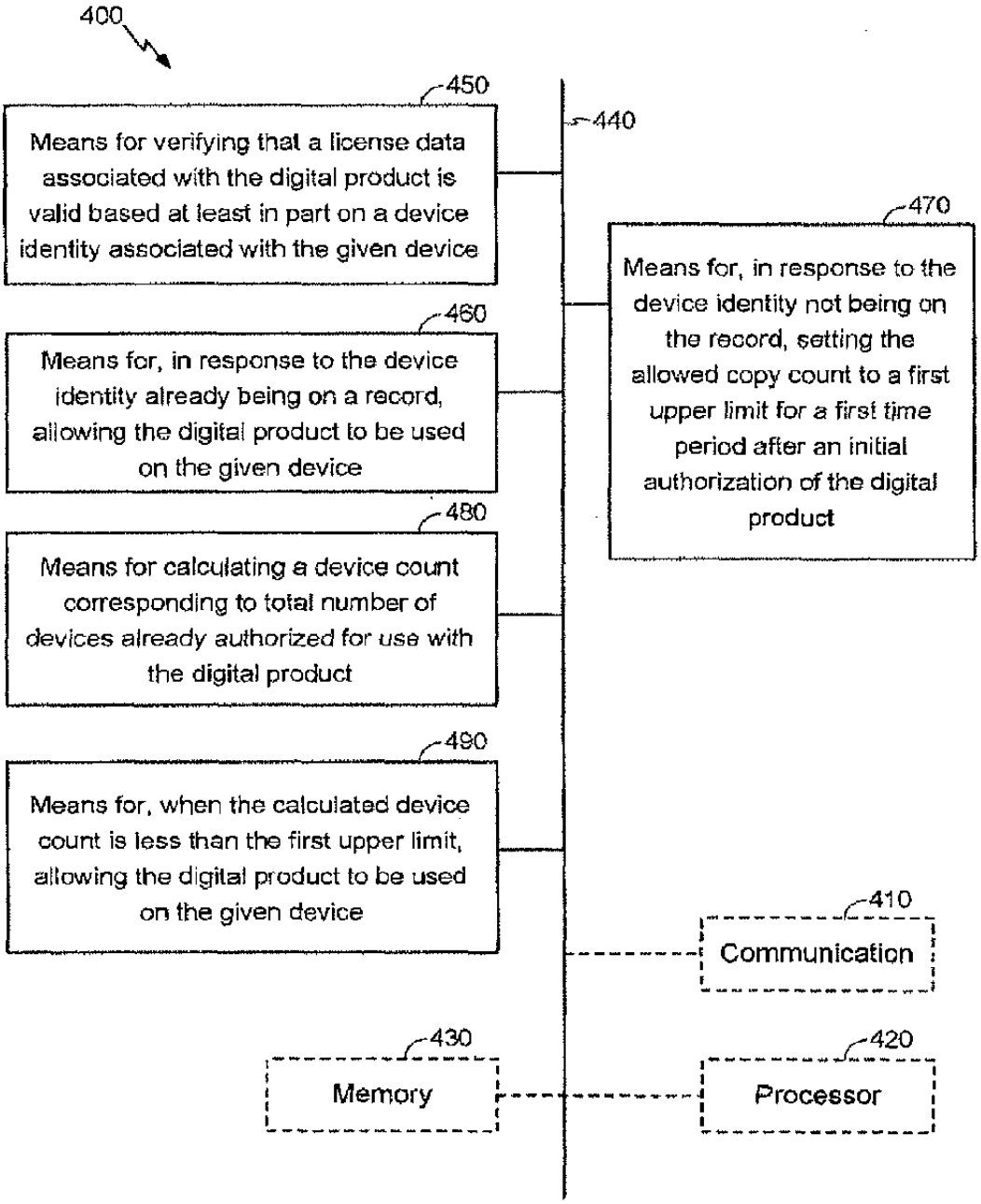


Figure 4

APPLE EXHIBIT 1102, Page 653 of 1048

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2008/083809

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/282511 A1 (TAKANO HARUKO [JP] ET AL) 14 December 2006 (2006-12-14) paragraphs [0051] - [0058]	1-25
Y	US 2004/143746 A1 (LIGETI JEAN-ALFRED [CA] ET AL) 22 July 2004 (2004-07-22) paragraph [0143]	1-25
Y	US 2001/034712 A1 (COLVIN DAVID S [US]) 25 October 2001 (2001-10-25) abstract; figure 4b	1-25
Y	US 6 243 468 B1 (PEARCE DAVID B [US] ET AL) 5 June 2001 (2001-06-05) column 2, lines 15-34	1-25
A	US 2004/024860 A1 (SATO KATSUHIKO [JP] ET AL) 5 February 2004 (2004-02-05) abstract; figure 3	1-25

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
16 April 2009	29/04/2009

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Kerschbaumer, J

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2008/083809

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006282511 A1	14-12-2006	JP 2006352289 A	28-12-2006
US 2004143746 A1	22-07-2004	NONE	
US 2001034712 A1	25-10-2001	NONE	
US 6243468 B1	05-06-2001	US 2001044782 A1	22-11-2001
US 2004024860 A1	05-02-2004	WO 0235362 A1	02-05-2002
		JP 3763393 B2	05-04-2006
		JP 2002132584 A	10-05-2002
		TW 565800 B	11-12-2003

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
18 June 2009 (18.06.2009)

PCT

(10) International Publication Number
WO 2009/076232 A1

(51) International Patent Classification:
H04L 9/08 (2006.01)

(21) International Application Number:
PCT/US2008/085730

(22) International Filing Date:
5 December 2008 (05.12.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/055,129 21 May 2008 (21.05.2008) US
60/992,704 5 December 2007 (05.12.2007) US

(71) Applicant *(for all designated States except US)*: **UNILOC CORPORATION** [US/US]; 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **RICHARDSON, Ric, B.** [AU/US]; C/o Uniloc Corporation, 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US). **ETCHEGOYEN, Craig, S.** [US/US]; C/o Uniloc Corporation, 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US). **HARJANTO, Dono** [US/US]; C/o Uniloc Corporation, 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US). **DAVIS, Bradley, C.** [US/US]; C/o Uniloc Corporation, 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz Llp, P.o. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR DEVICE BOUND PUBLIC KEY INFRASTRUCTURE

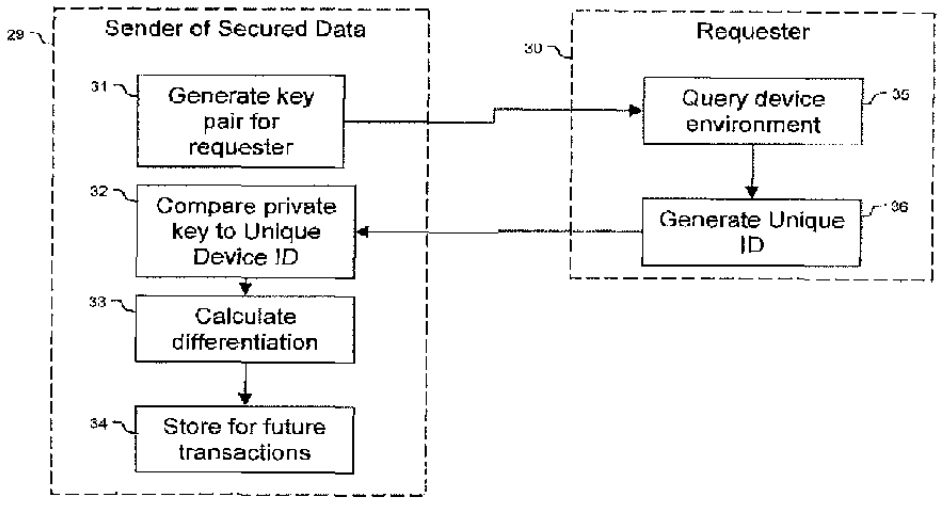


Figure 3

(57) Abstract: Techniques are provided secured communication of data, such as in the context of a public key infrastructure (PKI). In one embodiment, the technique may involve using a private key that is bound to the device requesting the secure data, thereby making it harder for someone to copy, steal or fake. The private key may be generated by adding a filler code to a unique device identifier. The identifier may be based on at least one user-configurable parameter and at least one non-user-configurable parameter of the device.

WO 2009/076232 A1

FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

SYSTEM AND METHOD FOR DEVICE BOUND PUBLIC KEY INFRASTRUCTURE

Background of the Invention

Field of the Invention

[0001] The present application relates generally to managing software use, and more specifically to systems and methods to enable the monitoring and adjusting software usage under a software license.

Description of the Related Art

[0002] Public key infrastructure (PKI) encryption is used to secure data exchanges in digital form. It comprises a unique mathematical property that allows part of the PKI encryption subsystem to be public and part of the subsystem to remain secret, but in the process negates the need for a secret to be shared between two parties wishing to share protected data. This has been a major breakthrough in secure data exchange.

[0003] Figure 1 represents a simple public key infrastructure encryption system as commonly used in the art. It is included for informational purposes. This encryption system comprises two sets of two keys, one each for the sender and recipient of the encrypted data. The key pairs on each side of the data exchange consist of a private key 12, 15 that is kept secret by the user and a public key 13, 14 which is shared with the other party in the exchange of data. There is a unique relationship between the private key 14 and the public key 15 of the sender 11 and the private key 12 and the public key 13 of the receiver 10 that allows the sender 11 of data to use their private key 15 and the recipient's public key 13 to produce an encryption code 17 that is used to encrypt the data. Conversely, the simple public key infrastructure encryption system allows the requester 10 to use the sender's public key 14 and their own private key 12 to produce a decryption code 16 that can in turn be used to decrypt the data after receipt.

[0004] However, a major problem or shortcoming with such existing systems is that the secret part of the PKI subsystem, also called the private key, can be copied, stolen or faked. Accordingly, there is a need for an improved technique for securing communication via PKI encryption that prevents the copying or stealing of private keys, the use of fake private keys, etc.

APPLE EXHIBIT 1102, Page 658 of 1048

Summary of the Invention

[0005] The following presents a simplified summary of one or more embodiments in order to provide a basic understanding of such embodiments. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later.

[0006] The present invention addresses the above-described shortcomings of existing encryption systems by using a private key that is bound to the device requesting the secure data, thereby making it harder for someone to copy, steal or fake. The present invention also adds a layer of authentication to the data exchange in that the unique device ID or identifier can also be used for forensic purposes in proving who has received or sent particular protected data.

[0007] In accordance with one or more aspects of the present invention, there are provided techniques that involve: (a) generating a requester key pair for a data requesting device, the pair comprising a requester private key and a requester public key; (b) receiving a unique device identifier from the device; (c) calculating a difference between the identifier and the requester private key; and (d) storing the difference as a filler code.

[0008] In related aspects, the techniques may also involve: (e) in response to a request for data from the device, identifying the device; (f) encrypting the data using a sender private key and the requester public key; and (g) sending the encrypted data and the filler code to the device. In further related aspects, Step (g) may further involve publishing a sender public key, such that the device is able to (i) compute the requester private key by adding the filler code to the identifier and (ii) use the computed requester private key to decrypt the encrypted data.

[0009] In accordance with one or more aspects of the present invention, there are provided techniques that involve: (a) receiving from a data sender a request for a unique device identifier of a requesting device; (b) in response to the request from the sender, compiling unique identifying information from a computing environment of the device; (c) generating the identifier based at least in part on the compiled information; and (d) providing the generated identifier to the sender.

[0010] In related aspects, the techniques may also involve: (e) receiving encrypted data and a filler code from the sender; (f) computing a requester private key by adding the filler

code to the identifier; and (g) using the computed requester private key to decrypt the encrypted data. In further related aspects, step (b) may further involve compiling at least one user-configurable parameter and at least one non-user-configurable parameter of the device.

[0011] To the accomplishment of the foregoing and related ends, the one or more embodiments comprise the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects of the one or more embodiments. These aspects are indicative, however, of but a few of the various ways in which the principles of various embodiments may be employed and the described embodiments are intended to include all such aspects and their equivalents.

Brief Description of the Drawings

[0012] Figure 1 shows a known public key infrastructure (PKI).

[0013] Figure 2 illustrates an embodiment of a device bound PKI system.

[0014] Figure 3 provides a flow diagram for an exemplary approach to device bound private key generation.

[0015] Figure 4 provides a flow diagram for an exemplary approach to device bound private key use.

[0016] Figures 5A-B show one embodiment of an apparatus for sending data via secured communication in PKI.

[0017] Figure 6A-B show one embodiment of an apparatus for requesting data via secured communication in a PKI.

[0018] Figures 7A-B provide flow diagrams for an exemplary method for sending data via secured communication in a PKI.

[0019] Figures 8A-B provide flow diagrams for an exemplary method for requesting data via secured communication in a PKI.

Detailed Description

[0020] Various embodiments are now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of one or more embodiments. It may be evident, however, that such embodiment(s) can be practiced without these specific details. In other instances,

APPLE EXHIBIT 1102, Page 660 of 1048

well-known structures and devices are shown in block diagram form in order to facilitate describing one or more embodiments.

[0021] In accordance with one or more aspects of the embodiments described herein, there is provided a system and method for sending data via secured communication in a public key infrastructure. Figure 2 shows an example embodiment of how to bind the simple public key infrastructure of figure one to a specific computing device. The encryption system operates implements certain features of a simple public key infrastructure encryption system; however the requester's private key 22 is not stored locally but generated by producing a unique device ID 20 (also referred to herein as a unique device identifier) from the operating environment of the local computing device and combining it with a filler code 21 that together produce a comparable private key 22 that can, in turn, be used for decryption 27 of PKI encrypted data. The unique device ID is usually a string of data resulting from the compiling of unique identifying information from the devices computing environment. Examples of this are hard drive serial number and data storage damage locations in a computing device. Further examples of unique identifying information that be used to generate/compute the unique device ID are provided below.

[0022] Figure 3 shows an exemplary way in which a recipient's private key can be generated by a sender. Initially a key pair may be generated (step 31) using standard PKI key generation known in the art. The two keys may then be set aside for use as a public key and a private key. The sender 29 may then request the requester 30 to send back to them a unique device ID 35 generated from the operating environment of the recipient's device. The sender 29 then compares the unique device ID with the private key 32 to be used by the requester 30 (step 32). The sender 29 may calculate the difference between the unique device ID and the private key 33 (step 33). This calculated difference may be stored as a filler code (step 34).

[0023] Figure 4 illustrates an exemplary manner in which a device bound private key may be used in practice. The requester 37 asks for encrypted data to be sent to them and also publishes their public key 39 to be used as part of the data exchange (step 39). The sender 38 identifies the requester 37 (step 40), and encrypts the data 42 to be sent using their own private key 25 and the requester's public key 23 (step 42). Then (at step 44) the sender 38 sends to the requester 37 the encrypted data, the requester's filler code 21 that has been previously generated, and publishes their own public key 24.

APPLE EXHIBIT 1102, Page 661 of 1048

[0024] The requester then generates or computes unique device ID 20 (step 41), and computes its own private key 22 by adding the filler code 21 to the computed ID (step 43). This computed value is then used to decrypt 27 the data for use (step 45).

[0025] In another embodiment, the above described techniques for generating the device bound private key, as described with reference to Figure 3, and techniques for using the device bound private key, as described with reference to Figure 4, may be applied to or used with public key cryptography (PKC) and authentication, where there is no key pair on both sides of the data exchange. Additionally, device binding may be used to generate public as well as private keys allowing for an additional layer of security and authentication.

[0026] In yet another embodiment, the technique for generating and using keys may involve the generation and subsequent storage for use later of the unique device ID without the need to generate a unique device ID every time data is exchanged.

[0027] In related aspects, the device identity (i.e., the unique device identifier) may comprise and/or be generated from unique device identifying information, wherein the unique device identifying information may comprise at least one user-configurable parameter and/or at least one non-user-configurable parameter of the given device. The device identity may be generated by utilizing at least one irreversible transformation of the at least one user-configurable and the at least one non-user-configurable parameters of the given device. The device identity may be generated by utilizing a cryptographic hash function on the at least one user-configurable and the at least one non-user-configurable parameters of the given device.

[0028] It is noted that generating the device identity may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties as well as a variety of additional parameters such as uniquely generated hashes and time sensitive values. Physical device parameters available for sampling may include, for example, unique manufacturer characteristics, carbon and silicone degradation and small device failures.

[0029] The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as heat and additional stresses placed on a chip during the sampling process allows for each and every benchmark to reproduce the expected values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to

APPLE EXHIBIT 1102, Page 662 of 1048

degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms.

[0030] In addition to the chip benchmarking and degradation measurements, the process for generating a device identity may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

[0031] Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further extended by the use of complex encryption algorithms to convolute the device identity values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

[0032] The device identity may be generated by utilizing machine or device parameters associated with one or more of the following: machine model; machine or hard drive serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

[0033] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

[0034] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: memory model; memory slots; memory total; and memory details.

[0035] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: video model; video details; display model; display details; audio model; and audio details.

[0036] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: network model; network address; Bluetooth address; Blackbox model (including IDE and SCSI); Blackbox serial; Blackbox details; Blackbox damage map; Blackbox volume name; NetStore details; and NetStore volume name.

APPLE EXHIBIT 1102, Page 663 of 1048

[0037] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

[0038] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

[0039] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

[0040] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller.

[0041] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

[0042] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

[0043] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

[0044] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD.

[0045] The device identity may also be generated by utilizing machine parameters associated with one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

[0046] In accordance with one or more aspects of the embodiments described herein, there is provided a device for sending data via secured communication in a public key infrastructure. With reference to the embodiment of Figure 5A, there is provided a device 100 may include a communication module 110 for communicating with a data requesting device; at least one processor 120 in operative communication with the communication

APPLE EXHIBIT 1102, Page 664 of 1048

module 110; and a memory 130 in operative communication with the at least one processor 120.

[0047] The memory 130 may include executable code for the at least one processor 120 to: (a) generate a requester key pair for the device, the pair comprising a requester private key and a requester public key; and (b) in response to receiving a unique device identifier from the device, calculate a difference between the identifier and the requester private key.

[0048] In related aspects, the at least one processor 120 may store the calculated difference as a filler code in the memory. In further related aspects, the at least one processor may be adapted to: (a) in response to a request for the data from the device, identify the device; (b) encrypt the data using a sender private key and the requester public key; and (c) send the encrypted data and the filler code to the device. In yet further related aspects, the at least one processor 120 may publish a sender public key, such that the device is able to (i) compute the requester private key by adding the filler code to the identifier and (ii) use the computed requester private key to decrypt the encrypted data.

[0049] It is noted that device 100 may optionally include: a means 150 for generating a requester key pair for a data requesting device, the pair comprising a requester private key and a requester public key; a means 152 for receiving a unique device identifier from the device; a means 154 for calculating a difference between the identifier and the requester private key; and a means 156 for storing the difference as a filler code.

[0050] With reference to Figure 5B, device 100 may optionally include: a means 158 for in response to a request for data from the device, identifying the device; a means 160 for encrypting the data using a sender private key and the requester public key; and a means 162 for sending the encrypted data and the filler code to the device and publishing a sender public key. The at least one processor 120, in such case, may be in operative communication with the means 150-162 via a bus 170 or similar communication coupling. The at least one processor 120 may effect initiation and scheduling of the processes or functions performed by the means 150-162, and any components thereof.

[0051] In accordance with one or more aspects of the embodiments described herein, there is provided a device for requesting data via secured communication in a public key infrastructure. With reference to the embodiment of Figure 6A, there is provided a device 200 may include a communication module 210 for communicating with a data sender; at least one processor 220 in operative communication with the communication module 210; and a memory 230 in operative communication with the at least one processor 220.

APPLE EXHIBIT 1102, Page 665 of 1048

[0052] The memory 230 may include executable code for the at least one processor 220 to: (a) receive from a data sender a request for a unique device identifier of a requesting device; (b) in response to the request from the sender, compile unique identifying information from a computing environment of the device 200; (c) generate the identifier based at least in part on the compiled information; and (d) instruct the communication module 210 to transmit the generated identifier to the sender.

[0053] In related aspects, the at least one processor 220 may be adapted to: (a) in response to receiving encrypted data and a filler code from the sender, compute a requester private key by adding the filler code to the identifier; and (b) use the computed requester private key to decrypt the encrypted data.

[0054] In further related aspects, the at least one processor 220 may compile the unique identifying information by compiling at least one user-configurable parameter and at least one non-user-configurable parameter of the device 200. The at least one non-user-configurable parameter may include at least one of hard drive serial number, CPU ID, CPU model, CPU manufacturer, and CPU voltage for the device 200. The at least one non-user-configurable parameter may be based on a carbon degradation characteristic of a computer chip of the device 200. The at least one non-user-configurable parameter may be based on a silicone degradation characteristic of a computer chip of the device 200. The at least one user-configurable may include at least one of hard disk volume name, user name, device name, user password, and hard disk initialization date for the device 200.

[0055] It is noted that device 200 may optionally include: a means 250 for receiving from a data sender a request for a unique device identifier of a requesting device; a means 252 for in response to the request from the sender, compiling unique identifying information from a computing environment of the device; a means 254 for generating the identifier based at least in part on the compiled information; and a means 256 for providing the generated identifier to the sender.

[0056] With reference to Figure 6B, device 200 may optionally include: a means 258 for receiving encrypted data and a filler code from the sender; a means 260 for computing a requester private key by adding the filler code to the identifier; and a means 262 for using the computed requester private key to decrypt the encrypted data. The at least one processor 220, in such case, may be in operative communication with the means 250-262 via a bus 270 or similar communication coupling. The at least one processor 220 may effect initiation and

APPLE EXHIBIT 1102, Page 666 of 1048

scheduling of the processes or functions performed by the means 250-262, and any components thereof.

[0057] In accordance with one or more aspects of the embodiments described herein, there is provided a method for sending data via secured communication in a public key infrastructure. With reference to the embodiment of Figure 7A, there is shown a flow chart for a method 300 that involves: generating a requester key pair for a data requesting device, the pair comprising a requester private key and a requester public key (step 310); receiving a unique device identifier from the device (step 320); calculating a difference between the identifier and the requester private key (step 330); and storing the difference as a filler code (340).

[0058] In related aspects, with reference to Figure 7B, the method 300 may further involve: in response to a request for data from the device, identifying the device (step 350); encrypting the data using a sender private key and the requester public key (step 360); and sending the encrypted data and the filler code to the device (370). In further related aspects, step 370 may further involve publishing a sender public key, such that the device is able to (i) compute the requester private key by adding the filler code to the identifier and (ii) use the computed requester private key to decrypt the encrypted data.

[0059] In accordance with one or more aspects of the embodiments described herein, there is provided a method for requesting data via secured communication in a public key infrastructure. With reference to the embodiment of Figure 8A there is provided a method 400 that involves: receiving from a data sender a request for a unique device identifier of a requesting device (step 410); in response to the request from the sender, compiling unique identifying information from a computing environment of the device (420); generating the identifier based at least in part on the compiled information (430); and providing the generated identifier to the sender (440).

[0060] In related aspects, with reference to Figure 8B, the method 400 may further involve: receiving encrypted data and a filler code from the sender (450); computing a requester private key by adding the filler code to the identifier (460); and using the computed requester private key to decrypt the encrypted data (470). In further related aspects, step 420 may involve compiling at least one user-configurable parameter and at least one non-user-configurable parameter of the device.

[0061] While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the

APPLE EXHIBIT 1102, Page 667 of 1048

invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

[0062] As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0063] It is understood that the specific order or hierarchy of steps in the processes disclosed herein in an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure. The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

[0064] Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described

APPLE EXHIBIT 1102, Page 668 of 1048

functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 669 of 1048

WHAT IS CLAIMED IS:

1. A device for sending data via secured communication in a public key infrastructure, comprising:

a communication module for communicating with a data requesting device;

at least one processor in operative communication with the communication module; and

a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to:

generate a requester key pair for the device, the pair comprising a requester private key and a requester public key; and

in response to receiving a unique device identifier from the device, calculate a difference between the identifier and the requester private key.

2. The device of Claim 1, wherein the at least one processor stores the calculated difference as a filler code in the memory.

3. The device of Claim 2, wherein the at least one processor is adapted to:

in response to a request for the data from the device, identify the device;

encrypt the data using a sender private key and the requester public key; and

send the encrypted data and the filler code to the device.

4. The device of Claim 3, wherein the at least one processor publishes a sender public key, such that the device is able to (i) compute the requester private key by adding the filler code to the identifier and (ii) use the computed requester private key to decrypt the encrypted data.

APPLE EXHIBIT 1102, Page 670 of 1048

5. A device for requesting data via secured communication in a public key infrastructure, comprising:

a communication module for communicating with a data sender;

at least one processor in operative communication with the communication module; and

a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to:

receive a request for a unique device identifier of a requesting device from the sender;

in response to the request from the sender, compile unique identifying information from a computing environment of the device;

generate the identifier based at least in part on the compiled information; and

instruct the communication module to transmit the generated identifier to the sender.

6. The device of Claim 5, wherein the at least one processor is adapted to:

in response to receiving encrypted data and a filler code from the sender, compute a requester private key by adding the filler code to the identifier; and

use the computed requester private key to decrypt the encrypted data.

7. The device of Claim 5, wherein the at least one processor compiles the unique identifying information by compiling at least one user-configurable parameter and at least one non-user-configurable parameter of the device.

8. The device of Claim 7, wherein the at least one non-user-configurable parameter comprises at least one of hard drive serial number, CPU ID, CPU model, CPU manufacturer, and CPU voltage for the device.

9. The device of Claim 7, wherein the at least one non-user-configurable parameter is based on a carbon degradation characteristic of a computer chip of the device.

APPLE EXHIBIT 1102, Page 671 of 1048

10. The device of Claim 7, wherein the at least one non-user-configurable parameter is based on a silicone degradation characteristic of a computer chip of the device.

11. The device of Claim 7, wherein the at least one user-configurable parameter comprises at least one of hard disk volume name, user name, device name, user password, and hard disk initialization date for the device.

12. A method for secured communication in a public key infrastructure, comprising:

generating a requester key pair for a data requesting device, the pair comprising a requester private key and a requester public key;

receiving a unique device identifier from the device;

calculating a difference between the identifier and the requester private key; and

storing the difference as a filler code.

13. The method of Claim 12, further comprising:

in response to a request for data from the device, identifying the device;

encrypting the data using a sender private key and the requester public key; and

sending the encrypted data and the filler code to the device.

14. The method of Claim 12, further comprising publishing a sender public key, such that the device is able to (i) compute the requester private key by adding the filler code to the identifier and (ii) use the computed requester private key to decrypt the encrypted data.

15. A method for secured communication in a public key infrastructure, comprising:

receiving from a data sender a request for a unique device identifier of a requesting device;

in response to the request from the sender, compiling unique identifying information from a computing environment of the device;

generating the identifier based at least in part on the compiled information; and

providing the generated identifier to the sender.

APPLE EXHIBIT 1102, Page 672 of 1048

16. The method of Claim 15, further comprising:

receiving encrypted data and a filler code from the sender;

computing a requester private key by adding the filler code to the identifier; and

using the computed requester private key to decrypt the encrypted data.

17. The method of Claim 15, wherein compiling the unique identifying information comprises compiling at least one user-configurable parameter and at least one non-user-configurable parameter of the device.

18. The method of Claim 17, wherein compiling the at least one non-user-configurable parameter comprises compiling at least one of hard drive serial number, CPU ID, CPU model, CPU manufacturer, and CPU voltage for the device.

19. The method of Claim 17, wherein compiling the at least one non-user-configurable parameter comprises compiling a carbon degradation characteristic of a computer chip of the device.

20. The method of Claim 17, wherein compiling the at least one non-user-configurable parameter comprises compiling a silicone degradation characteristic of a computer chip of the device.

21. The method of Claim 17, wherein compiling the at least one user-configurable parameter comprises compiling at least one of hard disk volume name, user name, device name, user password, and hard disk initialization date for the device.

APPLE EXHIBIT 1102, Page 673 of 1048

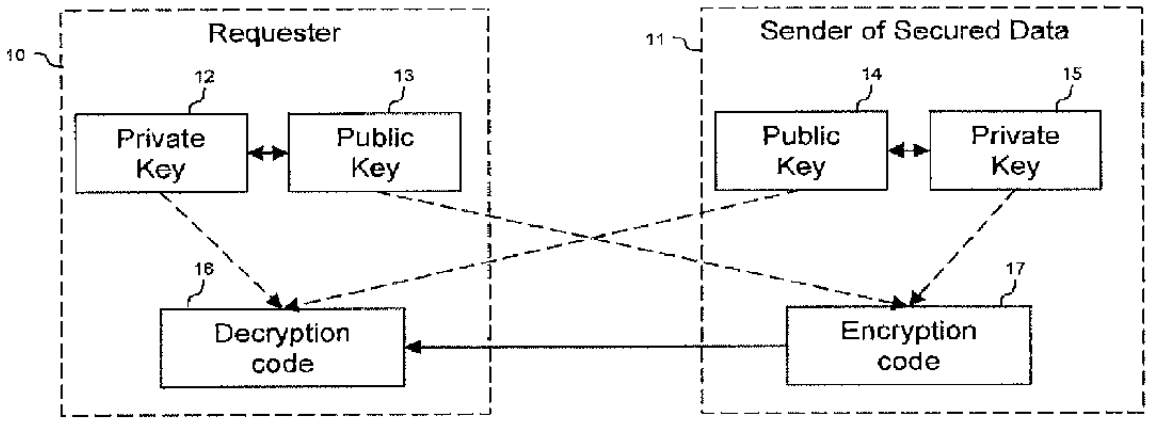


Figure 1 (Prior Art)

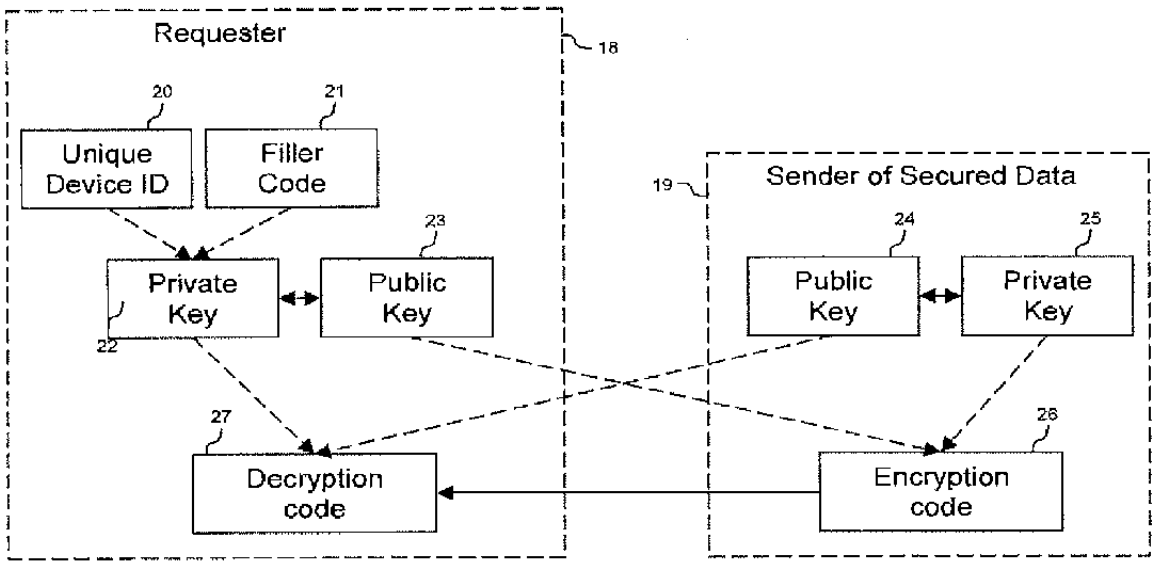


Figure 2

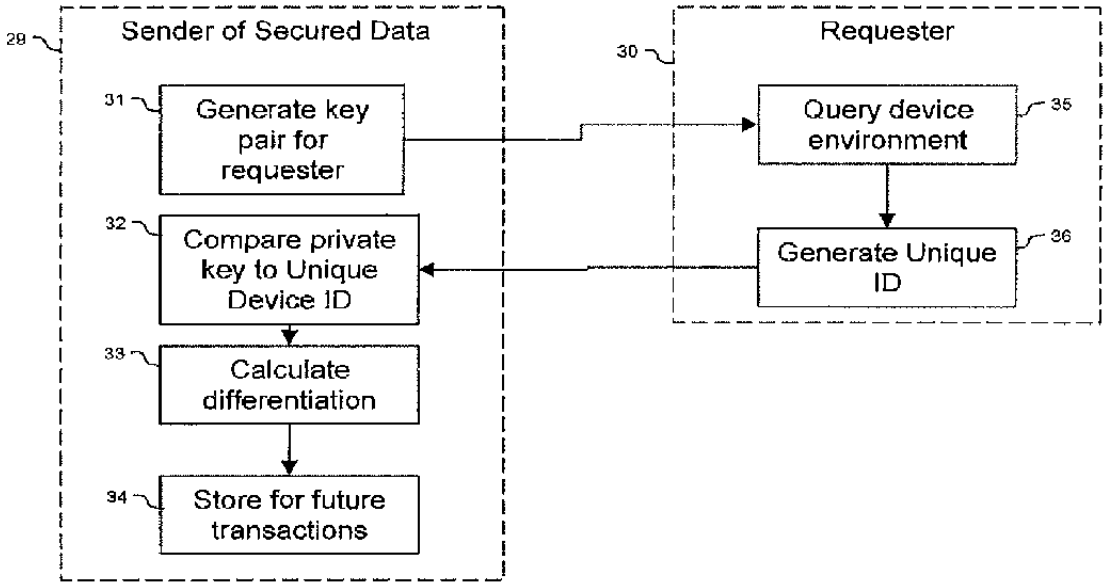

Sender of Secured Data
29
31
Generate key pair for requester
32
Compare private key to Unique Device ID
33
Calculate differentiation
34
Store for future transactions
Requester
30
Query device environment
35
Generate Unique ID
36

Figure 3

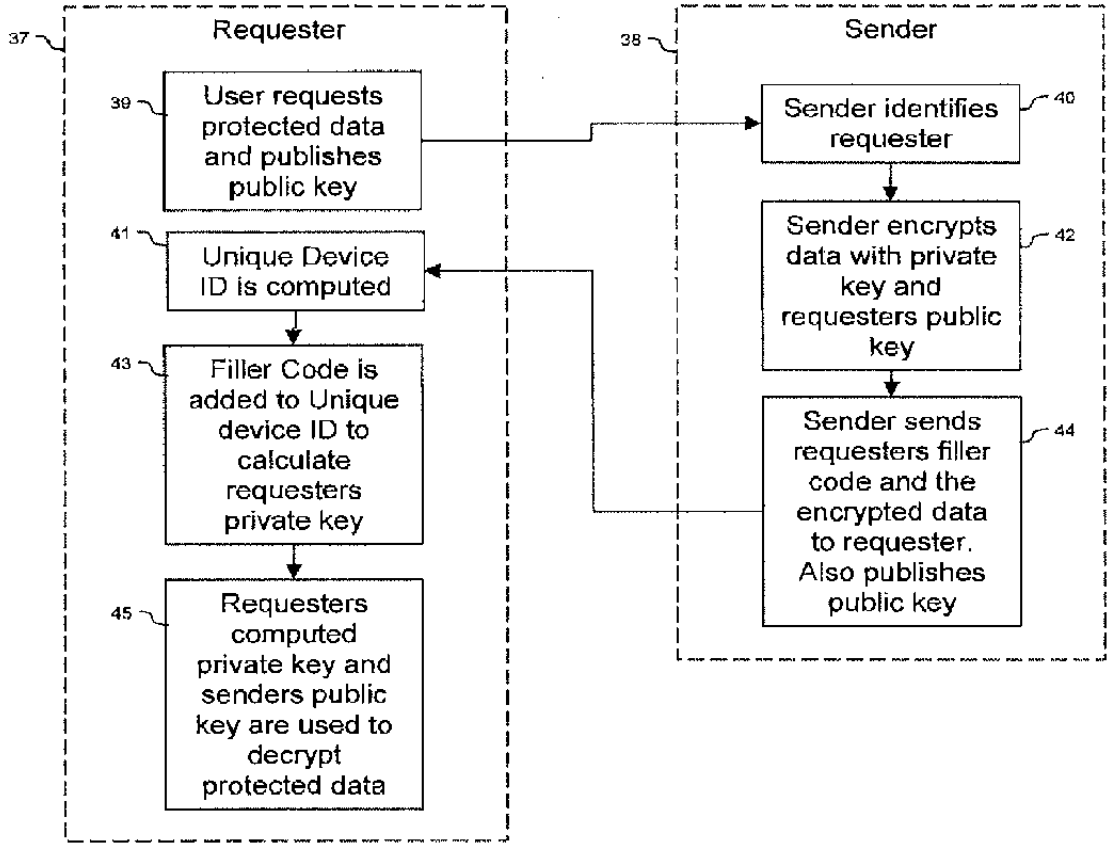

Requester
37
39
User requests protected data and publishes public key
41
Unique Device ID is computed
43
Filler Code is added to Unique device ID to calculate requesters private key
45
Requesters computed private key and senders public key are used to decrypt protected data
Sender
38
Sender identifies requester
40
Sender encrypts data with private key and requesters public key
42
Sender sends requesters filler code and the encrypted data to requester. Also publishes public key
44

Figure 4

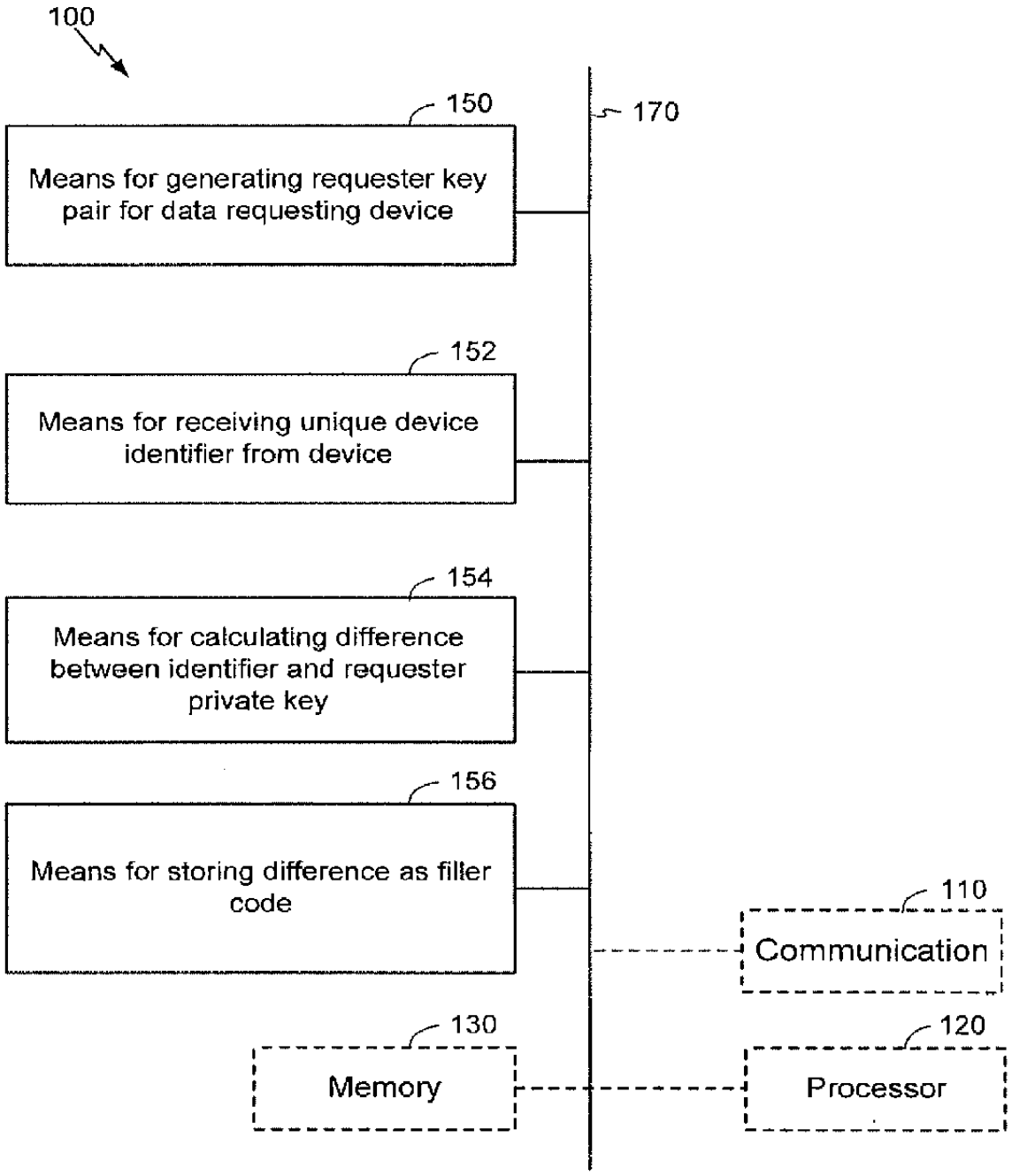

100
150
Means for generating requester key pair for data requesting device
170
152
Means for receiving unique device identifier from device
154
Means for calculating difference between identifier and requester private key
156
Means for storing difference as filler code
110
Communication
130
Memory
120
Processor

Figure 5A

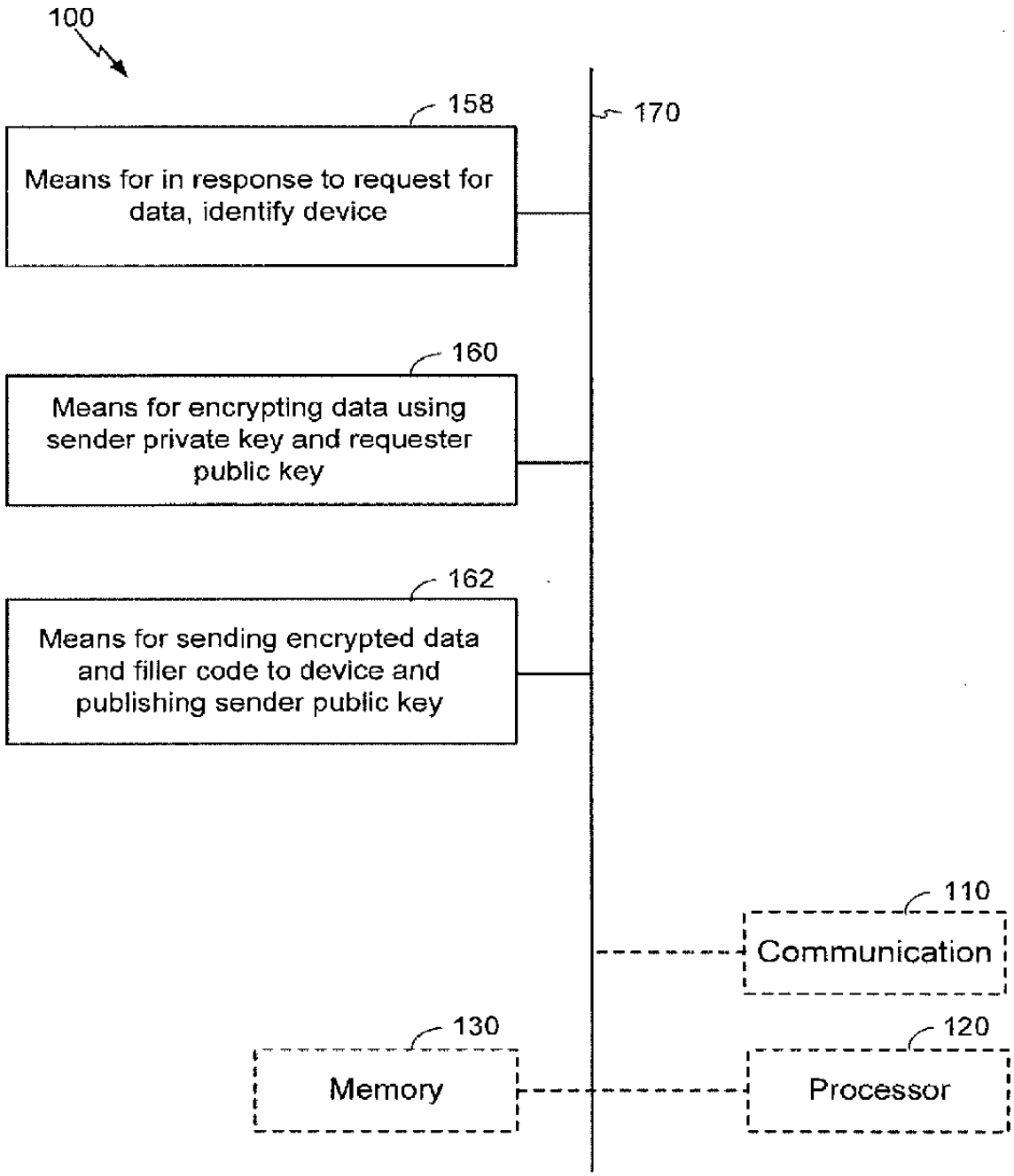

100
158
170
Means for in response to request for data, identify device
160
Means for encrypting data using sender private key and requester public key
162
Means for sending encrypted data and filler code to device and publishing sender public key
110
Communication
130
Memory
120
Processor

Figure 5B

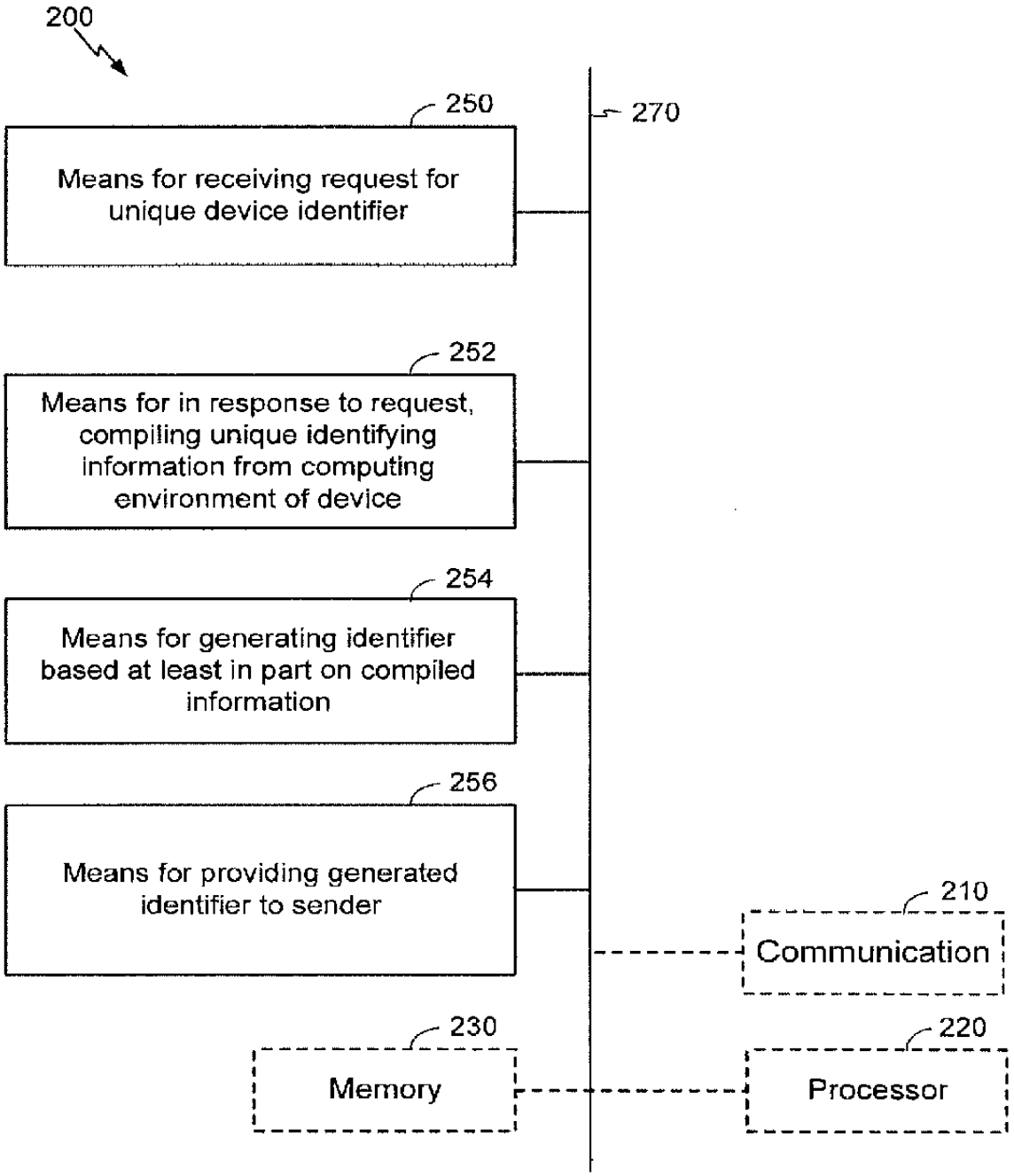

200
250
Means for receiving request for unique device identifier
270
252
Means for in response to request, compiling unique identifying information from computing environment of device
254
Means for generating identifier based at least in part on compiled information
256
Means for providing generated identifier to sender
210
Communication
230
Memory
220
Processor

Figure 6A

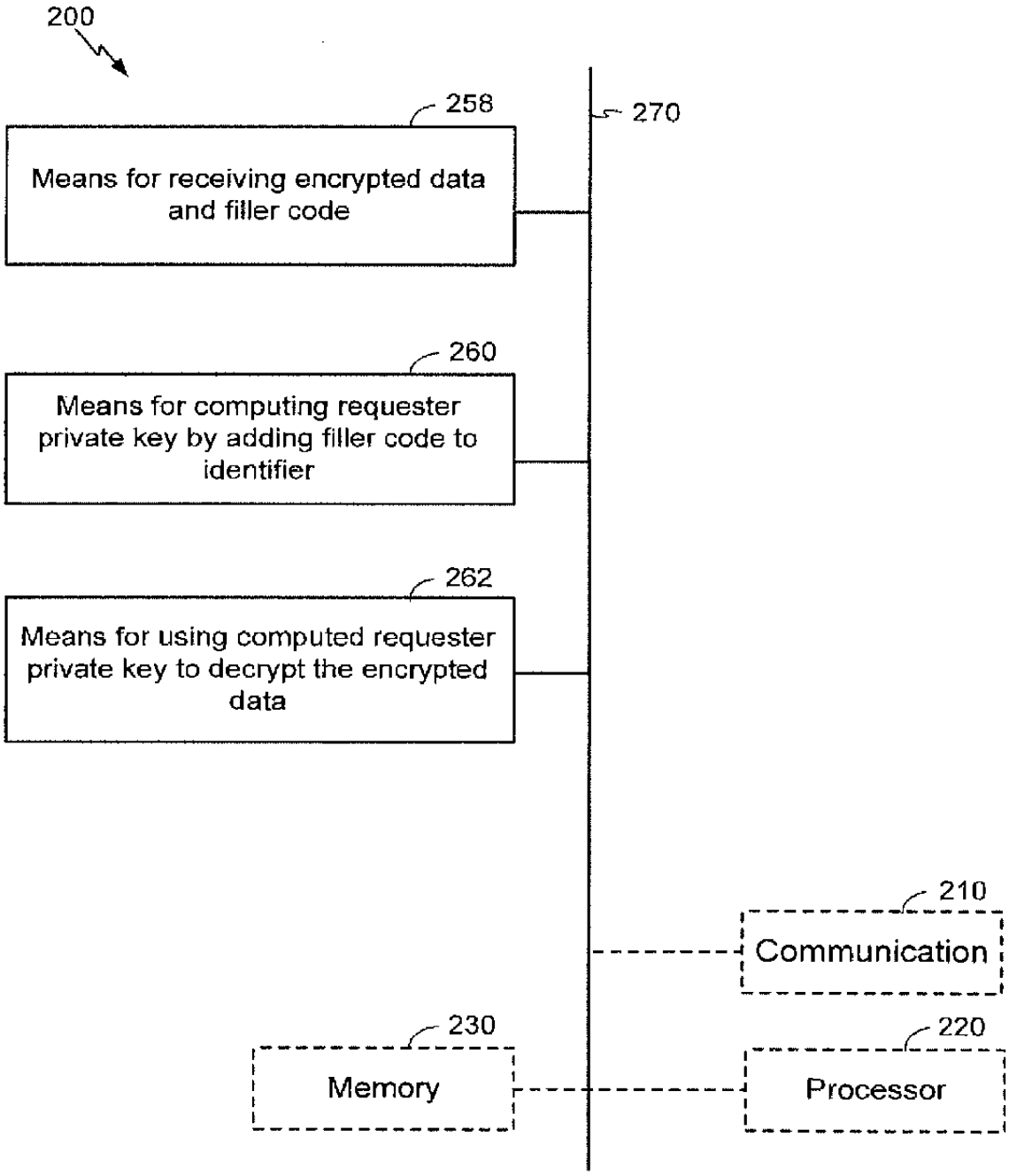

200
258
270
Means for receiving encrypted data and filler code
260
Means for computing requester private key by adding filler code to identifier
262
Means for using computed requester private key to decrypt the encrypted data
210
Communication
230
Memory
220
Processor

Figure 6B

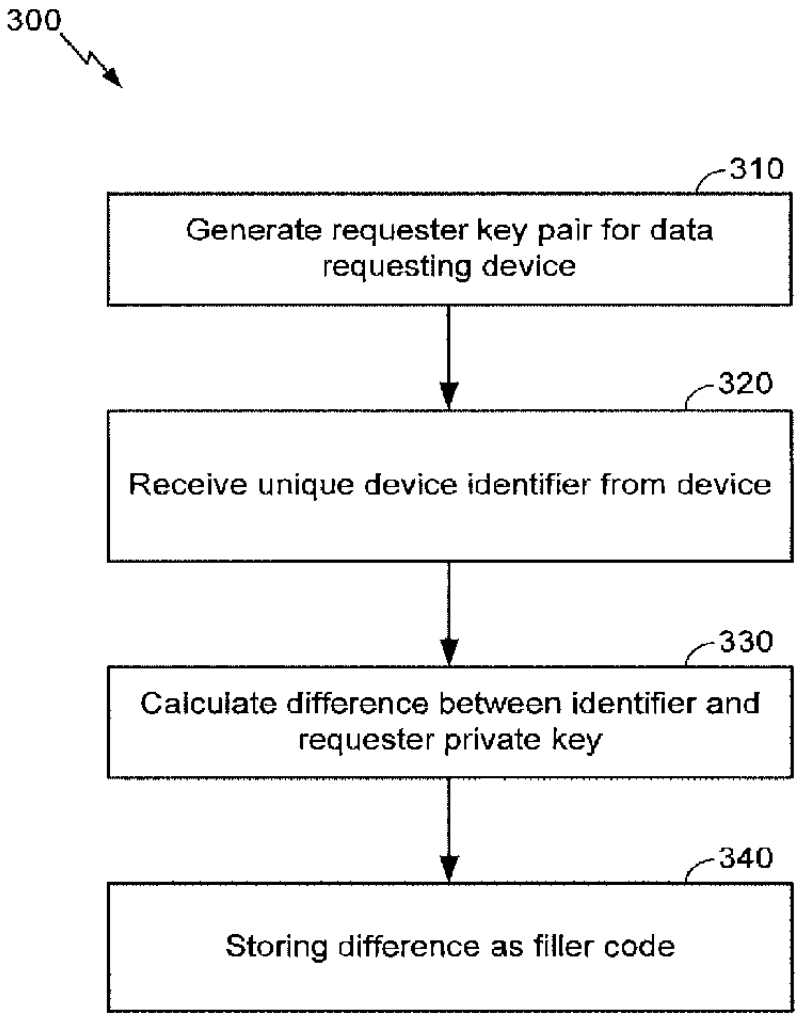

300
310
Generate requester key pair for data requesting device
320
Receive unique device identifier from device
330
Calculate difference between identifier and requester private key
340
Storing difference as filler code

Figure 7A

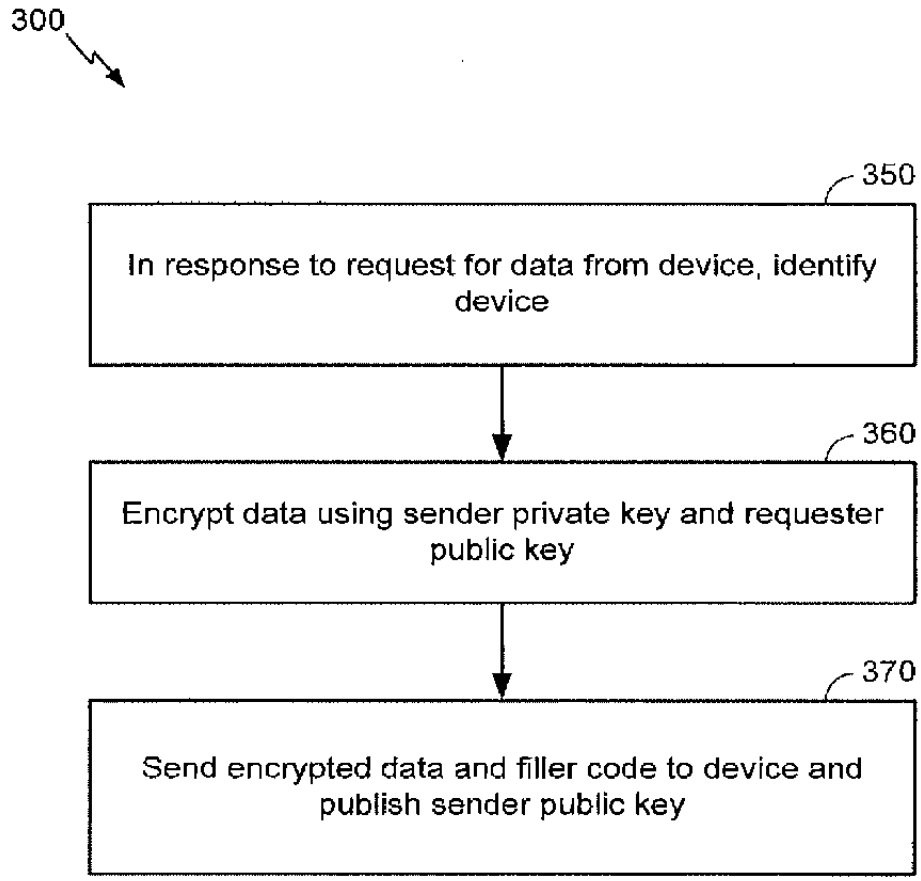

300
350
In response to request for data from device, identify device
360
Encrypt data using sender private key and requester public key
370
Send encrypted data and filler code to device and publish sender public key

Figure 7B

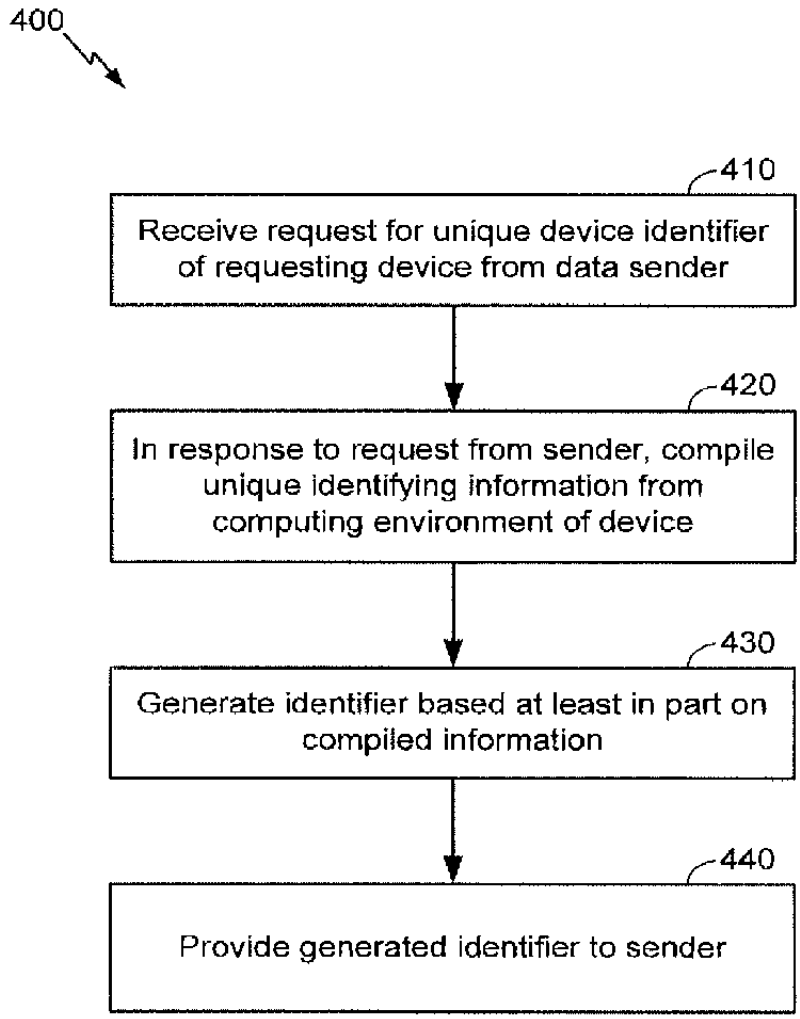

400
410
Receive request for unique device identifier of requesting device from data sender
420
In response to request from sender, compile unique identifying information from computing environment of device
430
Generate identifier based at least in part on compiled information
440
Provide generated identifier to sender

Figure 8A

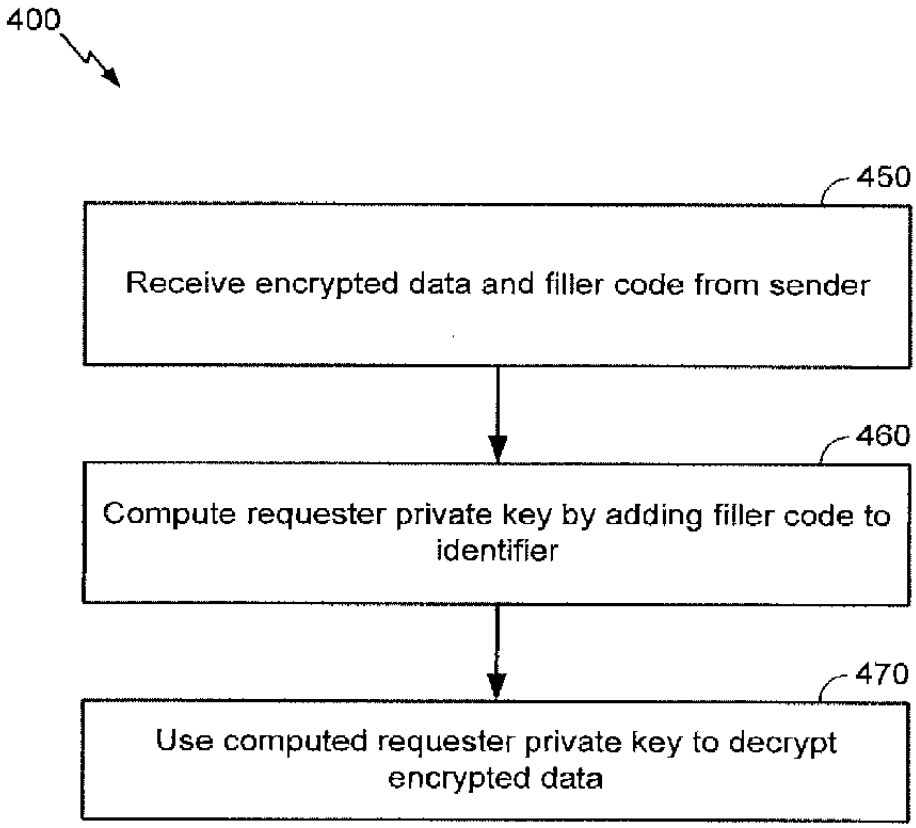

400
450
Receive encrypted data and filler code from sender
460
Compute requester private key by adding filler code to identifier
470
Use computed requester private key to decrypt encrypted data

Figure 8B

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2008/085730

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L9/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X A	US 2006/095454 A1 (SHANKAR NARENDAR [US] ET AL) 4 May 2006 (2006-05-04) paragraph [0030] - paragraph [0042] claim 2	1-4, 12-14 5-11, 15-21
X A	WO 95/35533 A (MEGALODE CORP [CA]) 28 December 1995 (1995-12-28) page 11, line 17 - page 13, line 12 page 15, line 32 - page 17, line 7 page 19, line 10 - page 30, line 15	5-11, 15-21 1-4, 12-14
X A	US 6 158 005 A (BHARATHAN VIPIN [US] ET AL) 5 December 2000 (2000-12-05) column 3, line 50 - column 4, line 20	5,15 1-4, 6-14, 16-21

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
11 May 2009	18/05/2009

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Fax: (+31–70) 340–3016	Apostolescu, Radu

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2008/085730

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 490 216 A (RICHARDSON III FREDERIC B [AU]) 6 February 1996 (1996-02-06)	5,15
A	abstract column 5, line 60 - column 7, line 35 column 11, line 53 - column 13, line 10	1-4, 6-14, 16-21
A	US 2004/187018 A1 (OWEN WILLIAM N [US] ET AL) 23 September 2004 (2004-09-23) the whole document	1-21

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2008/085730

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006095454 A1	04-05-2006	EP 1828931 A2	05-09-2007
		WO 2006050152 A2	11-05-2006
WO 9535533 A	28-12-1995	AU 2666595 A	15-01-1996
US 6158005 A	05-12-2000	AU 6141399 A	27-03-2000
		CA 2343388 A1	16-03-2000
		EP 1119817 A1	01-08-2001
		JP 2002524774 T	06-08-2002
		WO 0014658 A1	16-03-2000
US 5490216 A	06-02-1996	AU 678985 B2	19-06-1997
		AU 4811393 A	12-04-1994
		WO 9407204 A1	31-03-1994
		CA 2145068 A1	31-03-1994
		CN 1103186 A	31-05-1995
		EP 0689697 A1	03-01-1996
		JP 8504976 T	28-05-1996
		NZ 255971 A	26-05-1997
US 2004187018 A1	23-09-2004	NONE	

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 August 2009 (27.08.2009)

PCT

(10) International Publication Number
WO 2009/105702 A2

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/US2009/034765

(22) International Filing Date:
20 February 2009 (20.02.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/030,909 22 February 2008 (22.02.2008) US

(71) Applicant and
(72) Inventor: **ETCHEGOYEN, Craig, S.** [US/US]; C/o Unilock USA Inc., 3333 Michelson Drive, Suite 600, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

(54) Title: LICENSE AUDITING FOR DISTRIBUTED APPLICATIONS

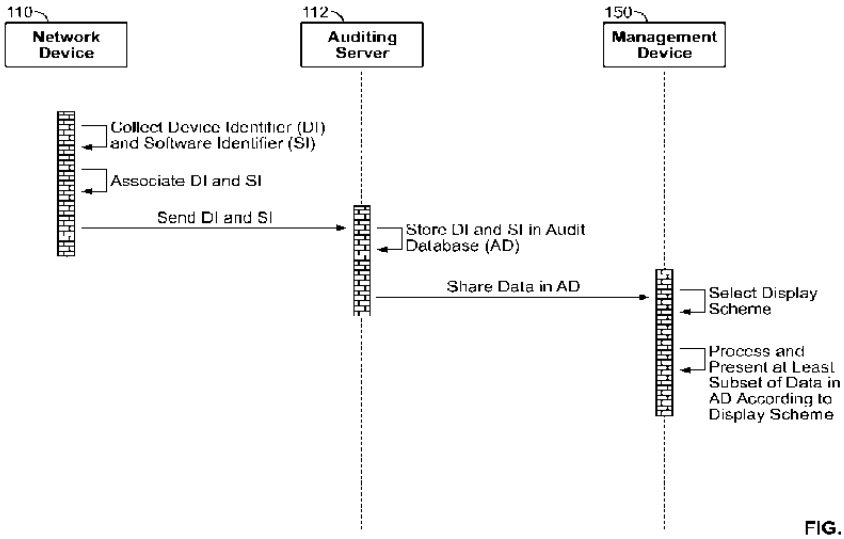


FIG. 13

(57) Abstract: The provided software application includes a module that determines a machine fingerprint of a client device at an appropriate time, such as during initial software load on the client. The fingerprint may comprise various machine-determinable measures of system configuration for the client. Each application copy may be associated with a serial number. A license host may collect serial number, fingerprint and/or IP address information from clients on which the application is installed. The host may generate a map of application installations, including geographic locations of installations and number of unique serial numbers per client in specified regions.

WO 2009/105702 A2

LICENSE AUDITING FOR DISTRIBUTED APPLICATIONS

Background of the Invention

Field of the Invention

[0001] The present invention is directed toward systems for auditing software licenses, and more particularly, to a system that interfaces with a user's device to measure the device hardware configuration and thereby generate a device identifier used to audit software on the device.

Description of the Related Art

[0002] Currently, there are limited ways to audit software licenses. At the same time, software piracy continues to grow at an alarming rate, particularly in emerging economics. In response, software companies have focused primarily on making it more difficult for would-be pirates to install a given software application illegally, such as a single copy of the application on multiple machines. Still, software pirates have found ways to bypass such security measures and install unauthorized software copies on multiple machines. Accordingly, it would be desirable to provide an auditing service for reliable software license authentication and to provide software owners with a measure of how many copies of their software have been legitimately registered and/or how many copies are pirated versions.

Summary of the Invention

[0003] The following presents a simplified summary of one or more embodiments in order to provide a basic understanding of such embodiments. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later.

[0004] In accordance with one or more embodiments and corresponding disclosure thereof, various aspects are described in connection with auditing a license for a software running on a network device. In one embodiment, the device may include: a communication module for communicating with a server; at least one processor in operative communication

with the communication module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor.

[0005] For example, a processor of the device may collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter. A processor (e.g., the same processor involved in collecting the machine parameters and/or a different processor) may generate a device identifier for the device based at least in part on the collected machine parameters. A processor may create an audit number based at least in part on the device identifier. A processor may instruct the communication module to transmit the audit number to an audit database for storage.

[0006] In related aspects, the memory of the device may include executable code for the at least one processor to: collect a software identifier of the software running on the device; and associate the software identifier with the device identifier to generate the audit number.

[0007] In further related aspects, the at least one processor may determine a geo-location code for the device (e.g., an Internet Protocol (IP) address); and associate the geo-location code with at least one of the software identifier and the device identifier to generate the audit number.

[0008] In accordance with other aspects of the embodiments described herein, there is provided a network device adapted to facilitate auditing of a license for a software running on the device. The device may include: a communication module for communicating with an auditing server and an authentication server; at least one processor in operative communication with the communication module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor.

[0009] For example, a processor of the device may collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter. A processor may instruct the communication module to send the collected machine parameters to the auditing server to generate a device identifier for the device based at least in part on the collected machine parameters. A processor may, in response to receiving the device identifier from the server, create an audit number based at least in part on the device identifier. A processor may

APPLE EXHIBIT 1102, Page 690 of 1048

instruct the communication module to transmit the audit number to at least one of the auditing server and the authentication server for storage in at least one audit database.

[0010] In related aspects, the memory may include executable code for the at least one processor to: collect a software identifier of the software running on the device; and associate the software identifier with the device identifier to generate the audit number. In the alternative, or in addition, the memory may include executable code for the at least one processor to: collect a software identifier of the software running on the network device; and instruct the communication module to send the software identifier to the auditing server to generate the device identifier based at least in part on the software identifier.

[0011] In further related aspects, the at least one processor may determine a geo-location code for the device; and associate the code with at least one of the device identifier and the software identifier to generate the audit number. In the alternative, or in addition, the at least one processor may send the code to the auditing server to generate the device identifier based at least in part on the code.

[0012] In accordance with other aspects of the embodiments described herein, there is provided a system for managing a license for a software running on one or more network devices. The system may include a communication module for accessing an audit database, the database comprising audit numbers for the one or more network devices, each audit number comprising a software identifier associated with a device identifier. The system may include: a display module; at least one processor in operative communication with the communication module, and the display module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor.

[0013] For example, a processor of the system may (a) instruct the communication module to access the database and (b) process the audit numbers to, for example, sort the audit numbers according to at least one of activated license seats and unactivated license seats. A processor may select a display scheme for presenting the processed audit numbers, and instruct the display module to present the processed audit numbers according to the selected display scheme. In one approach, the processor may select the display scheme in response to a user choice entered on a user input module in operative communication with the at least one processor.

APPLE EXHIBIT 1102, Page 691 of 1048

[0014] To the accomplishment of the foregoing and related ends, the one or more embodiments comprise the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects of the one or more embodiments. These aspects are indicative, however, of but a few of the various ways in which the principles of various embodiments may be employed and the described embodiments are intended to include all such aspects and their equivalents.

Brief Description of the Drawings

[0015] Figure 1A provides a block diagram of an exemplary system for auditing distributed software.

[0016] Figure 1B provides a block diagram of another exemplary system for auditing distributed software, wherein the system includes an authentication server.

[0017] Figure 2 illustrates the components of an exemplary device identifier.

[0018] Figures 3-12 illustrate exemplary display schemes for presenting data from the audit database regarding one or more network devices running a given software.

[0019] Figure 13 is a sequence diagram for the system of Figure 1A in accordance with an exemplary approach to auditing distributed software.

[0020] Figure 14 is a sequence diagram for the system of Figure 1B in accordance with another exemplary approach to auditing distributed software.

Detailed Description

[0021] The present invention addresses the need for an auditing service that provides reliable software license authentication and provides software owners with a measure of how many copies of their software have been legitimately registered and/or are pirated. Such an auditing service may be used alone, or in conjunction with other security/authentication measures.

APPLE EXHIBIT 1102, Page 692 of 1048

[0022] The present technology provides for an improved system and method for auditing distributed software. In accordance with one aspect of the present technology, there is provided a system and method for authenticating software licenses. With reference to Figure 1A, there is provided an embodiment of a system having a plurality of network devices 110 that are in operative communication with an auditing server 112. While only one network device 110 is illustrated in Figures 1A-1B, it will be understood that a given system may comprise any number of network devices. The network device 110 may be, but is not limited to, a personal computer, a server computer, a laptop computer, a tablet computer, a personal digital assistant, a mobile phone, a wireless communication device, an onboard vehicle computer, or any other device capable of communication with a computer network.

[0023] The network device 110 may comprise a software 120 that requires a license to be authorized for use. The device 110 may also comprise an auditing tool or application 122. The auditing application 122 may be any program or application that collects identifying information regarding the network device 110 and/or any software (e.g., software 120) on the network device 110. The auditing application 122 may comprise a stand alone application or an applet running within a web browser on the device 110 (e.g., an applet comprising executable code for a Java Virtual Machine). The auditing application 122 may be embedded in or associated with another software application, including but not limited to software 120. For example, the auditing application 122 may be embedded in or associated with a tool bar of a software application, such as, for example, a web browser. The auditing application 122 may prompt the user to register with an online software registration service, or may run in the background with little or no interaction with the user of device 110.

[0024] The auditing application 122 may include a registration routine that collects information regarding network device 110 by checking a number of parameters which are expected to be unique to the network device environment. The parameters checked may include, for example, hard disk volume name, user name, device name, user password, hard disc initialization date, etc.. The collected information may include information that identifies the hardware comprising the platform on which the web browser runs, such as, for example, CPU number, or unique parameters associated with the firmware in use. The system information may further include system configuration information, such as amount of memory, type of processor, software or operating system serial number, etc.

APPLE EXHIBIT 1102, Page 693 of 1048

[0025] Based on the collected information, the auditing application 122 may generate a device identifier 124 that is unique for the user computer 110. In the alternative, or in addition, the application 122 may gather and send the device parameters to a remote server, such as auditing server 112, which in turn generates the device identifier 124. The device identifier 124 may be stored in a hidden directory of the device 110 and/or at a remote location, such as the auditing server 112. The device identifier 124 may incorporate the device's IP address and/or other geo-location code to add another layer of specificity to device's unique identifier.

[0026] It is noted that an application (e.g., auditing application 122) running on the network device or otherwise having access to the network device's hardware and file system may generate a unique device identifier (e.g., device identifier 124) using a process that operates on data indicative of the network device's configuration and hardware. The device identifier may be generated using a combination of user-configurable and non-user-configurable machine parameters as input to a process that results in the device identifier, which may be expressed in digital data as a binary number. Each machine parameter is data determined by a hardware component, software component, or data component specific to the device that the unique identifier pertains to. Machine parameters may be selected based on the target device system configuration such that the resulting device identifier has a very high probability (e.g., greater than 99.999%) of being unique to the target device. In addition, the machine parameters may be selected such that the device identifier includes at least a stable unique portion up to and including the entire identifier, that has a very high probability of remaining unchanged during normal operation of the target device. Thus, the resulting device identifier should be highly specific, unique, reproducible and stable as a result of properly selecting the machine parameters.

[0027] The application for generating the device identifier may also operate on the collected parameters with one or more algorithms to generate the device identifier. This process may include at least one irreversible transformation, such as, for example, a cryptographic hash function, such that the input machine parameters cannot be derived from the resulting device identifier. Each device identifier, to a very high degree of certainty, cannot be generated except by the suitably configured application operating or otherwise having had access to the same field security device for which the device identifier was first

generated. Conversely, each identifier, again to a very high degree of certainty, can be successfully reproduced by the suitably configured application operating or otherwise having access to the same field security device on which the identifier was first generated.

[0028] The application may operate by performing a system scan to determine a present configuration of the field security device. The application may then select the machine parameters to be used as input for generating the unique device identifier. Selection of parameters may vary depending on the system configuration. Once the parameters are selected, the application may generate the identifier.

[0029] Further, generating the device identifier may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties as well as a variety of additional parameters such as uniquely generated hashes and time sensitive values. Physical device parameters available for sampling may include, for example, unique manufacturer characteristics, carbon and silicone degradation and small device failures.

[0030] The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as heat and additional stresses placed on a chip during the sampling process allows for each and every benchmark to reproduce the expected values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms.

[0031] In addition to the chip benchmarking and degradation measurements, the process for generating a device identifier may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

[0032] Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further

APPLE EXHIBIT 1102, Page 695 of 1048

extended by the use of complex encryption algorithms to convolute the device identifier values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

[0033] The device identifier may be generated by utilizing machine parameters associated with one or more of the following: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

[0034] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

[0035] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: memory model; memory slots; memory total; and memory details.

[0036] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: video model; video details; display model; display details; audio model; and audio details.

[0037] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: network model; network address; Bluetooth address; Blackbox model (including IDE and SCSI); Blackbox serial; Blackbox details; Blackbox damage map; Blackbox volume name; NetStore details; and NetStore volume name.

[0038] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

[0039] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

[0040] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

[0041] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller.

[0042] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

[0043] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

[0044] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

[0045] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD.

[0046] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

[0047] With reference to Figure 2, in one embodiment, the device identifier 124 may include two components – namely, a variable key portion 126 and a system key portion 128. The variable key portion 126 may be generated at the time of registration of network device 110 by reference to a variable platform parameter, such as via reference to system time information, although other parameters which are variable may be utilized in other embodiments. The system key portion 128 may include the above described parameters expected to be unique to the device 110, such as, for example, hard disk volume name, user

name, computer name, user password, hard disc initialization date, or combinations thereof. Portions 126 and/or 128 may be combined with the IP address and/or other platform parameters of the device 110. Further details regarding device identifiers 124, such as machine fingerprints and parameters expected to be unique for the a given computer, can be found in U.S. Application No. 08/124,718, filed September 21, 1993, titled "System for Software Registration," now issued as U.S. Patent No. 5,490,216, which application is specifically incorporated herein, in its entirety, by reference. It is noted that device identifiers, or portions thereof, may be encrypted to add an additional layer of specificity and security.

[0048] With reference once again to Figure 1A, the auditing application 122 may also include a registration routine that collects or receives information regarding the software 120 on device 110 by checking information which is expected to be unique to software 120, such as, for example, the software serial number. The collected software identifier may include the software serial number, product identification number, product key, etc. The collected software identifier may include information regarding where the software was sold or distributed, who the buyers, sellers, and/or distributors were, which stores the software was sold in, etc. It is noted that the software identifier may be unique to particular copy of software, such as when the software is licensed to a single user In the alternative, or in addition, the software identifier may be unique to particular type or group of a software, such as when the software is licensed to a defined group of users.

[0049] The embodiments described herein comprise an auditing application 122 that collects the software identifier 130 for software 120; however, it will be understood that the systems and components described herein can be adapted to collect one or more types of software identifiers for a plurality of software applications. The software identifier 130 may be stored in a hidden directory of the device 110 and/or at a remote location, such as the auditing server 112.

[0050] The auditing application 122 may also include a registration routine that collects or receives information regarding the geo-location code 140 of the device 110. The geo-locater 140 may comprise the IP address or the like of the device 110.

[0051] Auditing application 122 may electronically send the device identifier 124 and the software identifier 130 to the auditing server 112. In the alternative, or in addition, a geo-

APPLE EXHIBIT 1102, Page 698 of 1048

location code 140, such as the IP address of the device 110, may be associated with the device identifier 124 and/or the software identifier 130 and may sent to the auditing server 112, such as via a secured network connection. The auditing server 112 may encrypt and store the data, such as the device identifier 124, the software identifier 130, and/or the geo-location code 140, received from the network device 110. The auditing server 112 may receive such data from a plurality of network devices and store the received data in an audit database 114.

[0052] In one embodiment, the auditing application 122 may generate an audit number 142 by associating the software identifier 130 with the device identifier 124 and/or geo-location code 140, and may send the generated audit number 142 to the auditing server 112. In another embodiment, the application 122 may send the device identifier 124, the software identifier 130, and/or the geo-location code 140 to the server 112 in a piecemeal manner. The server 112 may in turn generate the audit number 142. The auditing server 112 may receive or generate audit numbers from a plurality of network devices and store the received audit numbers in the audit database 114.

[0053] It is noted that the audit number 142 may be generated from the device identifier 124, the software identifier 130, and/or the geo-location code 140 via any number of suitable approaches. For example, the software identifier 130 may be concatenated or linked with the device identifier 124 and/or geo-location code 140. It is also noted that the audit number 142 may be stored in a hidden directory of the device 110 and/or at a remote location, such as the auditing server 112. It is further noted that the device identifier 124, the software identifier 130, and/or the geo-location code 140 may at a later time be extracted from the audit number 142.

[0054] When a user of a network device, including but not limited to network device 110, installed with auditing application 122, attempts to run the software 120, the auditing application 122 in response may transmit the software identifier 130 associated with the device identifier 124 and/or the geo-location code 140 (or an audit number 142 generated from such data) to the auditing server 112, which in turn may store the received data in the audit database 114.

[0055] With reference to Figure 1B, there is provided an embodiment of a system that further comprises an optional authentication server 116 that is in operative communication

APPLE EXHIBIT 1102, Page 699 of 1048

with the auditing server 112. When a given user tries to run software 120 on his/her network device, the authentication server 116 may access the audit database 114 on auditing server 112 to determine whether to allow his/her of the software 120. The authentication server 116 may receive/access the license terms for a particular software from the auditing server 112 or another server or network device.

[0056] In one embodiment, the authentication server 116 may disallow use of the software 120 beyond a defined maximum number of allowed users or seats (which may be defined by the software license). The server 116 may analyze the data in the audit database 114 and determine how many seats are currently utilizing software 120. If the number of currently allowed seats meets or exceeds the maximum number of allowed seats, the server 116 may throttle or disallow the use of software 120 by more seats; otherwise, the server may allow the use of the software 120. In another embodiment, the authentication server 116 may throttle or disallow use of the software 120 if its software identifier is already associated with a different device identifier and/or a different IP address in the audit database 114; otherwise, the server 116 may allow use of the software 120. The authentication server 116 may collect data regarding the instances of allowed and disallowed software use, and may share such data with the auditing server 112.

[0057] With reference to the embodiments of Figures 1A and 1B, the auditing server 112 may be in operative communication with a management device 150, which may be any device capable of communication with a computer network, such as, for example, a personal computer, a server computer, a laptop computer, a tablet computer, a personal digital assistant, a mobile phone, or a wireless communication device. The management device 150 may comprise a management application 152, which may be any program or application, such as a stand alone application or an application that is embedded or associated with another software application, such as an applet running within a web browser on the device 150.

[0058] The management application 152 may be adapted to allow a user, such as, for example, a software manufacturer or distributor, to view the data collected and stored in the audit database 114 of the audit server 112. The present embodiment will be described in the context of a software manufacturer utilizing the management application 152. However, it will be understood that any user of the management device may utilize the management application 152.

APPLE EXHIBIT 1102, Page 700 of 1048

[0059] The management application 152 may present the data in the audit database 114 in a manner that allows its user to better understand how its software is being used, legitimately or otherwise. The information organized and presented according to one or more display schemes of the application 152 may allow a software manufacturer to better understand software consumer behaviors and habits, which in turn may allow the manufacturers to adjust or modify their licensing rules to comport with their business goals.

[0060] The management application 152 may be adapted to process and/or present at least a subset of the data in the audit database 114 according to one or more display schemes. The display schemes may be predefined or presented for selection by the software manufacturer. The data in the audit database 114 (i.e., the audit numbers 142 and components thereof for network devices 110) may be organized or sorted by the number of activated/unactivated seats, license seat trends, activations trends, piracy curves, etc., as shown in Figures 3-12. It is noted that the data in the audit database 114 may be organized, processed, and processed by the management device 152 and/or the auditing server 112.

[0061] The exemplary display scheme of Figure 3 provides an activation seats trend, which shows the number of activated seats over a period of time for three types of software 310, 320, and 330 (e.g., corresponding to "2D Sketch and Plan," "3D Floorplan" and "Home Plan Pro," respectively). The data presented according to the display scheme of Figure 3 may allow one to determine licensing trends and correlate marketing and sales efforts to activation events.

[0062] The exemplary display scheme of Figure 4 provides the license seat trend 410, the activation seats trend 420, the activated-per-licensed-seat trend 430, and the activations trend 440 for three types of software (e.g., 310, 320, 330 in Figure 3).

[0063] The exemplary display scheme of Figure 5 provides a piracy heat map which may allow one to visualize the extent/level of piracy in different geographic regions. It is noted that such piracy heat maps and other geographically based summaries may be based at least in part on the geo-location codes associated with the corresponding software identifiers. The data presented on the piracy heat map may be filtered by piracy, activations, number of seats or licenses sold, etc. The display scheme may further comprise a pivot table or the like to supplement the piracy heat map.

APPLE EXHIBIT 1102, Page 701 of 1048

[0064] The exemplary display scheme of Figure 6 presents the license seat utilization (activated vs. deactivated seats) summary 610 and the activation trend (activations vs. deactivations) summary 620. The exemplary display scheme of Figure 7 presents the license seat trend 710, the activation seats trend 720, the activated-per-licensed-seat trend 730, and the activation trend 740 for three types of software (e.g., "Escape from Alcatraz," "Marks vs. Ninjas II" and "Revenge of the Pirates").

[0065] The exemplary display scheme of Figure 8 presents activated-versus-unactivated pie charts 810, the licensee seat trend 820, and the activation seats trend 830. Similarly, the exemplary display scheme of Figure 9 presents activated-versus-unactivated pie charts 910, the licensee seat trend 920, and the activation seats trend 930 for five types of software (e.g., "2D Sketch and Plan," "3D Floorplan," "Design CAD," "Home Plan Pro" and "IC Digital Design"). The presentation of data according to the display schemes of Figures 8 and 9 may make it possible to identify accounts that are at or above their licensed seats. In addition, the data presented in Figures 8 and 9 make it possible to identify licenses that are not being utilized and to properly allocate resources to ensure continued revenue streams.

[0066] The exemplary display scheme of Figure 10 presents piracy curves 1010, the top ten abusing countries summary 1020, and the pirated activations summary 1030. It is again noted that such geographically based summaries may be based at least in part on the geo-location codes associated with the corresponding software identifiers.

[0067] It is noted that the data in the audit database 114 may be used to model or simulate the effect of certain actions taken by the software manufacturer. The information generated by such licensing models or simulators may also be based in part on other trends/data. For example, the models may be based in part on historical, current, and/or forecast trends/data associated with a particular software identifier. Similarly, the models may be based in part on trends/data associated with a particular class or types of software and/or trends/data for the software industry. The models may be based in part on trends/data associated with other industries, such as, for example, the hardware industry. For example, the management application 152 may simulate the effect of throttling or disallowing the further use of a given software. The exemplary display scheme of Figure 11 summarizes the effect of such throttling by displaying the throttling-effectiveness summary 1110 and the throttling-rule-impact summary 1120. Similarly, the exemplary display scheme of Figure 12 presents the throttling-

APPLE EXHIBIT 1102, Page 702 of 1048

effectiveness summary 1210 and the throttling-rule-impact summary 1220. The data presented according to the display schemes of Figures 11 and 12 may include data generated by what-if simulators or models, which in turn may allow one to better understand how the software licenses are consumed.

[0068] It will be understood that the described system for auditing software usage by network device users can comprise any number of components or modules adapted to perform various functions or tasks. For example, with reference to Figure 13, there is provided one embodiment of a software license auditing system wherein an auditing application on a network device 110 may collect information regarding the device 110 by checking hardware parameters expected to be unique to the network device environment. The auditing application may generate a device identifier based on the collected hardware information. In the alternative, or in addition, the auditing application may provide the collected hardware information to another device or server, which in turn may generate the hardware identifier.

[0069] The auditing application may collect or receive information regarding a given software on the network device 110, such as, for example, a software serial number. The auditing application may optionally collect or receive information regarding collects a geo-location code 140 of the device 110, such as, for example, the IP address for the device 110.

[0070] The auditing application may associate the software identifier with the hardware identifier and/or the geo-location code, and may provide such data to an auditing server 112. The auditing server 112 may receive such data from a plurality of network devices and store the received data in an audit database.

[0071] The auditing server 112 share the data in the audit database with a management device 150. The management device 150 may comprise a management application that selects a display scheme for presenting data in the audit database. The management application may automatically determine the appropriate display scheme, or may allow the management device user to select a display scheme from a menu or list. The management application may process and/or present at least a subset of the data in the audit database according to the selected display scheme.

[0072] With reference to Figure 14, there is provided another embodiment of an software license auditing system that is similar to the embodiment of Figure 13, but further comprises

APPLE EXHIBIT 1102, Page 703 of 1048

an optional authentication server 116. The authentication server 116 may detect when a network device user is trying to run a given software on his/her network device. The authentication server 116 may receive/access and use the data in the audit database of the auditing server 112 to determine whether to allow the attempted use the given software. The authentication server 116 may further receive/access the license terms for a particular software from the auditing server 112 or another server or network device. In one approach, the server 116 may send a throttle or disallow use command to the network device if the number of currently allowed seats meets or exceeds the maximum number of allowed seats for the given software, which may be defined by the software license terms. In another approach, the server 116 may send a throttle or disallow command to the network device if software identifier for the given software is already associated with a different device identifier and/or a different IP address in the audit database. Otherwise, the server 116 may send an allowed use command to the network device to allow use of the given software.

[0073] In accordance with one or more aspects of the embodiments described herein, there are provided techniques for auditing licenses for software running on one or more network devices. In one embodiment, there is provided a network device comprising: a communication module for communicating with a server (e.g., audit database); at least one processor in operative communication with the communication module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to perform a number of steps. For example, the at least one processor may: collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter; generate a device identifier for the device based at least in part on the collected machine parameters; create an audit number based at least in part on the device identifier; and instruct the communication module to transmit the audit number to an audit database for storage.

[0074] In related aspects, the memory may comprise executable code for the at least one processor to: collect a software identifier of the software running on the device; and associate the software identifier with the device identifier to generate the audit number. The software identifier may comprise: a software serial number; information regarding a geographic region

APPLE EXHIBIT 1102, Page 704 of 1048

associated with the software; and/or information regarding at least one of a seller, a buyer, and a distributor of the software.

[0075] In further related aspects, the at least one processor may associate the software identifier with the device identifier by concatenating the software identifier with the device identifier. In the alternative, or in addition, the at least one processor may associate the software identifier with the device identifier by linking the software identifier with the device identifier.

[0076] In yet further related aspects, the at least one processor may: determine a geo-location code for the device; and associate the geo-location code with at least one of the software identifier and the device identifier to generate the audit number. The geo-location code comprises an Internet Protocol (IP) address of the device.

[0077] In other aspects, the at least one processor may generate the device identifier by implementing at least one irreversible transformation (e.g., a cryptographic hash function) such that the collected machine parameters cannot be derived from the device identifier.

[0078] In accordance with one or more aspects of the embodiments described herein, there is provided another embodiment of a network device adapted to facilitate auditing of a license for a software running on the device. The device may comprise: a communication module for communicating with an auditing server and an authentication server; at least one processor in operative communication with the communication module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to perform a number of tasks. For example, the at least one processor may: collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter; instruct the communication module to send the collected machine parameters to the auditing server to generate a device identifier for the device based at least in part on the collected machine parameters; in response to receiving the device identifier from the server, create an audit number based at least in part on the device identifier; and instruct the communication module to transmit the audit number to at least one of the auditing server and the authentication server for storage in at least one audit database.

APPLE EXHIBIT 1102, Page 705 of 1048

[0079] In related aspects, the memory may further comprise executable code for the at least one processor to: collect a software identifier of the software running on the device; and associate the software identifier with the device identifier to generate the audit number.

[0080] In further related aspects, the memory may further comprise executable code for the at least one processor to: collect a software identifier of the software running on the network device; and instruct the communication module to send the software identifier to the auditing server to generate the device identifier based at least in part on the software identifier.

[0081] In other related aspects, the at least one processor may: determine a geo-location code (e.g., IP address) or the device; and associate the code with at least one of the device identifier and the software identifier to generate the audit number. In the alternative, or in addition, the at least one processor may: determine a geo-location code for the device; and instruct the communication module to send the code to the auditing server to generate the device identifier based at least in part on the code.

[0082] In accordance with one or more aspects of the embodiments described herein, there is provided a system for managing a license for a software running on one or more network devices, comprising: a communication module for accessing an audit database, the database comprising audit numbers for the one or more network devices, each audit number comprising a software identifier associated with a device identifier, each device identifier being generated from a combination of user-configurable and non-user-configurable machine parameters for a given network device; a display module; at least one processor in operative communication with the communication module, and the display module; and a memory in operative communication with the at least one processor and comprising executable code for the at least one processor. For example, the at least one processor may: instruct the communication module to access the database; process the audit numbers to sort the audit numbers according to at least one of activated license seats and unactivated license seats; select a display scheme for presenting the processed audit numbers; and instruct the display module to present the processed audit numbers according to the selected display scheme. The at lest one processor may select the display scheme in response to a user choice entered on a user input module in operative communication with the at least one processor.

APPLE EXHIBIT 1102, Page 706 of 1048

[0083] In related aspects, the at least one processor may sort the audit numbers according to at least one of authorized software copies and pirated software copies. In the alternative, or in addition, the at least one processor may sort the audit numbers according to geographic data regarding the software.

[0084] In further related aspects, the at least one processor may identify a trend with respect to at least one of the activated license seats and the unactivated license seats. The at least one processor may identify a trend with respect to at least one of authorized software copies and pirated software copies. The at least one processor identifies a trend with respect to geographic data regarding the software. The at least one processor may instruct the display module to display the processed audit numbers as a piracy heat map or the like.

[0085] While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

[0086] As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

APPLE EXHIBIT 1102, Page 707 of 1048

[0087] It is understood that the specific order or hierarchy of steps in the processes disclosed herein in an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

[0088] Moreover, various aspects or features described herein can be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer-readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips, etc.), optical disks (e.g., compact disk (CD), digital versatile disk (DVD), etc.), smart cards, and flash memory devices (e.g., Erasable Programmable Read Only Memory (EPROM), card, stick, key drive, etc.). Additionally, various storage media described herein can represent one or more devices and/or other machine-readable media for storing information. The term "machine-readable medium" can include, without being limited to, wireless channels and various other media capable of storing, containing, and/or carrying instruction(s) and/or data.

[0089] Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 708 of 1048

WHAT IS CLAIMED IS:

1. A network device adapted to facilitate auditing of a license for a software running on the device, comprising:

a communication module for communicating with a server;

at least one processor in operative communication with the communication module; and

a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to:

collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter;

generate a device identifier for the device based at least in part on the collected machine parameters;

create an audit number based at least in part on the device identifier; and

instruct the communication module to transmit the audit number to an audit database for storage.

2. The device of Claim 1, wherein the memory further comprises executable code for the at least one processor to:

collect a software identifier of the software running on the device; and

associate the software identifier with the device identifier to generate the audit number.

3. The device of Claim 2, wherein the software identifier comprises a software serial number.

4. The device of Claim 2, wherein the software identifier comprises information regarding a geographic region associated with the software.

5. The device of Claim 2, wherein the software identifier comprises information regarding at least one of: a seller; a buyer; and a distributor of the software.

APPLE EXHIBIT 1102, Page 709 of 1048

6. The device of Claim 2, wherein the at least one processor associates the software identifier with the device identifier by concatenating the software identifier with the device identifier.

7. The device of Claim 2, wherein the at least one processor associates the software identifier with the device identifier by linking the software identifier with the device identifier.

8. The device of Claim 2, wherein the at least one processor:

determines a geo-location code for the device; and

associates the geo-location code with at least one of the software identifier and the device identifier to generate the audit number.

9. The device of Claim 8, wherein the geo-location code comprises an Internet Protocol (IP) address of the device.

10. The device of Claim 1, wherein the at least one processor generates the device identifier by implementing at least one irreversible transformation such that the collected machine parameters cannot be derived from the device identifier.

11. The device of Claim 10, wherein the at least one irreversible transformation comprises a cryptographic hash function.

12. The device of Claim 1, wherein the server comprises the audit database.

13. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

14. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

15. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: network model; network address; Bluetooth address; Blackbox

APPLE EXHIBIT 1102, Page 710 of 1048

model (including IDE and SCSI); Blackbox serial; Blackbox details; Blackbox damage map; Blackbox volume name; NetStore details; and NetStore volume name.

16. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

17. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

18. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: chassis manufacturer; chassis type; chassis version; and chassis serial number.

19. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: IDE controller; SATA controller; RAID controller; and SCSI controller.

20. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: port connector designator; port connector type; port connector port type; and system slot type.

21. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

22. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

23. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: device model; device model IMEI; device model IMSI; and device model LCD.

APPLE EXHIBIT 1102, Page 711 of 1048

24. The device of Claim 1, wherein the machine parameters comprise information regarding at least one of: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

25. A network device adapted to facilitate auditing of a license for a software running on the device, comprising:

a communication module for communicating with an auditing server and an authentication server;

at least one processor in operative communication with the communication module; and

a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to:

collect machine parameters of the device, the collected machine parameters comprising a combination of at least one user-configurable parameter and at least one non-user-configurable parameter;

instruct the communication module to send the collected machine parameters to the auditing server to generate a device identifier for the device based at least in part on the collected machine parameters;

in response to receiving the device identifier from the server, create an audit number based at least in part on the device identifier; and

instruct the communication module to transmit the audit number to at least one of the auditing server and the authentication server for storage in at least one audit database.

APPLE EXHIBIT 1102, Page 712 of 1048

26. The device of Claim 25, wherein the memory further comprises executable code for the at least one processor to:

collect a software identifier of the software running on the device; and

associate the software identifier with the device identifier to generate the audit number.

27. The device of Claim 25, wherein the memory further comprises executable code for the at least one processor to:

collect a software identifier of the software running on the network device; and

instruct the communication module to send the software identifier to the auditing server to generate the device identifier based at least in part on the software identifier.

28. The device of Claim 27, wherein the at least one processor:

determines a geo-location code for the device; and

associates the code with at least one of the device identifier and the software identifier to generate the audit number.

29. The device of Claim 28, wherein the code comprises an Internet Protocol (IP) address of the device.

30. The device of Claim 25, wherein the at least one processor:

determines a geo-location code for the device; and

instructs the communication module to send the code to the auditing server to generate the device identifier based at least in part on the code.

APPLE EXHIBIT 1102, Page 713 of 1048

31. An system for managing a license for a software running on one or more network devices, comprising:

a communication module for accessing an audit database, the database comprising audit numbers for the one or more network devices, each audit number comprising a software identifier associated with a device identifier, each device identifier being generated from a combination of user-configurable and non-user-configurable machine parameters for a given network device;

a display module;

at least one processor in operative communication with the communication module, and the display module; and

a memory in operative communication with the at least one processor and comprising executable code for the at least one processor to:

instruct the communication module to access the database;

process the audit numbers to sort the audit numbers according to at least one of activated license seats and unactivated license seats;

select a display scheme for presenting the processed audit numbers; and

instruct the display module to present the processed audit numbers according to the selected display scheme.

32. The system of Claim 31, wherein the at lest one processor selects the display scheme in response to a user choice entered on a user input module in operative communication with the at least one processor.

33. The system of Claim 31, wherein the at least one processor sorts the audit numbers according to at least one of authorized software copies and pirated software copies.

34. The system of Claim 31, wherein the at least one processor sorts the audit numbers according to geographic data regarding the software.

35. The system of Claim 31, wherein the at least one processor identifies a trend with respect to at least one of the activated license seats and the unactivated license seats.

36. The system of Claim 31, wherein the at least one processor identifies a trend with respect to at least one of authorized software copies and pirated software copies.

APPLE EXHIBIT 1102, Page 714 of 1048

37. The system of Claim 31, wherein the at least one processor identifies a trend with respect to geographic data regarding the software.

38. The system of Claim 31, wherein the at least one processor instructs the display module to display the processed audit numbers as a piracy heat map.

39. A method for auditing a license for a software running on a network device, comprising:

collecting machine parameters of the device, the collected machine parameters a combination of user-configurable and non-user-configurable machine parameters;

generating a device identifier for the device based at least in part on the collected machine parameters;

collecting a software identifier of the software running on the device;

associating the software identifier with the device identifier to generate an audit number;

transmitting the audit number to a server for storage in an audit database.

40. The method of Claim 39, further comprising:

determining a geo-location code for the device; and

associating the code with at least one of the software identifier and the device identifier to generate the audit number.

APPLE EXHIBIT 1102, Page 715 of 1048

41. A method for auditing a license for a software running on a network device, comprising:

collecting machine parameters of the device, the collected machine parameters a combination of user-configurable and non-user-configurable machine parameters;

collecting at least one software identifier of the software running on the device;

sending the machine parameters and the at least one software identifier to a server to generate a device identifier based at least in part on the machine parameters and the at least one software identifier;

in response to receiving the device identifier, creating an audit number based at least in part on the received audit number; and

transmitting the audit number to an audit database for storage.

42. The method of Claim 39, further comprising:

determining a geo-location code associated with the device; and

sending the code to the server to generate the device identifier based at least in part on the code.

APPLE EXHIBIT 1102, Page 716 of 1048

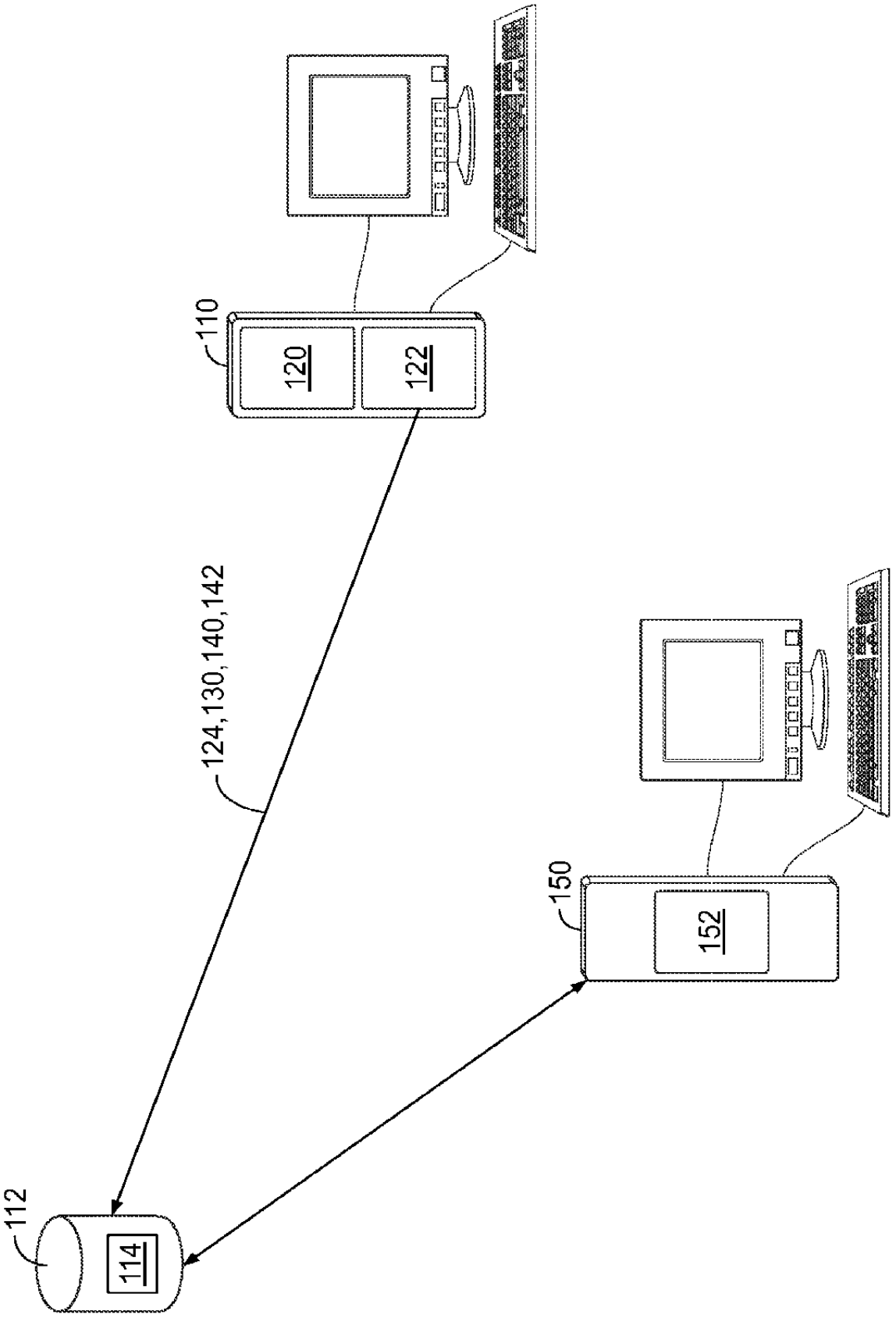


FIG. 1A

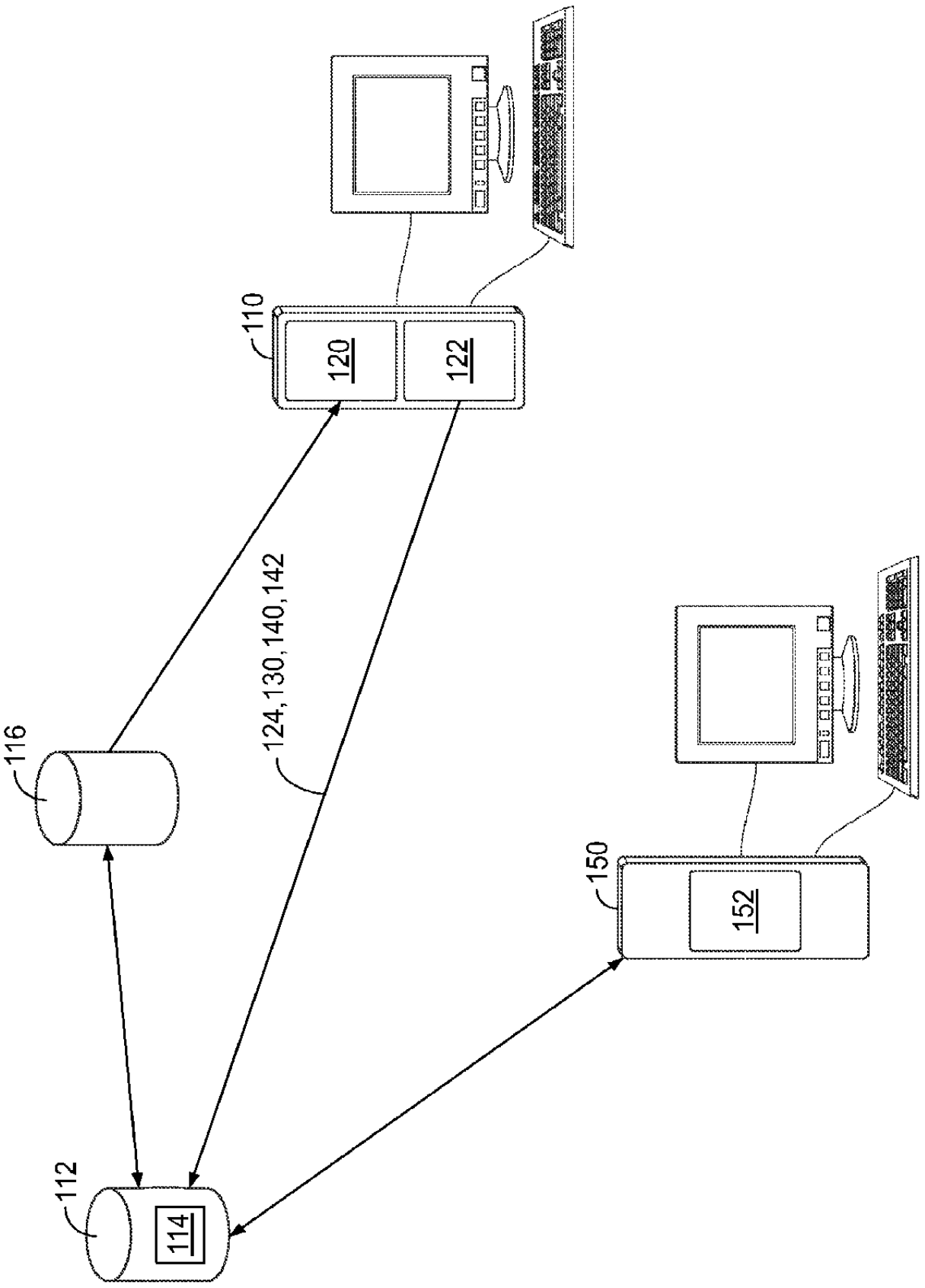


FIG. 1B

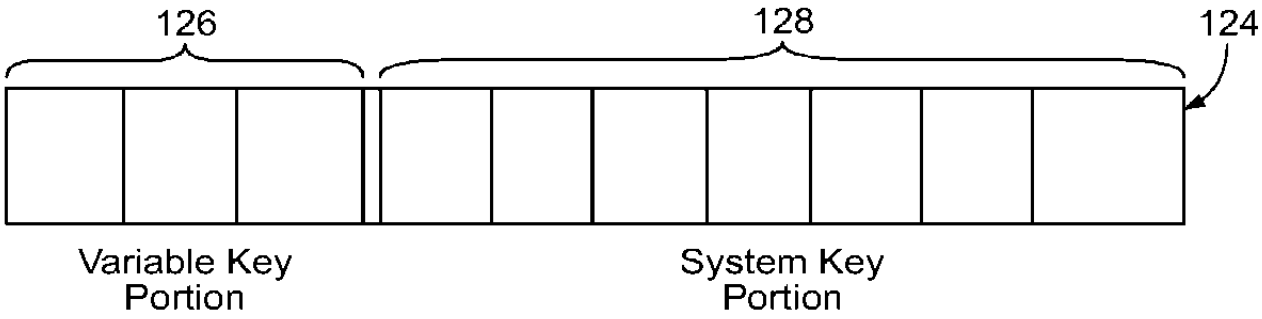


FIG. 2

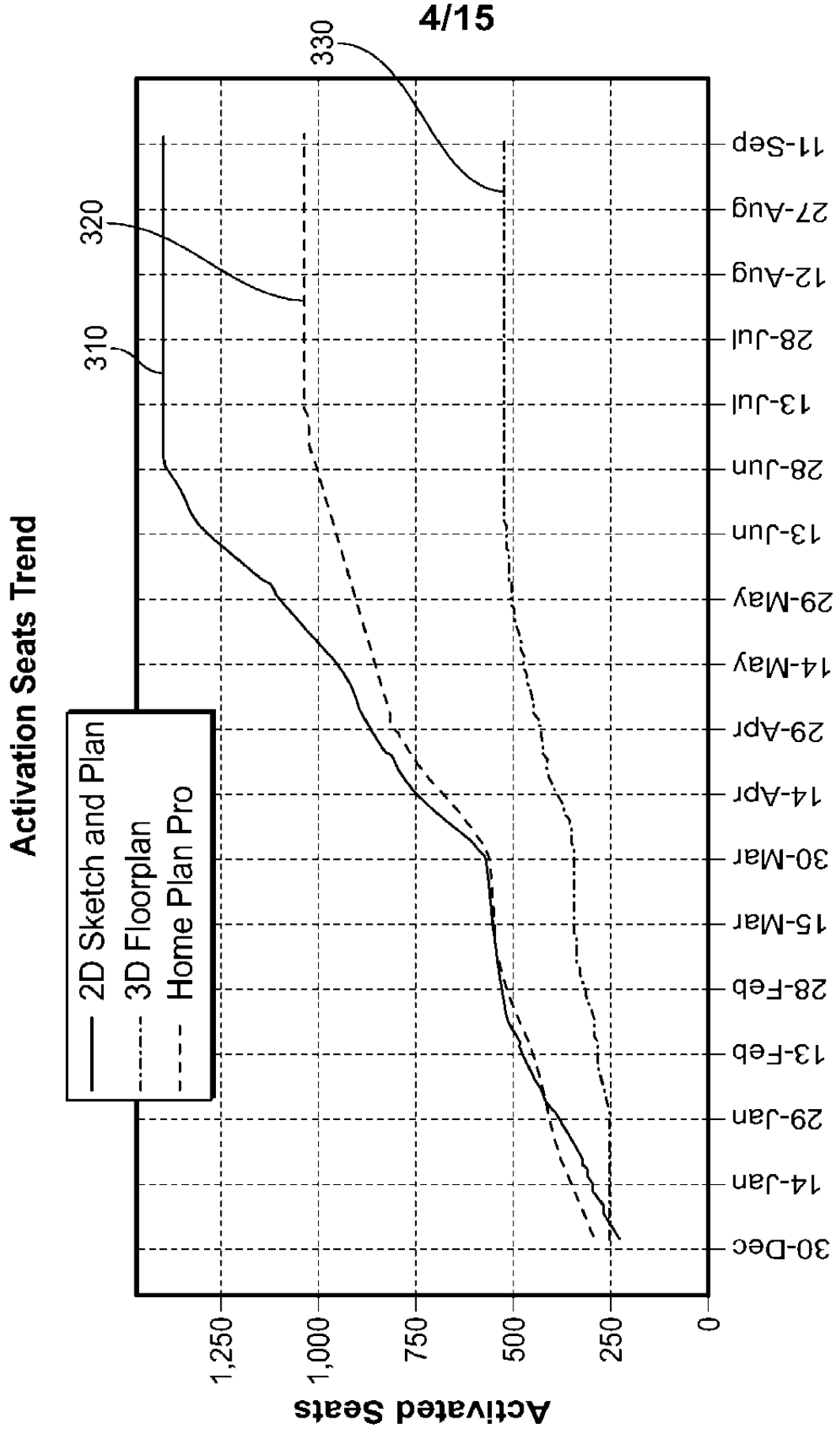

Activation Seats Trend
2D Sketch and Plan
3D Floorplan
Home Plan Pro
330
320
310
Activated Seats
1,250
1,000
750
500
250
0
Time
30-Dec
14-Jan
29-Jan
13-Feb
28-Feb
15-Mar
30-Mar
14-Apr
29-Apr
14-May
29-May
13-Jun
28-Jun
13-Jul
28-Jul
12-Aug
27-Aug
11-Sep

FIG. 3

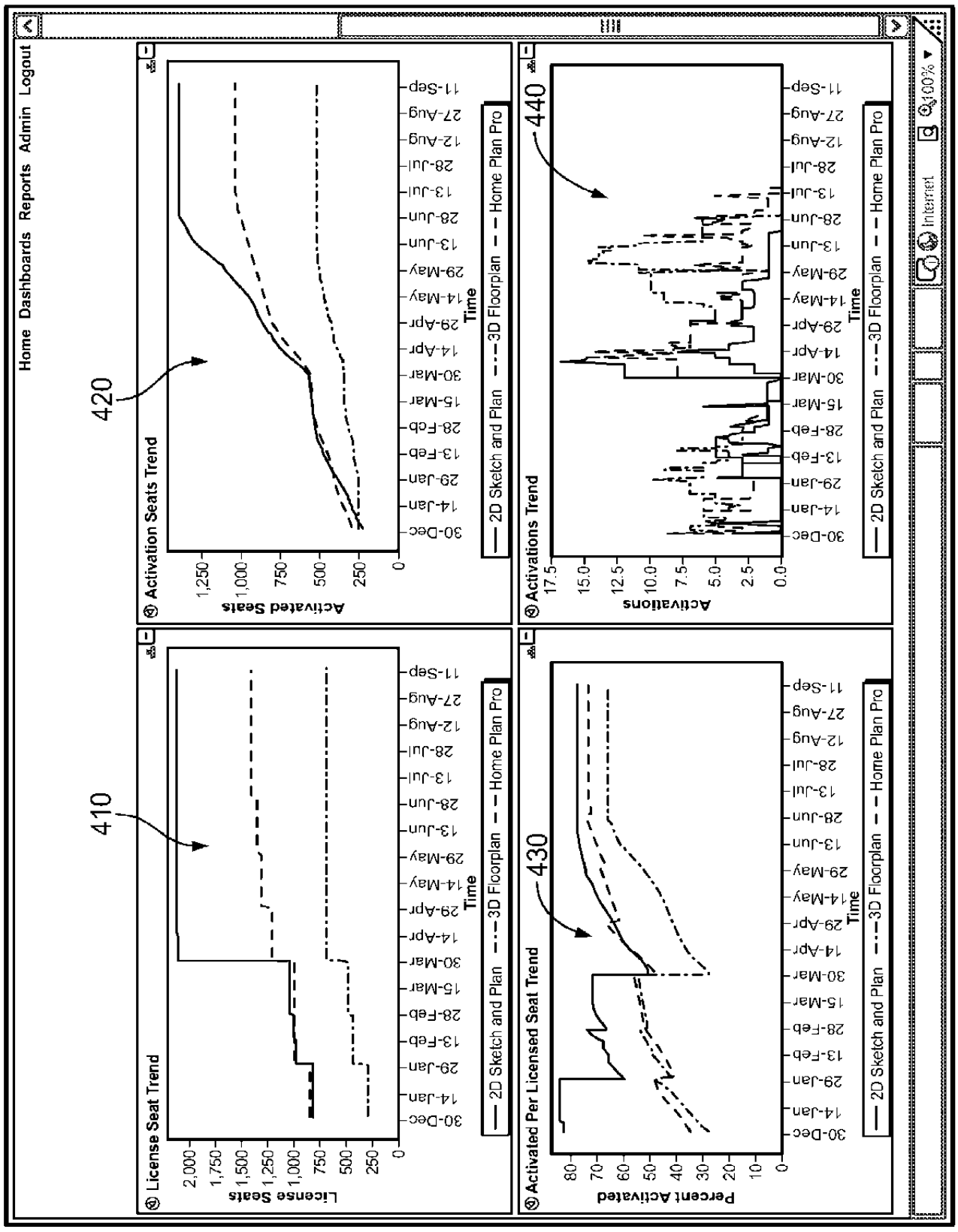
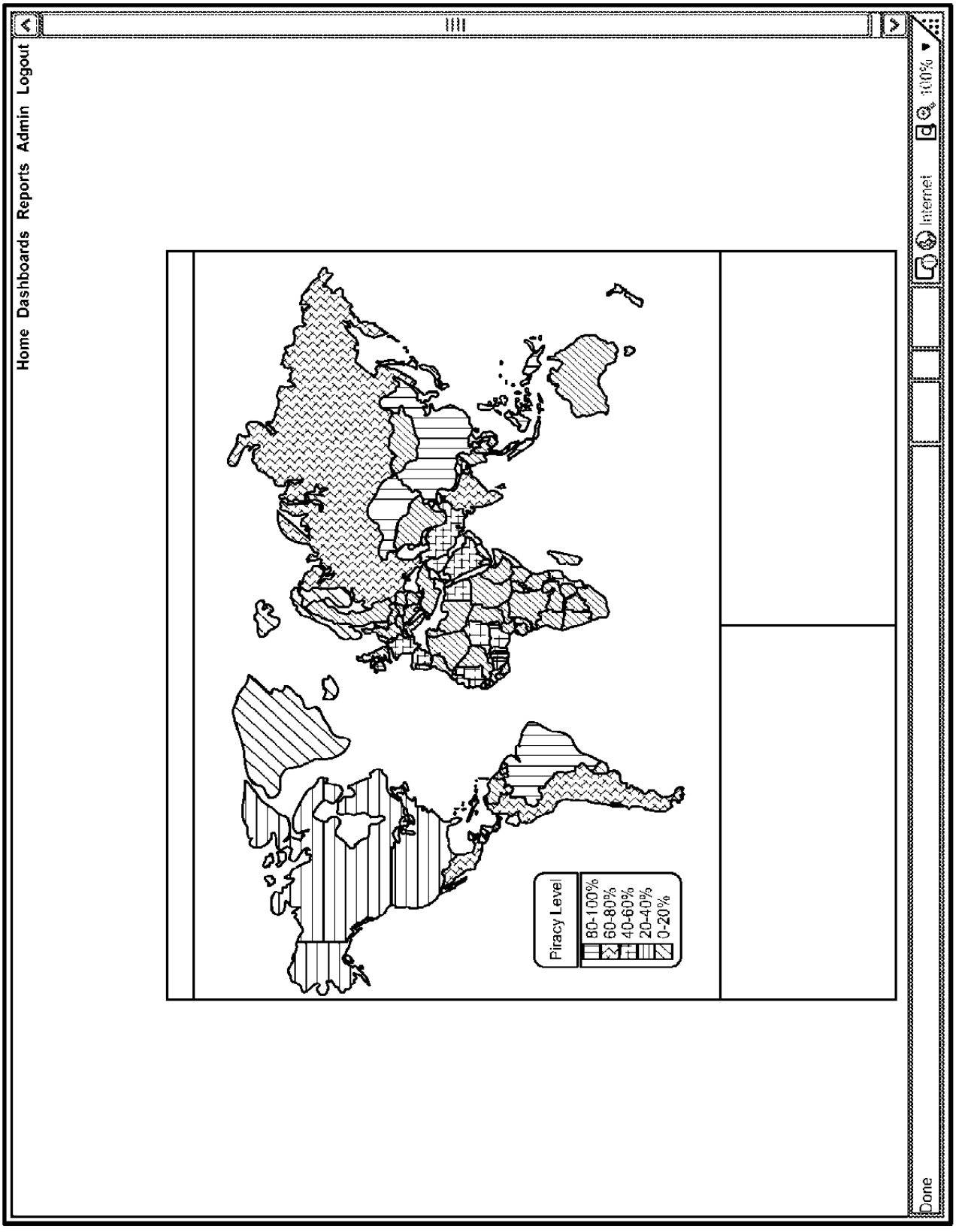

Home Dashboards Reports Admin Logout
410
420
430
440
License Seat Trend
Activation Seats Trend
Activated Per Licensed Seat Trend
Activations Trend
License Seats
Activated Seats
Percent Activated
Activations
Time
2D Sketch and Plan
3D Floorplan
Home Plan Pro
Internet
100%

FIG. 4

FIG. 5


Home Dashboards Reports Admin Logout
Piracy Level
80-100%
60-80%
40-60%
20-40%
0-20%
Internet
100%
Done

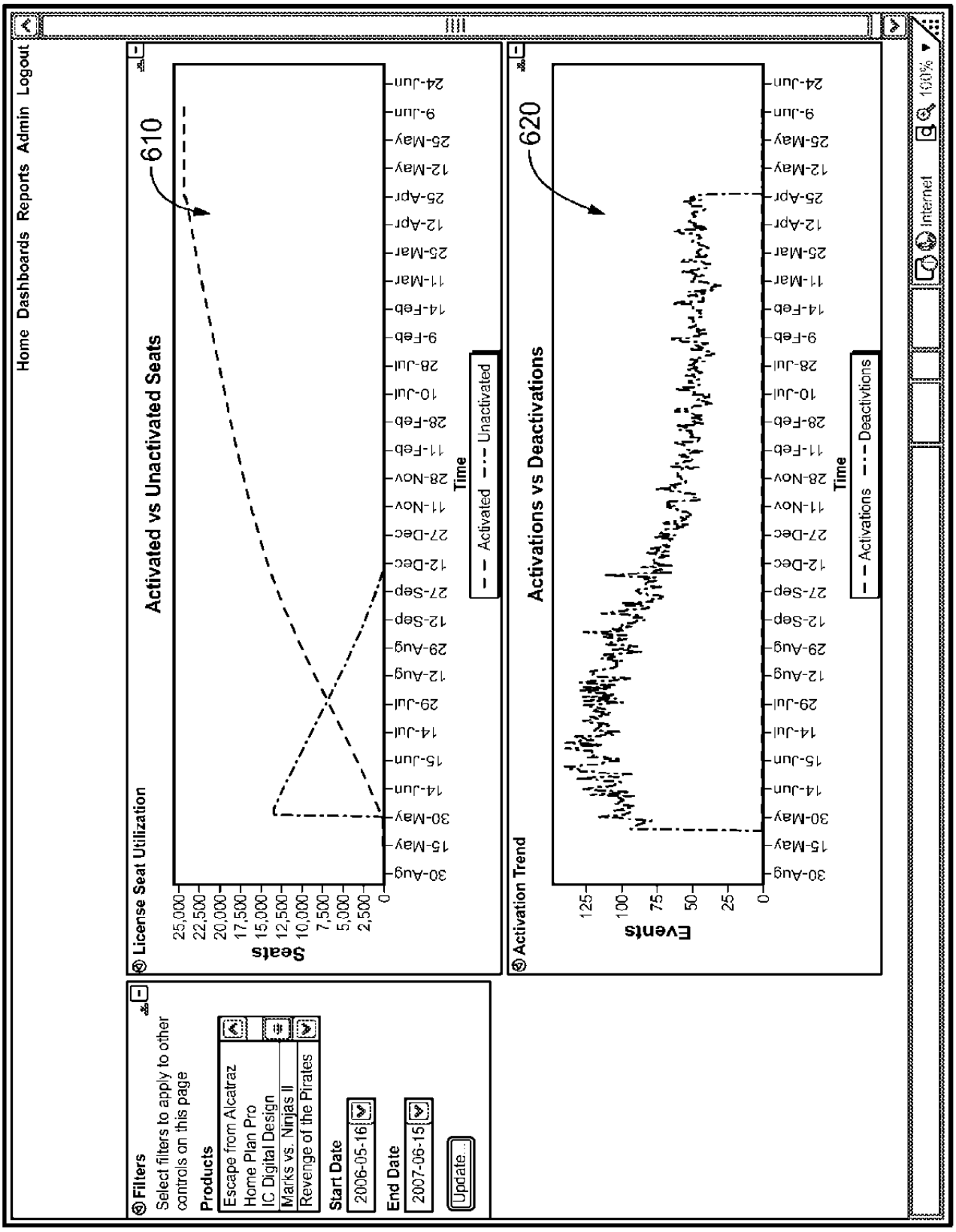

Home Dashboards Reports Admin Logout
Filters
Select filters to apply to other controls on this page
Products
Escape from Alcatraz
Home Plan Pro
IC Digital Design
Marks vs. Ninjas II
Revenge of the Pirates
Start Date
2006-05-16
End Date
2007-06-15
Update
License Seat Utilization
Activated vs Unactivated Seats
610
Seats
Time
Activated
Unactivated
Activation Trend
Activations vs Deactivations
620
Events
Time
Activations
Deactivations
Internet
100%

FIG. 6

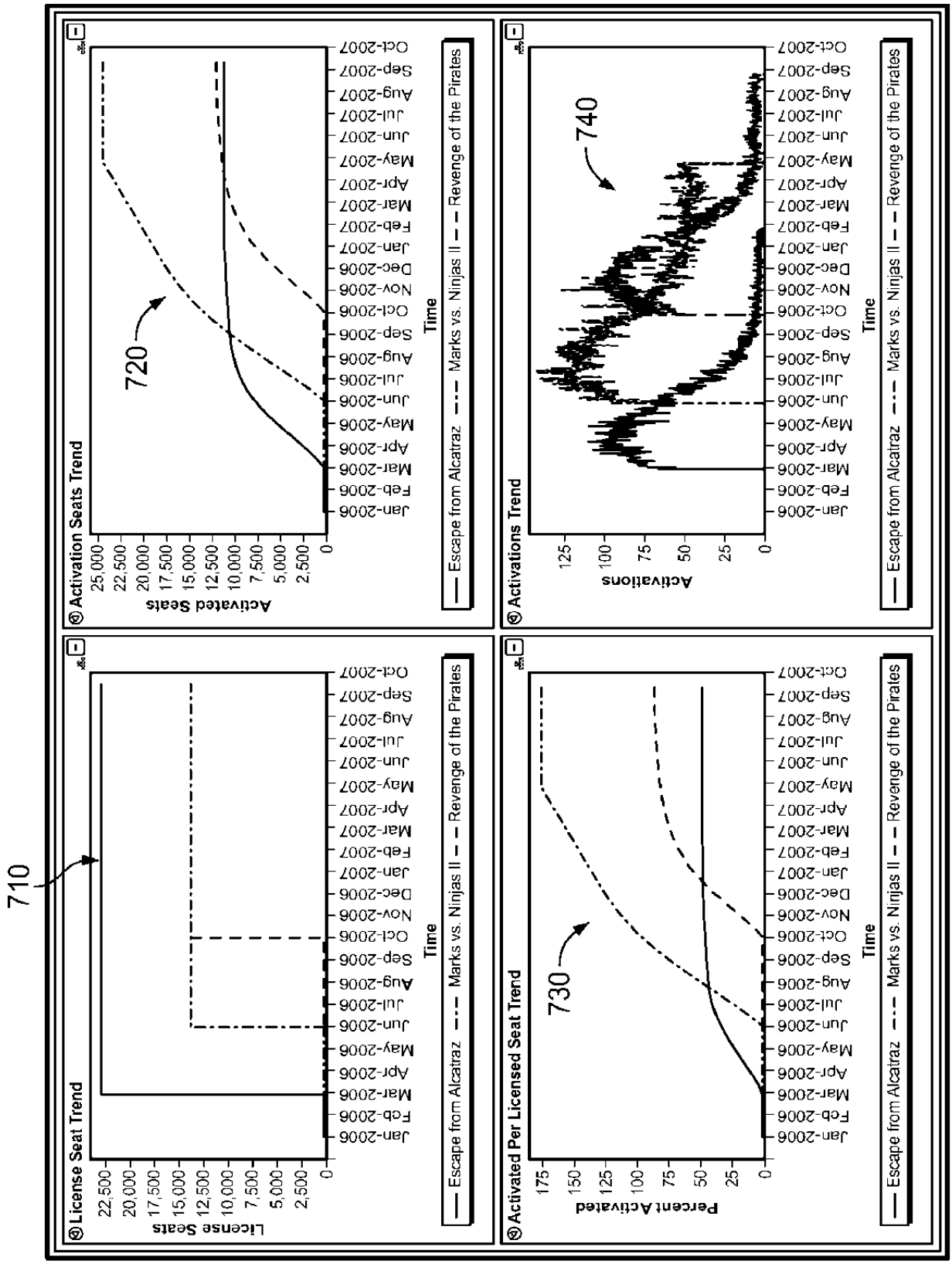

710
License Seat Trend
Activated Seats
Time
Escape from Alcatraz
Marks vs. Ninjas II
Revenge of the Pirates
720
Activation Seats Trend
740
Activations Trend
730
Activated Per Licensed Seat Trend
Percent Activated

FIG. 7

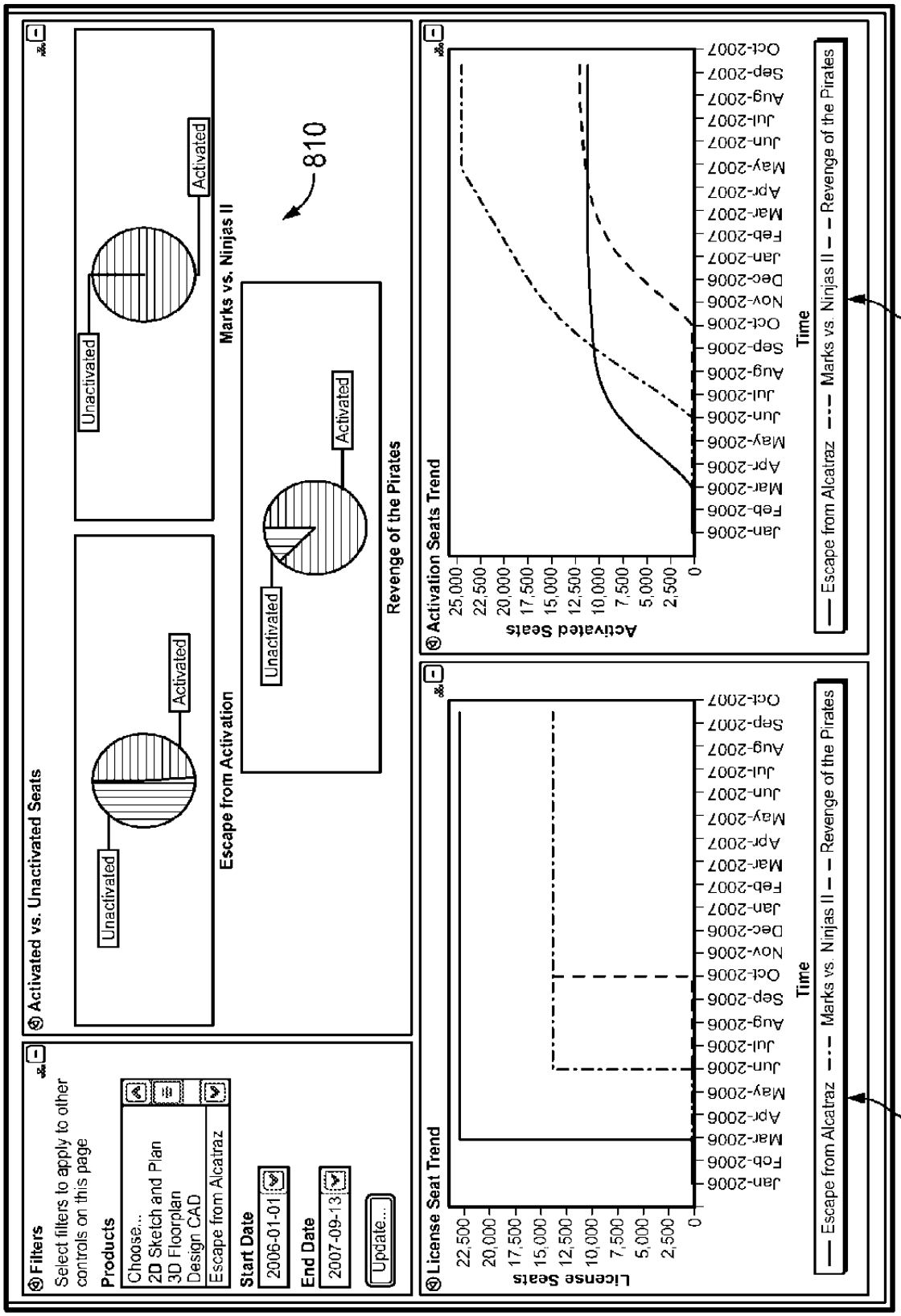

Filters
Select filters to apply to other controls on this page
Products
Choose...
2D Sketch and Plan
3D Floorplan
Design CAD
Escape from Alcatraz
Start Date
2006-01-01
End Date
2007-09-13
Update...
Activated vs. Unactivated Seats
Unactivated
Activated
Escape from Activation
Marks vs. Ninjas II
Revenge of the Pirates
810
License Seat Trend
License Seats
Time
Escape from Alcatraz
Marks vs. Ninjas II
Revenge of the Pirates
820
Activation Seats Trend
Activated Seats
Time
830

FIG. 8

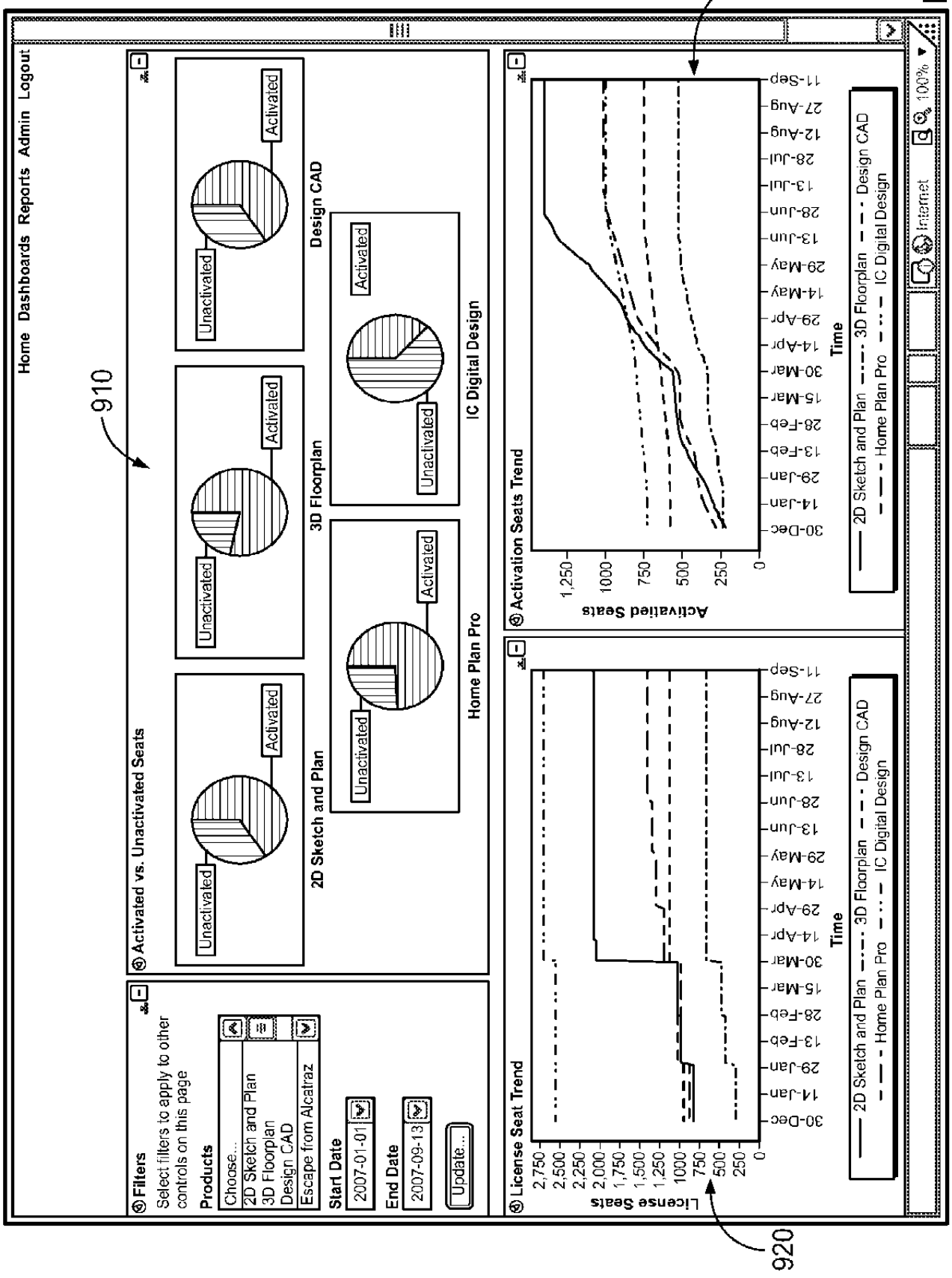

Home Dashboards Reports Admin Logout
910
Activated vs. Unactivated Seats
Unactivated
Activated
2D Sketch and Plan
3D Floorplan
Design CAD
Home Plan Pro
IC Digital Design
Filters
Select filters to apply to other controls on this page
Products
Choose...
2D Sketch and Plan
3D Floorplan
Design CAD
Escape from Alcatraz
Start Date
2007-01-01
End Date
2007-09-13
Update...
License Seat Trend
License Seats
Time
2D Sketch and Plan
3D Floorplan
Design CAD
Home Plan Pro
IC Digital Design
Activation Seats Trend
Activated Seats
930
920
Internet
100%

FIG. 9

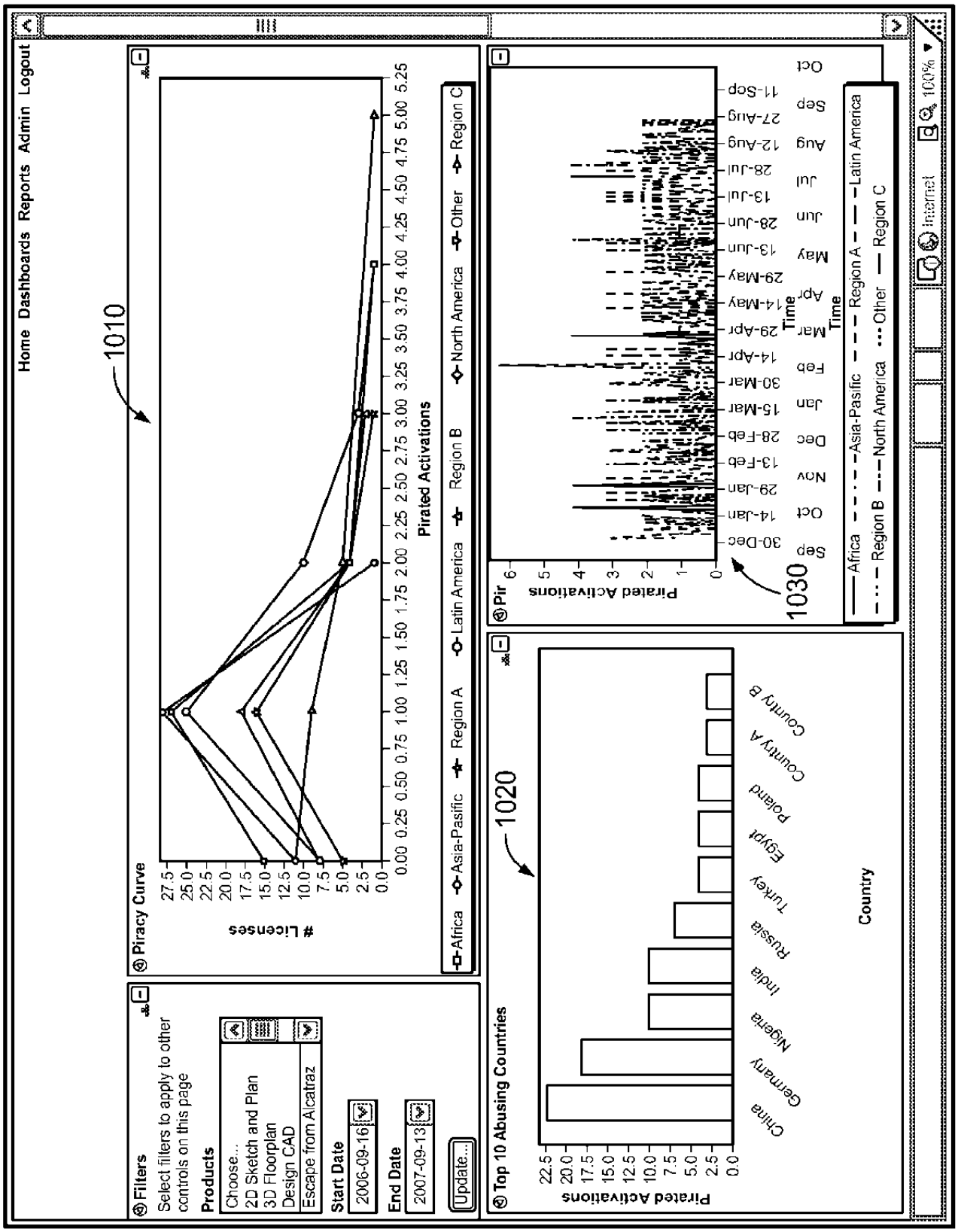

Home Dashboards Reports Admin Logout
Filters
Select filters to apply to other controls on this page
Products
Choose...
2D Sketch and Plan
3D Floorplan
Design CAD
Escape from Alcatraz
Start Date
2006-09-16
End Date
2007-09-13
Update...
Piracy Curve
1010
Licenses
Pirated Activations
Africa
Asia-Pasific
Region A
Latin America
Region B
North America
Other
Region C
Top 10 Abusing Countries
1020
Pirated Activations
Country
China
Germany
Nigeria
India
Russia
Turkey
Egypt
Poland
Country A
Country B
Pir
1030
Time
Africa
Asia-Pasific
Region A
Latin America
Region B
North America
Other
Region C
Internet
100%

FIG. 10

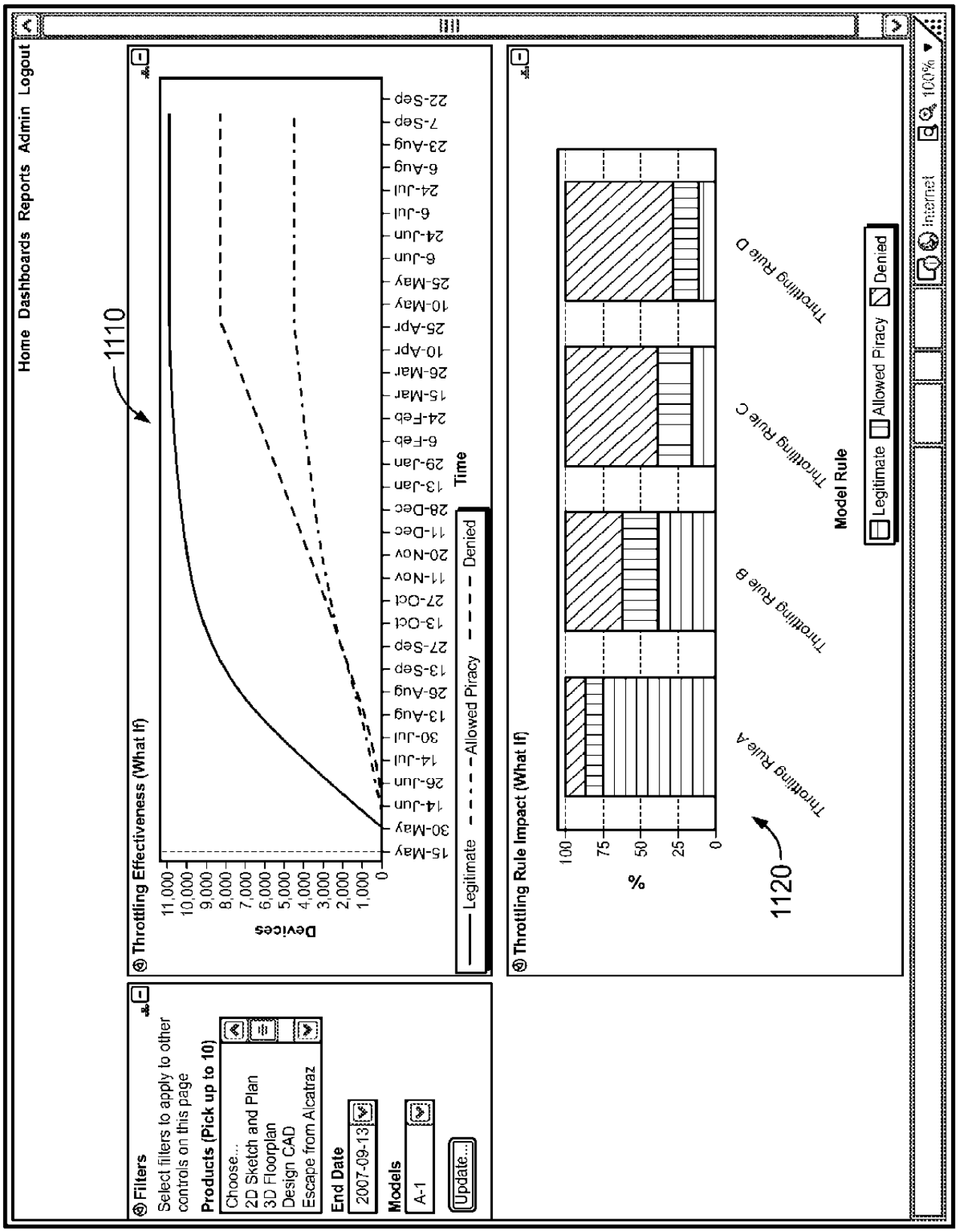

Home Dashboards Reports Admin Logout
Filters
Select filters to apply to other controls on this page
Products (Pick up to 10)
Choose...
2D Sketch and Plan
3D Floorplan
Design CAD
Escape from Alcatraz
End Date
2007-09-13
Models
A-1
Update...
Throttling Effectiveness (What If)
1110
Devices
Time
Legitimate
Allowed Piracy
Denied
Throttling Rule Impact (What If)
%
Model Rule
Throttling Rule A
Throttling Rule B
Throttling Rule C
Throttling Rule D
1120
Legitimate
Allowed Piracy
Denied
Internet
100%

FIG. 11

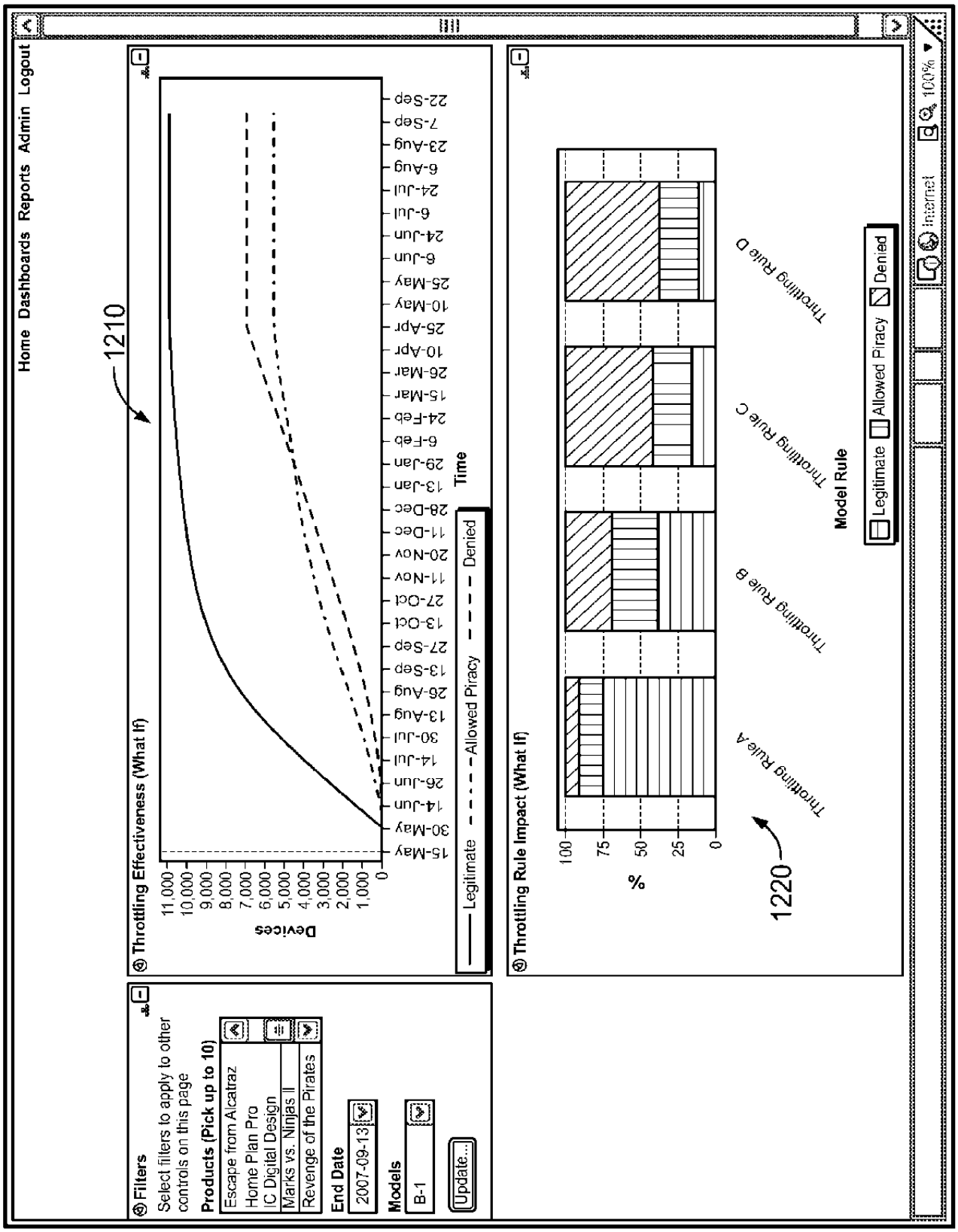

Home Dashboards Reports Admin Logout
Filters
Select filters to apply to other controls on this page
Products (Pick up to 10)
Escape from Alcatraz
Home Plan Pro
IC Digital Design
Marks vs. Ninjas II
Revenge of the Pirates
End Date
2007-09-13
Models
B-1
Update...
Throttling Effectiveness (What If)
1210
Devices
Time
Legitimate
Allowed Piracy
Denied
Throttling Rule Impact (What If)
%
Model Rule
Throttling Rule A
Throttling Rule B
Throttling Rule C
Throttling Rule D
1220
Legitimate
Allowed Piracy
Denied
Internet
100%

FIG. 12

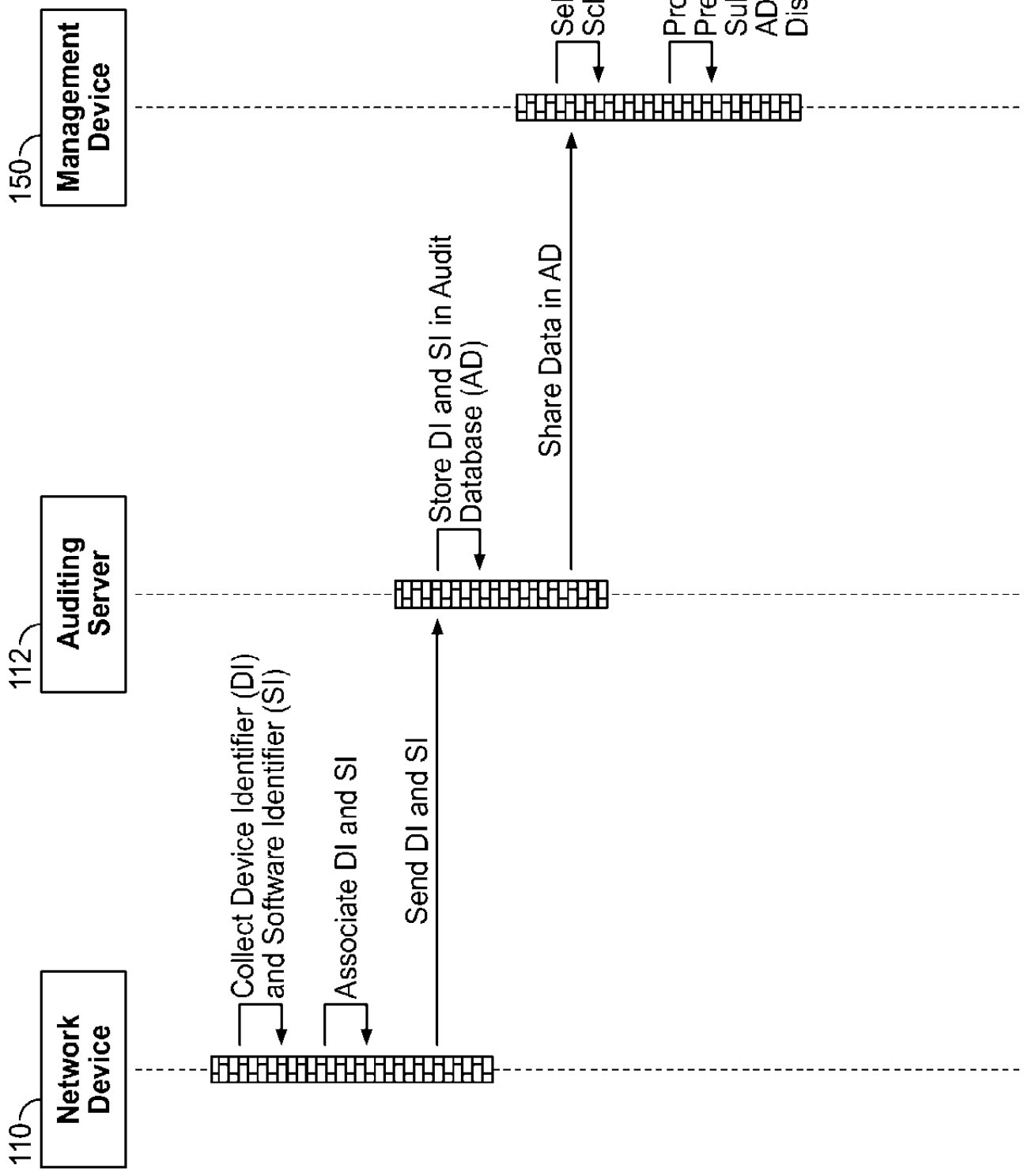

110
Network Device
112
Auditing Server
150
Management Device
Collect Device Identifier (DI) and Software Identifier (SI)
Associate DI and SI
Send DI and SI
Store DI and SI in Audit Database (AD)
Share Data in AD
Select Display Scheme
Process and Present at Least Subset of Data in AD According to Display Scheme

FIG. 13

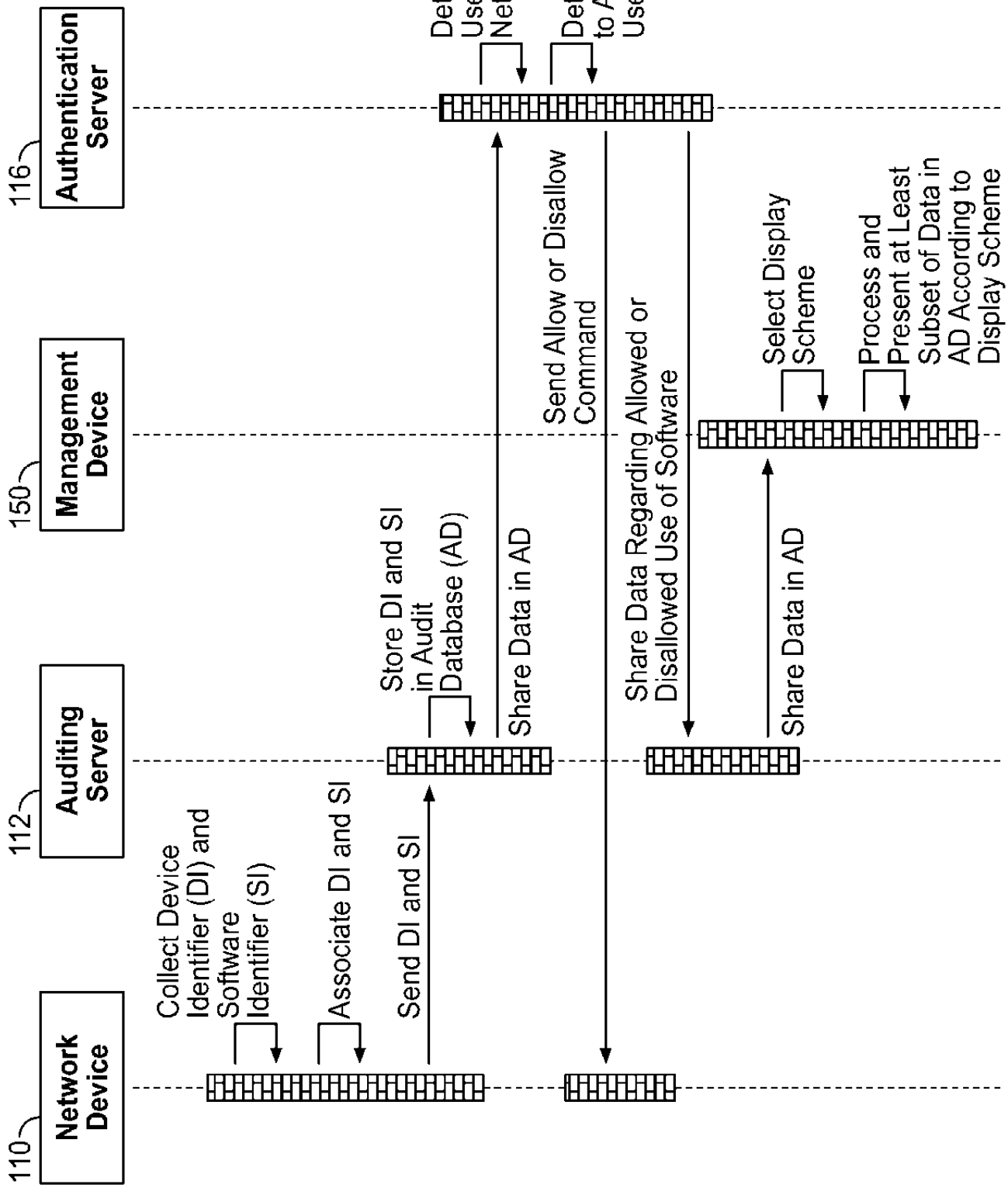

110
Network Device
112
Auditing Server
150
Management Device
116
Authentication Server
Collect Device Identifier (DI) and Software Identifier (SI)
Associate DI and SI
Send DI and SI
Store DI and SI in Audit Database (AD)
Share Data in AD
Detect Attempted Use of Software on Network Device
Determine Whether to Allow or Disallow Use of Software
Send Allow or Disallow Command
Share Data Regarding Allowed or Disallowed Use of Software
Share Data in AD
Select Display Scheme
Process and Present at Least Subset of Data in AD According to Display Scheme

FIG. 14

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 November 2009 (26.11.2009)

PCT

(10) International Publication Number
WO 2009/143115 A1

(51) International Patent Classification:
H04L 29/06 (2006.01) *G08G 1/00* (2006.01)
H04L 29/08 (2006.01)

(21) International Application Number:
PCT/US2009/044467

(22) International Filing Date:
19 May 2009 (19.05.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/055,129 21 May 2008 (21.05.2008) US

(71) Applicant *(for all designated States except US)*: **UNILOC USA, INC.** [US/US]; 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **ETCHEGOYEN, Craig, S.** [US/US]; C/o Uniloc Usa, Inc., 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US). **HARJANTO, Dono** [US/US]; C/o Uniloc Usa, Inc., 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US). **DAVIS, Bradley, C.** [US/US]; C/o Uniloc Usa, Inc., 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz, LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) Title: DEVICE AND METHOD FOR SECURED COMMUNICATION

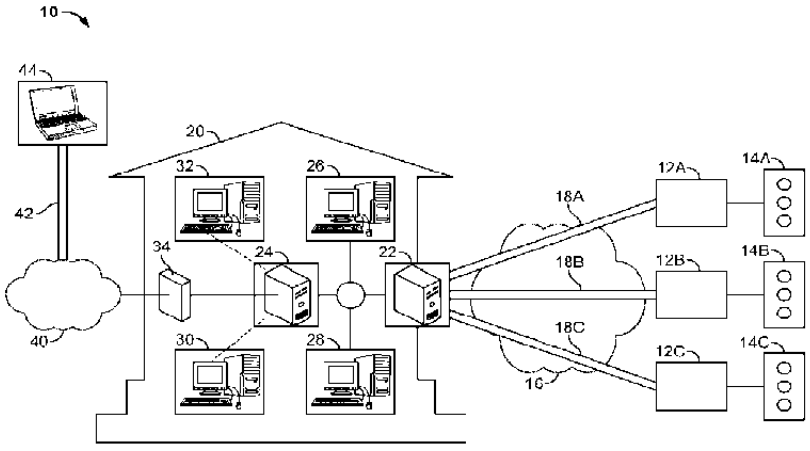


FIG. 1

(57) Abstract: Devices and methods are provided for securing communication between a traffic management center (TMC) (20) and a traffic controller (14) via utilization of a field security device (12). In one embodiment, the field security device (12) transmits a device identifier to the TMC (20) upon being powered up or connected to the traffic controller (14). The device identifier is generally based on a combination of user-configurable and non-user-configurable parameters of the field security device. In response to the TMC authenticating the device identifier, the field security device establishes a secure private network (SPN) (18) between the field security device and the TMC.

WO 2009/143115 A1

DEVICE AND METHOD FOR SECURED COMMUNICATION

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0001] The present invention is directed toward systems for securing communications (e.g., with traffic management centers or the like), and related methods.

DESCRIPTION OF THE RELATED ART

[0002] A trend in the transportation industry is to utilize more cost-effective modes of communication between traffic management centers (TMCs) and traffic controllers located at or near street intersections. The traffic controllers typically comprise, or are otherwise in operative communication with, traffic lights/signals, surveillance cameras, sensors, detectors, etc., one or more of which may be housed in field traffic cabinets at or near the intersections. The traffic controllers and other devices capable of communicating with the TMC often utilize Ethernet and Internet Protocol (IP) based field communications or the like to communicate with and interconnect signalized intersections. A further trend is the utilization of wireless communication protocols for communicating with TMCs and/or traffic controllers.

[0003] With the use of Ethernet and Internet as common platforms of choice in many new transportation management applications, there is an increased possibility for security breaches into such traffic networks. Accordingly, current and future traffic management systems may be vulnerable to attack or abuse from unauthorized intruders, e.g., "hackers" or insiders operating outside their authority, gaining access to the system using stolen or "cracked" security information or using authorized emergency control devices to manipulate traffic signals, etc. Such attacks may endanger public safety, erode public confidence in the traffic control and enforcement systems, and reduce municipal revenues.

[0004] Accordingly, it would be desirable to provide a cost-effective system and method for improving the security of communications for traffic controllers, such as, for example, controllers, detectors, surveillance cameras, uninterruptible power supply systems, and other devices supporting an IP or web based user interface or the like.

SUMMARY OF THE INVENTION

[0005] The following presents a simplified summary of one or more embodiments in order to provide a basic understanding of such embodiments. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some

APPLE EXHIBIT 1102, Page 733 of 1048

concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later.

[0006] In accordance with one or more embodiments and corresponding disclosure thereof, various aspects are described in connection with a field security device/apparatus for providing a secure private network (SPN) between a field traffic controller and a traffic management center (TMC). The field security device may include a first connector (e.g., a receiving port) for interfacing with the field traffic controller; a communication/transceiver module; at least one processor operative coupled to the first connector and the communication module; and a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor.

[0007] The memory module may include executable code for the at least one processor to: access a public network via the communication module; locate an authentication server of the TMC via the public network; and send a device identifier to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the field security device. The memory module may further include executable code for the at least one processor to, in response to the authentication server authenticating the device identifier from the field security device, establish the SPN between the field security device and the TMC. The established SPN may tunnel across one or more segments of the public network.

[0008] In related aspects, the at least one non-user-configurable parameter may comprise at least one of CPU ID, CPU model, CPU manufacturer, and CPU voltage. In the alternative, or in addition, the at least one non-user-configurable parameter may be based on a carbon degradation characteristic of a computer chip. In the alternative, or in addition, the at least one non-user-configurable parameter may be based on a silicone degradation characteristic of a computer chip.

[0009] In further related aspects, the at least one user-configurable parameter may comprise one of hard disk volume name, user name, device name, user password, and hard disk initialization date.

[0010] In yet further related aspects, the device identifier may be generated by utilizing at least one irreversible transformation of the at least one user-configurable and the at least one non-user-configurable parameters. For example, the device identifier may be generated by utilizing a cryptographic hash function on the at least one user-configurable and the at least one non-user-configurable parameters.

[0011] In still further related aspects, the public network may comprise a wireless communication network. The wireless communication network may implement at least one of CDMA and GSM standards. In the alternative, or in addition, the wireless communication network may implement at least one of 802.11a, 802.11b, 802.11g, 802.11n, and 802.11p (Dedicated Short Range Communications) standards.

APPLE EXHIBIT 1102, Page 734 of 1048

[0012] In further related aspects, the traffic controller may comprise a traffic signal, a surveillance camera, etc. The traffic controller may be housed in a field traffic cabinet or the like. The field security device may be adapted to be housed in the field traffic cabinet.

[0013] In accordance with other aspects of the embodiments described herein, there is provided a server (e.g., an authentication server) for providing a SPN between a TMC and a field security device, the field security device being in operative communication with a field traffic controller. For example, the server may include: a communication module adapted to receive a device identifier over a public network from the field security device, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the field security device; at least one processor operatively coupled to the communication module; and a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor.

[0014] The memory module may include executable code for the at least one processor to, in response to the communication module receiving the device identifier from the field security device, access a database of authorized device identifiers corresponding to known field security devices. The memory module may further include executable code for the at least one processor to, in response to the received device identifier matching one of the authorized device identifiers, establish the SPN between the field security device and the TMC.

[0015] In accordance with other aspects of the embodiments described herein, there is provided a network device for securely communicating with a TMC. The network device may include: a communication module adapted to access a public network; at least one processor operatively coupled to the communication module; and a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor.

[0016] The memory module may include executable code for the at least one processor to: access the public network via the communication module; locate an authentication server of the TMC via the public network; and send a device identifier to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the network device. The memory module may further include executable code for the at least one processor to, in response to the authentication server authenticating the device identifier from the network device, establish a SPN between the network device and the TMC.

[0017] In accordance with other aspects of the embodiments described herein, there is provided a method for providing a SPN between a TMC and a device (e.g., a field security device, a network device, etc.). The method may involve: accessing a public network; locating an authentication server of the TMC via the public network; and sending a device identifier for the device to the authentication

APPLE EXHIBIT 1102, Page 735 of 1048

server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the device. The method may further involve, in response to the authentication server authenticating the device identifier, establishing the SPN between the TMC and the device.

[0018] To the accomplishment of the foregoing and related ends, the one or more embodiments comprise the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects of the one or more embodiments. These aspects are indicative, however, of but a few of the various ways in which the principles of various embodiments may be employed and the described embodiments are intended to include all such aspects and their equivalents.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] Figure 1 provides a block diagram of certain components of an exemplary system for secured communication with a traffic management center (TMC).

[0020] Figure 2 illustrates components of an exemplary device identifier.

[0021] Figure 3 illustrates an exemplary embodiment of a network for secure communication between field security devices and an authentication server.

[0022] Figure 4 illustrates one embodiment of an apparatus for providing a secure private network (SPN) between a field traffic controller and a TMC.

[0023] Figure 5 illustrates one embodiment of an apparatus for providing a SPN between a TMC and a field security device.

[0024] Figure 6 illustrates one embodiment of an apparatus for securely communicating with a TMC.

[0025] Figure 7 shows one embodiment of a method for providing a SPN between a TMC and a device (e.g., a field security device, a network device, etc.).

[0026] Figure 8 shows one embodiment of a method for securely communicating with a TMC.

DETAILED DESCRIPTION

[0027] The present invention addresses the need for a system and method for providing secured communication, wireless or otherwise. In the exemplary embodiments described herein, there are presented systems and methods for securing communication between and among traffic controllers and traffic management centers (TMCs), thereby protecting field traffic control systems from the effects of external cyber-threats. It is noted the invention is not limited to securing communication with TMCs; rather, the technology described herein may be used to secure communications between a plurality of locations.

APPLE EXHIBIT 1102, Page 736 of 1048

[0028] Such a system preferably shields traffic management systems against denial-of-service (DOS) attacks and address resolution protocol (ARP) redirecting or spoofing originating from malicious code threats. Such a system preferably implements device-based access control to restrict field-control network access only to authorized PCs or devices. Such a system preferably eliminates transportation network vulnerabilities due to unknown security compliance by private network sharers, and makes it possible to monitor and manage field security configuration and status from the TMC.

[0029] Such a system may include field security devices that send device identifiers to the TMC in an automated manner, and that establish a secured private network between selected system components based at least in part on whether the device identifier is on the list of authorized device identifiers, thereby determining whether a field security device qualifies as a known device. The device identifiers may be based on a combination of user-configurable and non-user-configurable parameters of the field security device. Such authentication and secured communication techniques may be used alone, or in conjunction with other security or authentication measures.

System for Secured Communication with a Traffic Management Center (TMC):

[0030] With reference Figure 1, there is provided an embodiment of a system 10 for securing communication with a TMC 20. Three traffic controllers 14A, 14B, 14C are shown; however, it will be understood that the system 10 may comprise any number of traffic controllers 14. Each traffic controller 14 may comprise a traffic light or signal, a surveillance camera, detectors, sensors, etc., one or more of which may be housed in a field traffic cabinet. In one embodiment, a traffic controller 14 is operatively coupled to a traffic light.

[0031] In the illustrated embodiment, field security devices/apparatuses 12A, 12B, and 12C are operatively coupled to the traffic controllers 14A, 14B, and 14C, respectively. Each field security device 12 may function as a security appliance that creates a secure, virtual-network layer connection between a given traffic controller 14 (coupled to the given field security device 12) and the TMC 20. As will be explained in further detail below, the field security devices 12A, 12B, 12C and authentication server 22 at the TMC 20 utilize device recognition technology to establish secure private networks 18A, 18B, and 18C between the TMC 20 and the field security devices 12A, 12B, and 12C, respectively.

[0032] Each secure private network (SPN) 18 may tunnel across one or more segments of a public network 16. The public network 16 (as well as public network 40) may comprise one or more public portions of the Internet (e.g., 802.3, DSL, cable, Ethernet, etc.). The public networks 16, 40 may comprise a wireless communication network, such as, for example, CDMA, GSM, etc. The public networks 16, 40 may comprise a wireless local area network (WLAN), such as, for example, 802.11a, 802.11b, 802.11g, 802.11n, 802.11p, etc. It is noted that the public networks 16, 40 may comprise any

APPLE EXHIBIT 1102, Page 737 of 1048

communication network, wired or wireless, utilizing any known standards, such as, for example, wide area networks (WANs), campus area networks (CANs), metropolitan area networks (MANs), wireless application protocol (WAP), etc. In the alternative, or in addition, the SPN 18 may tunnel across a traffic control network, a portion of which is public.

[0033] The TMC 20 may include an authentication server 22 that is in operative communication with one or more workstations 26, 28, such as, for example, via a node/switch in between the authentication server 22 and a general server 24 (i.e., not an authentication server). The TMC may include a firewall 34 between the general server 24 and the public network 40, and thereby add another layer of protection for communications to and from the TMC 20. In the alternative, or in addition, the TMC may comprise a firewall (not shown) between the authentication server 22 and the public network 16. In the alternative, or in addition, one or more authentication servers and/or workstations operatively coupled to the authentication servers may be located outside of the TMC, such as, for example, at a remote site.

[0034] The system 10 may include a network device 44, such as, for example, laptop computer, tablet computer, PDA, mobile phone or device, etc. The network device 44 may comprise, for example, a field technician's laptop for troubleshooting traffic controllers 14A, 14B, and 14C. Device 44 needs to connect to authentication server 22 in order to establish an SPN 42 between a user of the network device 44 (e.g., a field engineer) and the TMC 20. In one embodiment, the device 44 bypasses the firewall 34 via a VPN soft-server on the server 24. Once the authentication server 22 authorizes device 44, the SPN 42 is established. The SPN 42 may essentially function as a tunnel within the VPN soft-server, and therefore may be analogous to a tunnel within a tunnel. In another embodiment (not shown), a field security device 12 may acts as a proxy for a network device 44 whose user wishes to access the network, when the network device 44 is connected behind the field security device 12.

[0035] It is noted that SPN 18 has the ability to provide a star topology whereby the field security devices 12A, 12B, 12C may communicate with each other, through server 22, thereby providing a way for traffic controllers 14A, 14B, and 14C to communicate with each other as well. For example, in one embodiment, SPN 18 may be configured to that field security devices 12A, 12B, 12C can only communicate with server 22 (and workstations 26, 28). Such an embodiment would normally be applicable to an Enterprise Server deployment, thereby preventing a TMC for one city from affecting critical assets of a TMC of another city.

[0036] Figure 3 illustrates an exemplary embodiment of a network for securing communication between the field security devices 12A, 12B and the authentication server 22. Portions 15A, 15B, and 23 of the shown network represent the secured portions of the network. Portion 15A may include a field security device 12A in operative communication with a traffic signal/light and/or surveillance/video camera(s). Portion 15B may include a field security device 12B in operative

APPLE EXHIBIT 1102, Page 738 of 1048

communication with an Advanced Traffic Management Systems (ATMS) client, which is in operative communication with a traffic controller. Portion 23 may include an authentication server 22 in operative communications with other servers, such as, for example, an ATMS server or a streaming server, via an Ethernet switch or the like. The network device 44 (e.g., laptop computer) may also be authenticated via the server 22 for access to the field security devices 12A, 12B.

Device Identifiers:

[0037] As noted above, the field security devices 12A, 12B, 12C and the authentication servers 22, 24, as well as the network device 44, may utilize device recognition technology to establish SPNs 18A, 18B, and 18C. For example, each field security device 12 may be adapted to transmit self-identification information to the authentication server 22 upon being powered up in the field. The self-identification information or device identifier generally comprises information that is expected to be unique for the field security device 12. For example, the device identifier for a given field security device 12 may comprise a serial number and/or location information (e.g., an IP address, geo-location code, etc.).

[0038] The device identifier is preferably generated from machine parameters of the field security device 12, such as, for example, hard disk volume name, user name, device name, user password, hard disk initialization date, etc. The machine parameters may relate to the platform on which the web browser runs, such as, for example, CPU number, or unique parameters associated with the firmware in use. The machine parameters may also include system configuration information, such as amount of memory, type of processor, software or operating system serial number, etc. The device identifier generated from the machine parameters may include the field security device's IP address and/or other geo-location code to add another layer of specificity to field security device's unique identifier. In the alternative, or in addition, the device identifier may comprise a randomly generated and assigned number that is unique for the field security device 12.

[0039] In one embodiment, the device identifier for the field security device 12 is generated and stored in the field security device's memory before the field security device 12 is deployed into the field. In another embodiment, the device identifier, or a portion thereof, is generated after the field security device 12 is deployed and/or powered on in the field.

[0040] It is noted that an application running on the field security device 12 or otherwise having access to the field security device's hardware and file system may generate a unique device identifier using a process that operates on data indicative of the field security device's configuration and hardware. The device identifier may be generated using a combination of user-configurable and non-user-configurable machine parameters as input to a process that results in the device identifier, which may be expressed in digital data as a binary number. Each machine parameter may include data

APPLE EXHIBIT 1102, Page 739 of 1048

determined by a hardware component, software component, or data component specific to the device that the unique identifier pertains to. Machine parameters may be selected based on the target device system configuration such that the resulting device identifier has a very high probability (e.g., greater than 99.999%) of being unique to the target device. In addition, the machine parameters may be selected such that the device identifier includes at least a stable unique portion up to and including the entire identifier that has a very high probability of remaining unchanged during normal operation of the target device. Thus, the resulting device identifier should be highly specific, unique, reproducible and stable as a result of properly selecting the machine parameters.

[0041] The application for generating the device identifier may also operate on the collected parameters with one or more algorithms to generate the device identifier. This process may include at least one irreversible transformation, such as, for example, a cryptographic hash function, such that the input machine parameters cannot be derived from the resulting device identifier. Each device identifier, to a very high degree of certainty, cannot be generated except by the suitably configured application operating or otherwise having had access to the same field security device for which the device identifier was first generated. Conversely, each identifier, again to a very high degree of certainty, can be successfully reproduced by the suitably configured application operating or otherwise having access to the same field security device on which the identifier was first generated.

[0042] The application may operate by performing a system scan to determine a present configuration of the field security device. The application may then select the machine parameters to be used as input for generating the unique device identifier. Selection of parameters may vary depending on the system configuration. Once the parameters are selected, the application may generate the identifier.

[0043] Further, generating the device identifier may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties as well as a variety of additional parameters such as uniquely generated hashes and time sensitive values. Physical device parameters available for sampling may include, for example, unique manufacturer characteristics, carbon and silicone degradation and small device failures.

[0044] The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as heat and additional stresses placed on a chip during the sampling process allows for each and every benchmark to reproduce the expected values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms.

APPLE EXHIBIT 1102, Page 740 of 1048

[0045] In addition to the chip benchmarking and degradation measurements, the process for generating a device identifier may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

[0046] Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further extended by the use of complex encryption algorithms to convolute the device identifier values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

[0047] The device identifier may be generated by utilizing machine parameters associated with one or more of the following: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

[0048] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

[0049] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: memory model; memory slots; memory total; and memory details.

[0050] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: video model; video details; display model; display details; audio model; and audio details.

[0051] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: network model; network address; Bluetooth address; BlackBox model; BlackBox serial; BlackBox details; BlackBox damage map; BlackBox volume name; NetStore details; and NetStore volume name.

[0052] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

[0053] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

[0054] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

[0055] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller.

[0056] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

[0057] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

[0058] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

[0059] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD.

[0060] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

[0061] In one example, the device identifier may also be generated by utilizing machine parameters associated with one or more of the following: machine model, processor model, processor details, processor speed, memory model, memory total, network model of each Ethernet interface, network MAC address of each Ethernet interface, BlackBox Model, BlackBox Serial (e.g., using Dallas Silicone Serial DS-2401 chipset or the like), OS install date, nonce value, and nonce time of day.

[0062] With reference to Figure 2, in one exemplary embodiment, a device identifier 50 may include two components – namely, a variable key portion 52 and a system key portion 54. The variable key portion 52 may be generated by reference to a variable platform parameter, such as via reference to system time information, although other parameters which are variable may be utilized in other embodiments. The system key portion 54 may include the above described parameters expected to be unique to the field security device 12, such as, for example, hard disk volume name, user name, computer name, user password, hard disk initialization date, or combinations thereof. Portions 52 and/or 54 may be combined with the IP address and/or other platform parameters of the field security device 12. It is noted that device identifiers, or portions thereof, may be encrypted to add an additional layer of specificity and security.

[0063] It is noted that device identifiers may be generated for the network device 44, authentication server 22, and workstations 26, 28 in the same manner as described above for the field

APPLE EXHIBIT 1102, Page 742 of 1048

security devices 12. With reference to the exemplary embodiment of Figure 1, only server 22, workstations 26 and 28, and laptop 44 have been authenticated.

Secure Private Networks (SPNs):

[0064] With continued reference to the exemplary embodiment of Figure 1, it is noted that each field security device 12 is generally adapted to transmit its device identifier back to the TMC 20. Upon being powered on and/or connected to the traffic controller 14, the field security device 12 preferably accesses an available public network 16, locates or identifies an authentication server 22 at the TMC 20, and then establishes a connection with the authentication server 22. Upon establishing a connection with the authentication server 22, the field security device 12 may transmit its device identifier to the authentication server 22. The device identifier is preferably encrypted prior to being transmitted by the field security device 12 over to the public network 16, and then decrypted when received by the authentication server 22.

[0065] In response to receiving the device identifier from a given field security device 12, the authentication server 22 may access a database of authorized device identifiers corresponding to known devices that are authorized to establish a SPN 18 with the TMC 20. The database may be located at the TMC 20, such as, for example, on one of the servers 22, 24 and/or workstations 26, 28, 30, 32. The database is preferably located on server 22 and/or workstations 26, 28. In the alternative, or in addition, the database may be located on a server or machine that is not located at the TMC 20, yet is accessible by server 22.

[0066] When the device identifier from the field security device 12 matches one of the authorized device identifiers in the database, the authentication server 22 and the field security device establish a SPN with each other, and thereby create an SPN 18 between the TMC 20 and the traffic controller 14. The SPN 18 generally tunnels across one or more segments of the public network 16 to provide a secure channel of communication between the TMC 20 and the traffic controller 14.

[0067] The SPN 18 may be established according to any known technique, such as, for example, via the creation of virtual private networks (VPNs), in which some of the links between nodes are carried by open connections or virtual circuits in a larger network, such as, for example, public portions of the Internet. Link-layer protocols of the virtual network may be tunneled through the larger network.

[0068] The field security devices/appliances 12 may get serialized labeling at the manufacturing facility, similar to copies of software for authenticity and tracking/history. For plug-and-play in the field, the appliances may first be connected directly to the authentication server, which may be done at a field tech's offices before initial server deployment, and the IP address of the server may be stored. The device fingerprint may also be taken at this time. The deployment address for each appliance may

APPLE EXHIBIT 1102, Page 743 of 1048

be entered into the server, such as for use in automated geographic mapping of appliance locations. In the alternative, the appliances 12 may be configured from the field using an authenticated PC connected to the appliance.

[0069] It is noted that one or more SPNs 42 may be established between the authentication server 22 and any network devices 44 in the same manner as described above for the field security devices 12. The SPN 42 may tunnel across one or more segments of the public network 42 to provide a secure channel of communication between the TMC 20.

[0070] In one embodiment, the field security device 12 sends its device identifier or machine fingerprint to the authentication server 22. When the server 22 verifies that the device identifier corresponds to a known or authorized device, the server sends an authentication/verification signal to the device 12. The device 12 then sends a certificate or public key to the server 22 to establish the SPN 18. The server 22 uses a private key to check the certificate. The server 22 then sends a server certificate or public key back to the device 12 to establish the SPN 18.

Field Security Device:

[0071] The field security device 12 may also be referred to as a field appliance and creates a secure, virtual-network layer connection between the TMC 20 over otherwise public communication networks, including or utilizing the Internet, Ethernet, aud wireless technologies. The field security device 12 may be operatively coupled to controllers, sensors, detectors, surveillance cameras, uninterruptible power supply (UPS) systems, or other devices supporting an IP or web based user interface.

[0072] In accordance with one aspect of the embodiments described herein, there is provided a field security device 12 for providing a SPN 18 between a field traffic controller 14 and a TMC 20, comprising: a first connector for interfacing with the field traffic controller 14; a communication module; a processor module operatively coupled to the first connector and the communication module; and a memory module operatively coupled to the processor module. In one embodiment, the memory module comprises executable code for the processor module to: (a) access a public network 16 or traffic control network via the communication module; (b) locate and/or connect with an authentication server 22 of the TMC 20 via the public network 16; and (c) send a device identifier to the authentication server 22 via the communication module, the device identifier being based on a combination of both user-configurable and non-user-configurable parameters of the field security device 12; and (d) in response to the authentication server 22 authenticating the device identifier from the field security device 12, establish the SPN 18 between the field security device 12 and the TMC 20, wherein the established SPN 18 tunnels across at least one segment of the public network 16.

APPLE EXHIBIT 1102, Page 744 of 1048

[0073] The processor module of the field security device 12 may comprise one or more processors, such as, for example, a Motorola MPC8321EEC Microprocessor (333 MHz core processor speed, 32MB flash memory, 64MB DDR2 memory, 32 MBs VPN throughput) or the like. The first connector of the field security device 12 may comprise a receiving port or the like (e.g., 1WAN, 4WAN, RJ45, 10/100 Mbit/s Ethernet, etc.).

[0074] The field security device 12 is preferably adapted for easy plug-and-play field installation, with no field PC required, no device configuration required in the field, and no passwords or keys required to manage. In essence, when the field security device 12 is connected or powered up, it preferably "phones home" to an authentication server and establishes its own device-locked point-to-point SPN 18.

[0075] The memory module of the field security device 12 may further comprise executable code for the processor module to detect network intrusions, determine locations of the intrusions, and notify the TMC 20. The field security device 12 may be adapted to continuously or periodically verify its operational status via one or more authentication servers at the TMC 20. The field security device 12 is preferably cross-platform compatible with any operating system and field control hardware. The field security device 12 is preferably adapted to be NEMA TS2 compliant.

[0076] The field security device 12 may be adapted to connect to any known network routers, switches, and/or firewall security devices. The field security device 12 may be adapted to perform a self-test at startup. The field security device 12 may comprise one or more LED indicators to power and communications link status, or activities status.

[0077] The field security device 12 may be field hardened for use inside or outside of the field traffic cabinet. The field security device 12 may be shelf mountable for easy in-cabinet placement with optional DIN rail or sidewall mounting. The field security device 12 may be adapted to defined environmental conditions, such as, for example, -29°F to +165°F (-34°C to +74°C), 0 to 95% relative humidity.

[0078] It is noted that the security device/appliance 12 may be adapted to access, learn, or otherwise determine the MAC IDs of traffic controllers 14 or other devices operatively coupled with (e.g., plugged into) the device 12. Further, the device 12 may utilize the learned MAC IDs to establish bi-directional security with such traffic controllers 14, thereby prohibiting unknown/unauthorized network devices from connecting to the secured network via the device 12. For example, the device 12 may comprise a memory module storing executable code for a processor module to access and store into the memory module MAC IDs of those traffic controllers 14 connected to the device 12. The executable code may further comprise instructions for the processor module to relay the MAC ID or derivations thereof to the TMC 20 to verify whether the MAC ID or derivation thereof corresponds to a known or authorized device. In response to the authentication server 22 of the TMC 20 authenticating

APPLE EXHIBIT 1102, Page 745 of 1048

the MAC ID or derivation thereof, the device 12 may allow the traffic controller 14 to communicate via a SPN 18 between the TMC 20 and the device 12. Otherwise, the traffic controller 14 is blocked or prohibited from communicating with the TMC 20 via SPN 18.

Authentication Server:

[0079] In accordance with another aspect of the embodiments described herein, there is provided an authentication server 22 for providing a SPN 18 between a TMC 20 and a field security device 12, the field security device 12 being in operative communication with a field traffic controller 14, comprising: a communication module adapted to receive a device identifier over a public network 16 from the field security device 12, the device identifier being based on a combination of both user-configurable and non-user-configurable parameters of the field security device 12; a processor module operatively coupled to the communication module; and a memory module operatively coupled to the processor module. In one embodiment, the memory module comprises executable code for the processor module to: (a) in response to the communication module receiving the device identifier from the field security device 12, access a database of authorized device identifiers corresponding to known field security devices; and (b) in response to the received device identifier matching one of the authorized device identifiers, establish the SPN 18 between the field security device 12 and the TMC 20, wherein the established SPN 18 tunnels across at least one segment of the public network 16.

[0080] When multiple field security devices 12A, 12B, 12C establish SPNs 18A, 18B, 18C with a given authentication server 22, a point-to-multipoint SPN may be established between the TMC 20 with each field traffic cabinet in which the field security devices 12A, 12B, 12C may be located.

[0081] The authentication server 22 alone or in conjunction with the workstations 26, 28 and/or other components of the TMC 20, may allocate, manage, and control the field security devices 12 and/or PC clients from a single location, such as, for example, the TMC 20. The TMC 20 and components thereof make it possible to gain real-time insight into the status of the field security devices 12 and network devices 44 (e.g., a PC client or the like) participating in the secured network or system 10.

[0082] Further, the components of the system 10 described herein make it possible to define and receive instant status reports and updates regarding any changes to the secured network, and to receive alerts regarding any unauthorized access attempts by unauthorized devices. The notifications or alerts at the server 22 regarding such unauthorized connection attempts may include information regarding the unauthorized device, the time of the attempted access, the geo-location of the unauthorized device or point of attempted access, etc.

[0083] In accordance with another aspect of the embodiments described herein, there is provided an enterprise server that may connect or be in operative communication with a plurality of "child"

APPLE EXHIBIT 1102, Page 746 of 1048

authentication servers. The child authentication servers may be located at multiple TMCs. The master or enterprise server may be adapted to allow authorized field technicians to have access to the multiple TMCs via one enterprise server or service provider. Such technicians may have simultaneous access to the TMCs via the enterprise server. In the alternative, or in addition, each of the authorized technicians may have the ability to simultaneously access one or more of the field security devices that are in operative communicative communication with the TMCs via the enterprise server.

[0084] In accordance with yet another aspect of the embodiments described herein, there is provided a system wherein the authentication server 22 sends its own device identifier or machine fingerprint to the field security device 12 for mutual or two-way authentication. In addition to having the server 22 verify and authenticate the device 12's identifier, the device 12 also verifies and authenticates the server 22's identifier, before a SPN 18 is established between the device 12 and the server 22. Such a system would provide a more robust scheme for securing communication with the TMC 20. In the alternative, or in addition, the authentication server 22 may be adapted to sends its device identifier to a network device 44 (explained in further detail below) for mutual authentication between the server 22 and the device 44, without which the SPN 42 may not be established.

Network Device:

[0085] In accordance with another aspect of the embodiments described herein, there is provided a network device 44 (e.g., a laptop computer or PDA) for securely communicating with a TMC 20, comprising: a communication module adapted to access a public network; a processor module operatively coupled to the communication module; and a memory module operatively coupled to the processor module. In one embodiment, the memory module comprises executable code for the processor module to: (a) access the public network 40 via the communication module; (b) locate and/or connect with an authentication server 22 of the TMC 20 via the public network 40; (c) send a device identifier to the authentication server 22 via the communication module, the device identifier being based on a combination of both user-configurable and non-user-configurable parameters of the network device 44; and (d) in response to the authentication server 22 authenticating the device identifier from the network device 44, establish a SPN 42 between the network device 44 and the TMC 20, wherein the established SPN 42 tunnels across at least one segment of the public network 40.

[0086] The network device 44, as well as the workstations 26, 28, may comprise client software for device fingerprinting and registration on SPNs or the like. It is noted that the network device 44 may comprise a client software that designates the network device 44 as a field technician device, as opposed to TMC workstation devices 26 and 28, which may have licensing provisions that are different from other network devices. The client software on device 44 may comprise instructions for its host network device to: access a public network; locate an authentication server 22 of the TMC 20 via the

APPLE EXHIBIT 1102, Page 747 of 1048

public network 40; send a device identifier to the authentication server 22, wherein the device identifier is based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the host network device. The client software may further comprise instructions for its host network device to: in response to the authentication server 22 authenticating the device identifier, establish a SPN 42 with the TMC 20, wherein the established SPN 42 tunnels across at least one segment of the public network 40.

Method for Providing a SPN:

[0087] In accordance with another aspect of the embodiments descried herein, there is provided a method for providing a SPN between a device (e.g., field security device 12 or network device 44) and a TMC, comprising: accessing a public network (e.g., networks 16 or 40); and locating and/or connecting with an authentication server (e.g., server 22) of the TMC via the public network.

[0088] The method may further comprise sending a device identifier for the device to the authentication server via the communication module, the device identifier being based on a combination of both user-configurable and non-user-configurable parameters of the network appliance.

[0089] The method may further comprise, in response to the authentication server authenticating the device identifier, establishing the SPN between the TMC and the device. The established SPN preferably tunnels across at least one segment of the public network.

Apparatuses for Providing SPNs:

[0090] In accordance with one or more aspects of the embodiments described herein, there are provided devices and apparatuses (e.g., field security devices or the like) for providing a SPN between a field traffic controller and a TMC. With reference to Figure 4, there is provided an exemplary apparatus 400 that may be configured as either a computing device, or as a processor or similar device for use within a computing device. As illustrated, apparatus 400 may comprise a means 420 for accessing a public network (e.g., the public Internet), such as, for example, via a communication/transceiver module 406, adapted for wireless communication or otherwise. Apparatus 400 may comprise: a means 430 for locating an authentication server of a TMC or the like via the public network; and a means 440 for sending a device identifier to the authentication server via the communication module 406 or the like. The device identifier may be based on a combination of at least one user-configurable parameter and at least one non-user configurable parameter of the apparatus. In this way, the device identifier is unique and no device will share the same identifier. The apparatus 400 may comprise a means 450 for establishing a SPN between itself and the TMC, in response to the authentication server authenticating the device identifier from the apparatus 400. The established SPN may tunnel across one or more segments of the public network.

APPLE EXHIBIT 1102, Page 748 of 1048

[0091] In related aspects, the at least one non-user-configurable parameter may comprise at least one of CPU ID, CPU model, CPU manufacturer, and CPU voltage for apparatus 400. In the alternative, or in addition, the at least one non-user-configurable parameter may be based on a carbon degradation characteristic of a computer chip of apparatus 400. In the alternative, or in addition, the at least one non-user-configurable parameter may be based on a silicone degradation characteristic of a computer chip of apparatus 400.

[0092] In further related aspects, the at least one user-configurable parameter may comprise one of hard disk volume name, user name, device name, user password, and hard disk initialization date for apparatus 400.

[0093] In yet further related aspects, the device identifier may be generated by utilizing at least one irreversible transformation of the at least one user-configurable and the at least one non-user-configurable parameters of apparatus 400. For example, the device identifier may be generated by utilizing a cryptographic hash function on the at least one user-configurable and the at least one non-user-configurable parameters of apparatus 400.

[0094] In still further related aspects, the public network may comprise a wireless communication network. The wireless communication network may implement at least one of CDMA and GSM standards. In the alternative, or in addition, the wireless communication network may implement at least one of 802.11a, 802.11b, 802.11g, 802.11n, and 802.11p standards.

[0095] In further related aspects, the traffic controller may comprise a traffic signal, a surveillance camera, etc. The traffic controller may be housed in a field traffic cabinet or the like. The field security device may be adapted to be housed in the field traffic cabinet.

[0096] It is noted that apparatus 400 may optionally include a processor module 408 having at least one processor, in the case of apparatus 400 configured as computing device, rather than as a processor. Processor 408, in such case, may be in operative communication with means 420-450, and components thereof, via a bus 402 or similar communication coupling. Processor 408 may effect initiation and scheduling of the processes or functions performed by means 420 450, and components thereof.

[0097] In related aspects, apparatus 400 may include a connector 404 (e.g., a receiving port) for interfacing with the field traffic controller. Apparatus 400 may include a communication module 406 for communicating with means 420-450. A stand alone receiver and/or stand alone transmitter may be used in lien of or in conjunction with communication module 406.

[0098] In further related aspects, apparatus 400 may optionally include a means for storing information, such as, for example, a memory device/module 410. Computer readable medium or memory device/module 410 may be operatively coupled to the other components of apparatus 400 via bus 402 or the like. The computer readable medium or memory device 410 may be adapted to store

computer readable instructions and data for effecting the processes and behavior of means 420-450, and components thereof, or processor 408 (in the case of apparatus 400 configured as a computing device) or the methods disclosed herein.

[0099] In yet further related aspects, the memory module 410 may optionally include executable code for the processor module 408 to provide a SPN between a TMC and a device (e.g., a field traffic controller) by: (a) accessing a public network; (b) locating an authentication server of the TMC via the public network; (c) sending a device identifier for the device to the authentication server; and (d) establishing the SPN between the TMC and the device, in response to the authentication server authenticating the device identifier. One or more of steps (a)-(d) may be performed by processor module 408 in lieu of or in conjunction with the means 420-450 described above.

[0100] In accordance with one or more aspects of the embodiments described herein, Figure 5 illustrates an exemplary apparatus 500 (e.g., an authentication server) for providing a secure private network between a TMC and a field security device, the field security device being in operative communication with a field traffic controller. Apparatus 500 may include a communication module 506 adapted to receive a device identifier over a public network from the field security device, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the field security device. Apparatus 500 may include at least one processor operatively coupled to the communication module 506, and a memory module 510 operatively coupled to the at least one processor and comprising executable code for the at least one processor.

[0101] Apparatus 500 may comprise a means 520 for accessing a database of authorized device identifiers corresponding to known field security devices, in response to communication module 506 receiving the device identifier from the field security device. Means 520 may comprise a means (not illustrated) whereby registered and authorized field devices may access the SPN. Apparatus 500 may further comprise a means 530 for establishing the SPN between the field security device and the TMC, in response to the received device identifier matching one of the authorized device identifiers. In related aspects, apparatus 500 may comprise an authentication server that is optionally located at the TMC.

[0102] It is noted that apparatus 500 may optionally include a processor module 508 having at least one processor, in the case of apparatus 500 configured as computing device, rather than as a processor. Processor 508, in such case, may be in operative communication with means 520-530, and components thereof, via a bus 502 or similar communication coupling. Processor 508 may effect initiation and scheduling of the processes or functions performed by means 520-530, and components thereof.

APPLE EXHIBIT 1102, Page 750 of 1048

[0103] In related aspects, a stand alone receiver and/or stand alone transmitter may be used in lieu of or in conjunction with communication module 406. In further related aspects, apparatus 500 may optionally include a means for storing information, such as, for example, a memory device/module 510. Computer readable medium or memory device/module 510 may be operatively coupled to the other components of apparatus 500 via bus 502 or the like. The computer readable medium or memory device 510 may be adapted to store computer readable instructions and data for effecting the processes and behavior of means 520-530, and components thereof, or processor 508 (in the case of apparatus 500 configured as a computing device) or the methods disclosed herein.

[0104] In yet further related aspects, the memory module 510 may optionally include executable code for the processor module 508 to provide a SPN between a TMC and a field security device (in operative communication with a field traffic controller) by: (a) in response to the communication module receiving the device identifier from the field security device, accessing a database of authorized device identifiers corresponding to known field security devices; and (b) in response to the received device identifier matching one of the authorized device identifiers, establishing the SPN between the field security device and the TMC. One or more of steps (a)-(b) may be performed by processor module 508 in lieu of or in conjunction with the means 520-530 described above.

[0105] In accordance with one or more aspects of the embodiments described herein, Figure 6 illustrates an exemplary apparatus 600 (e.g., a network device) for securely communicating with a TMC. Apparatus 600 may include a communication module 606 adapted to access a public network. Apparatus 600 may include at least one processor operatively coupled to the communication module 606, and a memory module 610 operatively coupled to the at least one processor and comprising executable code for the at least one processor.

[0106] Apparatus 600 may comprise a means 620 for accessing the public network, and a means 630 for locating an authentication server of the TMC via the public network. Apparatus 600 may further comprise a means 640 for sending a device identifier to the authentication server, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the network device. Apparatus 600 may further comprise a means 650 for establishing a SPN with the TMC, in response to the authentication server authenticating the device identifier. In related aspects, apparatus 600 may comprise a laptop computer, a mobile phone, or any other network device.

[0107] It is noted that apparatus 600 may optionally include a processor module 608 having at least one processor, in the case of apparatus 600 configured as computing device, rather than as a processor. Processor 608, in such case, may be in operative communication with means 620-650, and components thereof, via a bus 602 or similar communication coupling. Processor 608 may effect

APPLE EXHIBIT 1102, Page 751 of 1048

initiation and scheduling of the processes or functions performed by means 620-650, and components thereof.

[0108] In related aspects, a stand alone receiver and/or stand alone transmitter may be used in lieu of or in conjunction with communication module 606. In further related aspects, apparatus 600 may optionally include a means for storing information, such as, for example, a memory device/module 610. Computer readable medium or memory device/module 610 may be operatively coupled to the other components of apparatus 600 via bus 602 or the like. The computer readable medium or memory device 610 may be adapted to store computer readable instructions and data for effecting the processes and behavior of means 620-650, and components thereof, or processor 608 (in the case of apparatus 600 configured as a computing device) or the methods disclosed herein.

[0109] In yet further related aspects, the memory module 610 may optionally include executable code for the processor module 608 to securely communicate with a TMC by: (a) accessing the public network; (b) locating an authentication server of the TMC via the public network; (c) sending a device identifier to the authentication server; and (d) establishing a SPN with the TMC, in response to the authentication server authenticating the device identifier. One or more of steps (a)-(d) may be performed by processor module 608 in lieu of or in conjunction with the means 620-650 described above.

Methods for Secured Communication:

[0110] In accordance with one or more aspects of the embodiments described herein, Figure 7 illustrates an exemplary method 700 for providing a SPN between a device (e.g., a field security apparatus/device, a network device, etc.) and a TMC that may involve steps 710-740 described below. At step 710, the method 700 may involve accessing a public network. An authentication server of the TMC may be located via the public network (step 720). Method 700 may involve sending a device identifier for the device to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the device (step 730). At step 740, in response to the authentication server authenticating the device identifier, the SPN may be established between the TMC and the device. The established SPN may tunnel across one or more segments of the public network. In one approach, step 740 may comprise establishing the SPN between the TMC and a field security apparatus/device. In the alternative, or in addition, step 740 may comprise establishing the SPN between the TMC and a network device.

[0111] In accordance with one or more aspects of the embodiments described herein, Figure 8 illustrates an exemplary method 800 for securely communicating with a TMC that may involve steps 810-840 described below. At step 810, method 800 may involve accessing a public network (e.g., via a

APPLE EXHIBIT 1102, Page 752 of 1048

communication module or the like). An authentication server of the TMC may be located via the public network or the like (step 820). Method 800 may involve sending a device identifier to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the network device (step 830). At step 840, in response to the authentication server authenticating the device identifier from the network device, method 800 may involve establishing a SPN between the network device and the TMC.

[0112] While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

[0113] As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0114] It is understood that the specific order or hierarchy of steps in the processes disclosed herein in an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure. The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

[0115] Moreover, various aspects or features described herein can be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer-readable media can

APPLE EXHIBIT 1102, Page 753 of 1048

include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips, etc.), optical disks (e.g., compact disk (CD), digital versatile disk (DVD), etc.), smart cards, and flash memory devices (e.g., Erasable Programmable Read Only Memory (EPROM), card, stick, key drive, etc.). Additionally, various storage media described herein can represent one or more devices and/or other machine-readable media for storing information. The term "machine-readable medium" can include, without being limited to, wireless channels and various other media capable of storing, containing, and/or carrying instruction(s) and/or data.

[0116] Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 754 of 1048

WHAT IS CLAIMED IS:

1. A field security device for providing a secure private network (SPN) between a field traffic controller and a traffic management center (TMC), comprising:

a first connector for interfacing with the field traffic controller;

a communication module;

at least one processor operative coupled to the first connector and the communication module; and

a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor to:

access a public network via the communication module;

locate an authentication server of the TMC via the public network;

send a device identifier to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the field security device; and

in response to the authentication server authenticating the device identifier from the field security device, establish the SPN between the field security device and the TMC, wherein the established SPN tunnels across at least one segment of the public network.

2. The field security device of Claim 1, wherein the at least one non-user-configurable parameter comprises at least one of CPU ID, CPU model, CPU manufacturer, and CPU voltage for the field security device.

3. The field security device of Claim 1, wherein the at least one non-user-configurable parameter is based on a carbon degradation characteristic of a computer chip of the field security device.

4. The field security device of Claim 1, wherein the at least one non-user-configurable parameter is based on a silicone degradation characteristic of a computer chip of the field security device.

5. The field security device of Claim 1, wherein the at least one user-configurable parameter comprises one of hard disk volume name, user name, device name, user password, and hard disk initialization date for the field security device.

6. The field security device of Claim 1, wherein the device identifier is generated by utilizing at least one irreversible transformation of the at least one user-configurable parameter and the at least one non-user-configurable parameter of the field security device.

APPLE EXHIBIT 1102, Page 755 of 1048

7. The field security device of Claim 6, wherein the device identifier is generated by utilizing a cryptographic hash function on the at least one user-configurable parameter and the at least one non-user-configurable parameter of the field security device.

8. The field security device of Claim 1, wherein the communication module is adapted for wireless communication.

9. The field security device of Claim 1, wherein the public network comprises the public Internet.

10. The field security device of Claim 1, wherein the public network comprises a wireless communication network, the wireless communication network implementing at least one of CDMA and GSM standards.

11. The field security device of Claim 1, wherein the public network comprises a wireless communication network, the wireless communication network implementing at least one of 802.11a, 802.11b, 802.11g, 802.11n, and 802.11p standards.

12. The field security device of Claim 1, wherein the traffic controller comprises a traffic signal.

13. The field security device of Claim 1, wherein the traffic controller comprises a surveillance camera.

14. The field security device of Claim 1, wherein the traffic controller is housed in a field traffic cabinet.

15. The field security device of Claim 14, wherein the field security device is adapted to be housed in the field traffic cabinet.

16. The field security device of Claim 1, wherein the first connector comprises a receiving port.

17. An authentication server for providing a secure private network (SPN) between a traffic management center (TMC) and a field security device, the field security device being in operative communication with a field traffic controller, comprising:

a communication module adapted to receive a device identifier over a public network from the field security device, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the field security device;

at least one processor operatively coupled to the communication module; and

a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor to:

APPLE EXHIBIT 1102, Page 756 of 1048

in response to the communication module receiving the device identifier from the field security device, access a database of authorized device identifiers corresponding to known field security devices; and

in response to the received device identifier matching one of the authorized device identifiers, establish the SPN between the field security device and the TMC, wherein the established SPN tunnels across at least one segment of the public network.

18. The server of Claim 17, wherein the authentication server is located at the TMC.

19. The server of Claim 17, wherein the communication module is adapted for wireless communication.

20. The server of Claim 17, wherein the public network comprises the public Internet.

21. A network device for securely communicating with a traffic management center (TMC), comprising:

a communication module adapted to access a public network;

at least one processor operatively coupled to the communication module; and

a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor to:

access the public network via the communication module;

locate an authentication server of the TMC via the public network;

send a device identifier to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the network device; and

in response to the authentication server authenticating the device identifier from the network device, establish a secure private network (SPN) between the network device and the TMC, wherein the established SPN tunnels across at least one segment of the public network.

22. The network device of Claim 21, wherein the network device comprises one of a laptop computer or a mobile phone.

23. The network device of Claim 21, wherein the at least one non-user-configurable parameter comprises at least one of CPU ID, CPU model, CPU manufacturer, and CPU voltage for the network device.

24. The network device of Claim 21, wherein the at least one non-user-configurable parameter is based on a carbon degradation characteristic of a computer chip of the network device.

25. The network device of Claim 21, wherein the at least one non-user-configurable parameter is based on a silicone degradation characteristic of a computer chip of the network device.

APPLE EXHIBIT 1102, Page 757 of 1048

26. The network device of Claim 21, wherein the at least one user-configurable parameter comprises one of hard disk volume name, user name, device name, user password, and hard disk initialization date for the network device.

27. The network device of Claim 21, wherein the device identifier is generated by utilizing at least one irreversible transformation of the at least one user-configurable parameter and the at least one non-user-configurable parameter of the network device.

28. The network device of Claim 27, wherein the device identifier is generated by utilizing a cryptographic hash function on the at least one user-configurable parameter and the at least one non-user-configurable parameter of the network device.

29. The network device of Claim 21, wherein the communication module is adapted for wireless communication.

30. The network device of Claim 21, wherein the public network comprises the public Internet.

31. A method for providing a secure private network (SPN) between a device and a traffic management center (TMC), comprising:

accessing a public network;

locating an authentication server of the TMC via the public network;

sending a device identifier for the device to the authentication server via the communication module, the device identifier being based on a combination of at least one user-configurable parameter and at least one non-user-configurable parameter of the device; and

in response to the authentication server authenticating the device identifier, establishing the SPN between the TMC and the device, wherein the established SPN tunnels across at least one segment of the public network.

32. The method of Claim 31, wherein establishing the SPN comprises establishing the SPN between the TMC and a field security apparatus.

33. The method of Claim 31, wherein establishing the SPN comprises establishing the SPN between the TMC and a network device.

APPLE EXHIBIT 1102, Page 758 of 1048

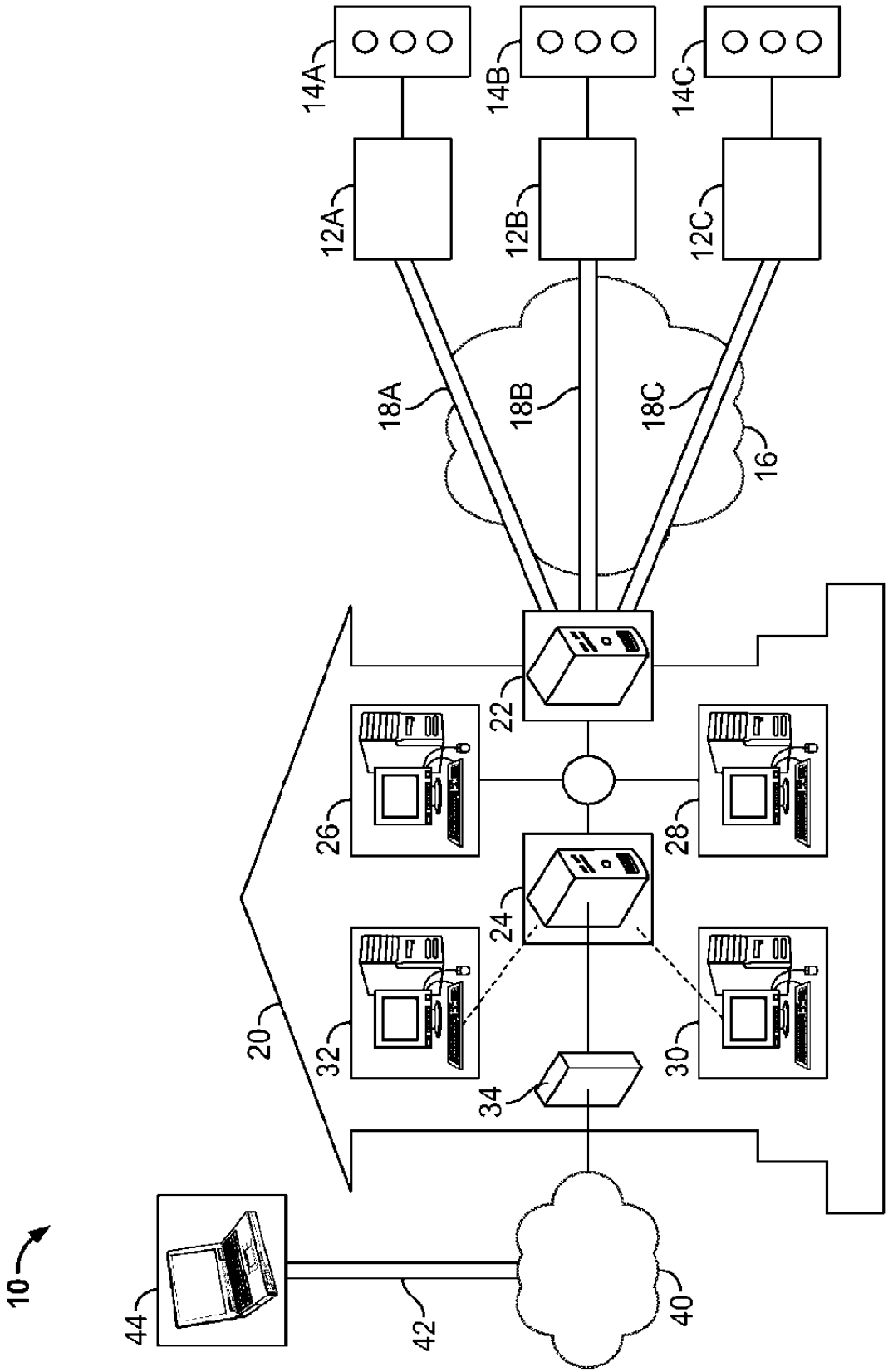

10
14A
14B
14C
12A
12B
12C
18A
18B
18C
16
22
26
28
24
20
32
30
34
44
42
40

FIG. 1

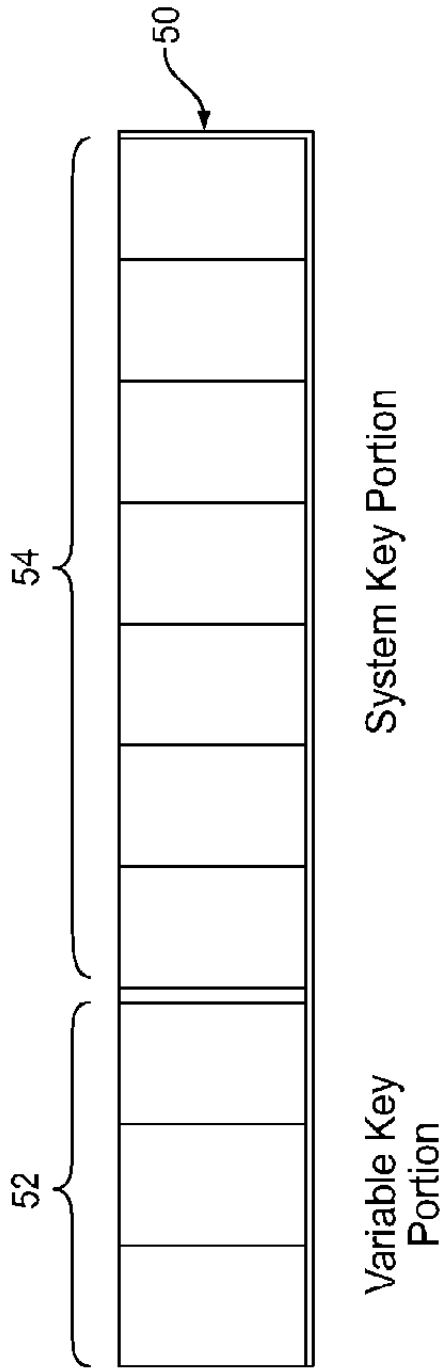


FIG. 2

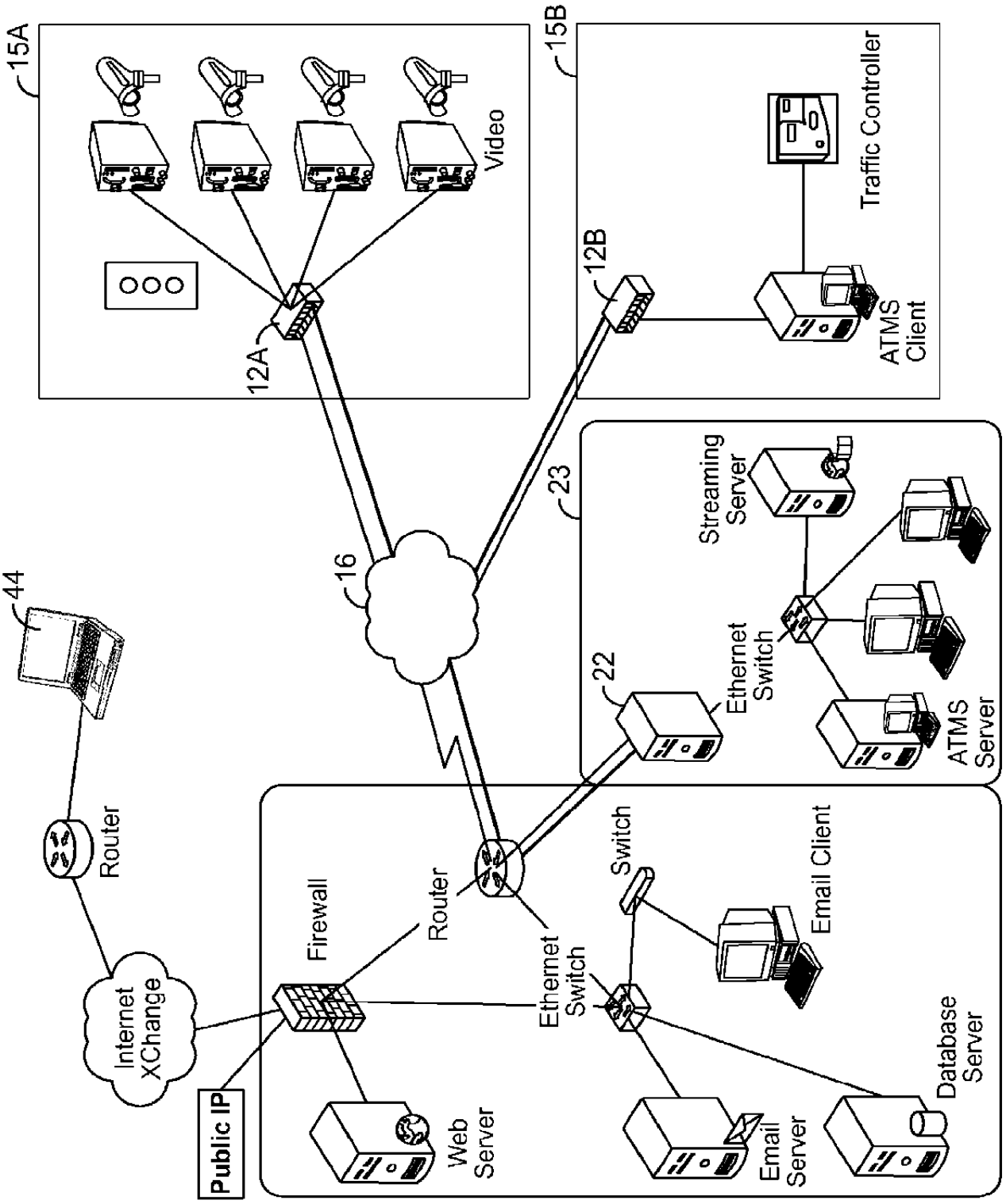

15A
Video
12A
15B
Traffic Controller
12B
ATMS Client
23
Streaming Server
Ethernet Switch
ATMS Server
22
16
44
Router
Internet XChange
Public IP
Firewall
Router
Switch
Email Client
Ethernet Switch
Database Server
Web Server
Email Server

FIG. 3

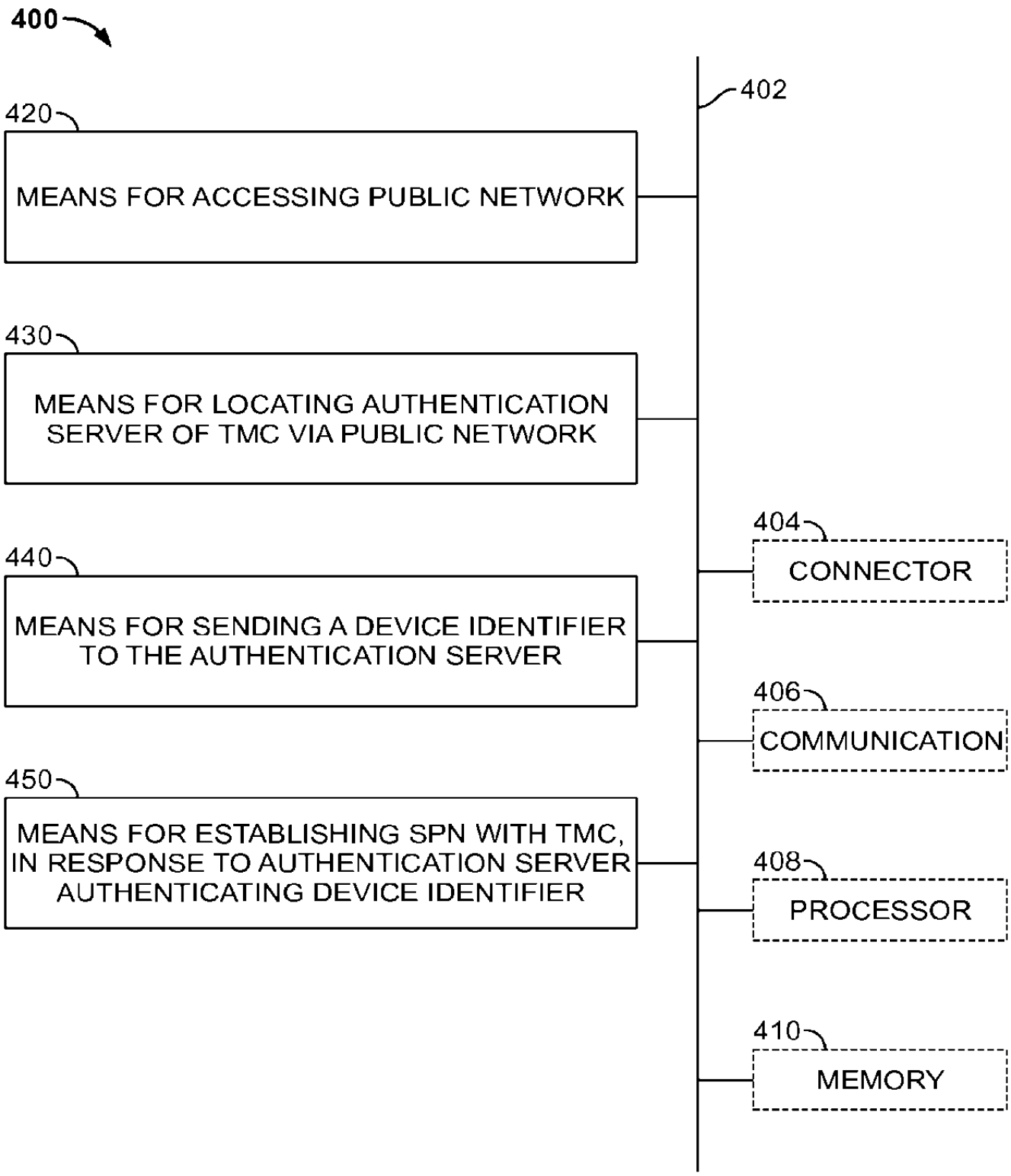

400
402
420
MEANS FOR ACCESSING PUBLIC NETWORK
430
MEANS FOR LOCATING AUTHENTICATION SERVER OF TMC VIA PUBLIC NETWORK
440
MEANS FOR SENDING A DEVICE IDENTIFIER TO THE AUTHENTICATION SERVER
450
MEANS FOR ESTABLISHING SPN WITH TMC, IN RESPONSE TO AUTHENTICATION SERVER AUTHENTICATING DEVICE IDENTIFIER
404
CONNECTOR
406
COMMUNICATION
408
PROCESSOR
410
MEMORY

FIG. 4

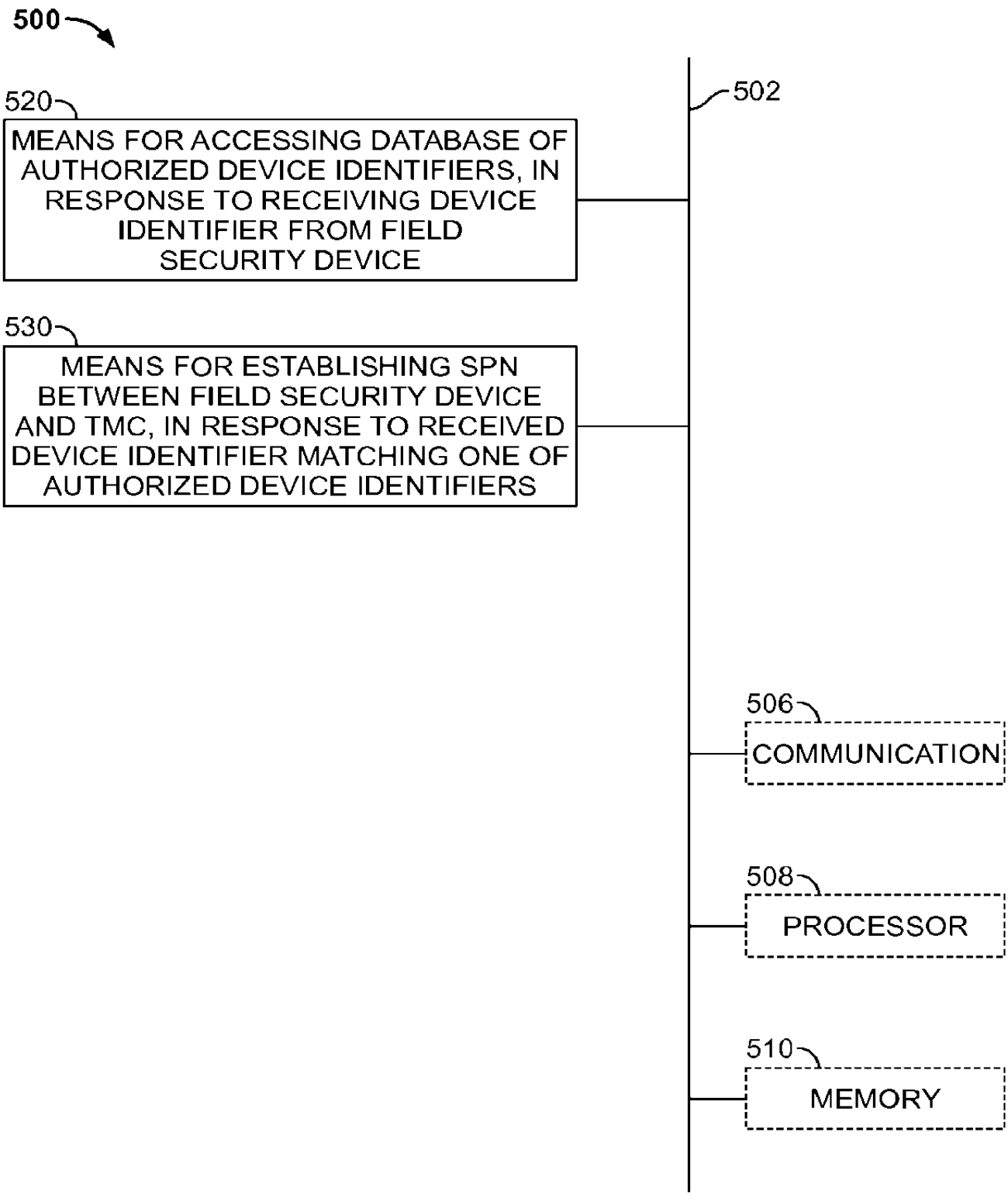

500
502
520
MEANS FOR ACCESSING DATABASE OF AUTHORIZED DEVICE IDENTIFIERS, IN RESPONSE TO RECEIVING DEVICE IDENTIFIER FROM FIELD SECURITY DEVICE
530
MEANS FOR ESTABLISHING SPN BETWEEN FIELD SECURITY DEVICE AND TMC, IN RESPONSE TO RECEIVED DEVICE IDENTIFIER MATCHING ONE OF AUTHORIZED DEVICE IDENTIFIERS
506
COMMUNICATION
508
PROCESSOR
510
MEMORY

FIG. 5

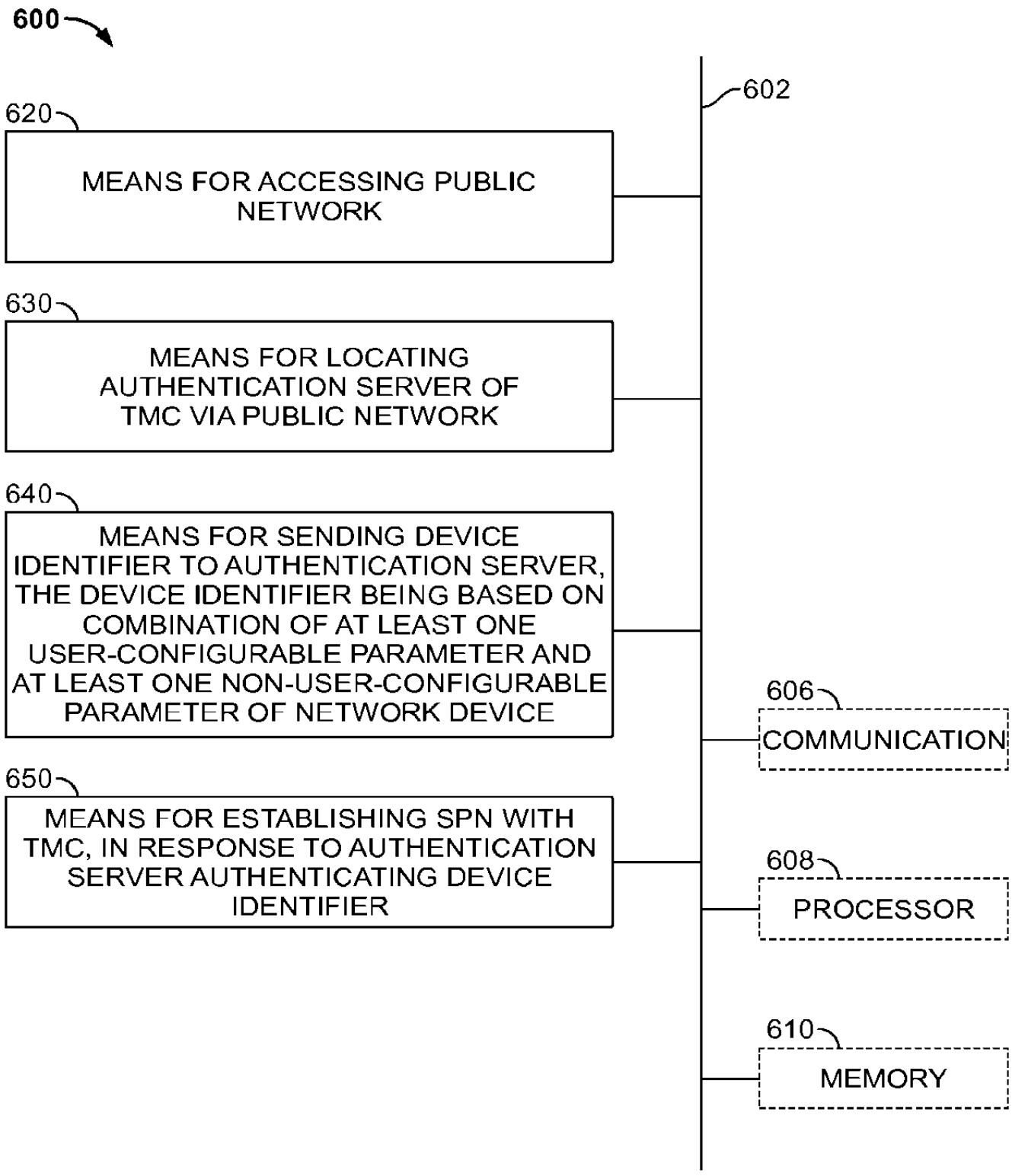

600
602
620
MEANS FOR ACCESSING PUBLIC NETWORK
630
MEANS FOR LOCATING AUTHENTICATION SERVER OF TMC VIA PUBLIC NETWORK
640
MEANS FOR SENDING DEVICE IDENTIFIER TO AUTHENTICATION SERVER, THE DEVICE IDENTIFIER BEING BASED ON COMBINATION OF AT LEAST ONE USER-CONFIGURABLE PARAMETER AND AT LEAST ONE NON-USER-CONFIGURABLE PARAMETER OF NETWORK DEVICE
650
MEANS FOR ESTABLISHING SPN WITH TMC, IN RESPONSE TO AUTHENTICATION SERVER AUTHENTICATING DEVICE IDENTIFIER
606
COMMUNICATION
608
PROCESSOR
610
MEMORY

FIG. 6

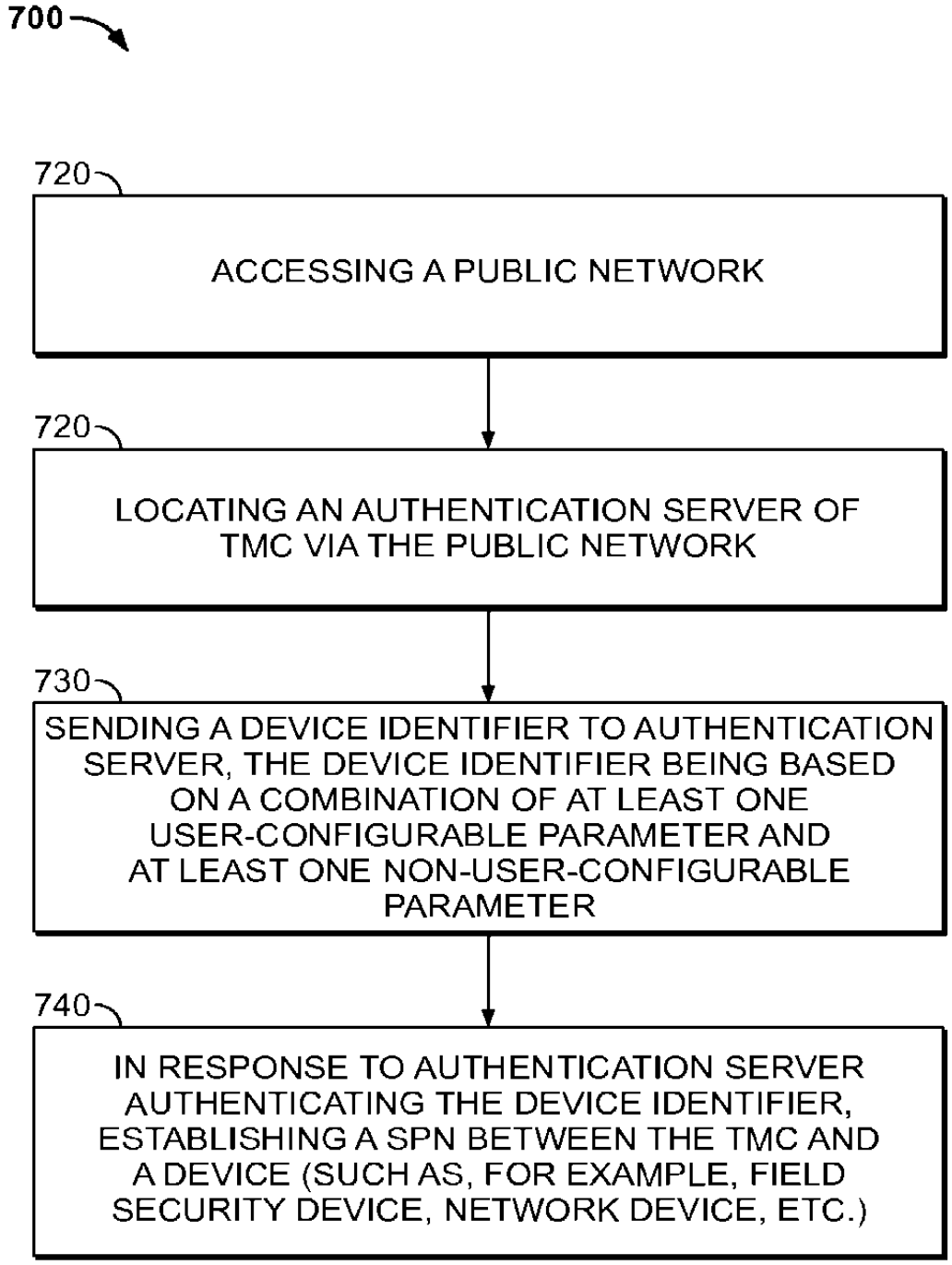

700
720
ACCESSING A PUBLIC NETWORK
720
LOCATING AN AUTHENTICATION SERVER OF TMC VIA THE PUBLIC NETWORK
730
SENDING A DEVICE IDENTIFIER TO AUTHENTICATION SERVER, THE DEVICE IDENTIFIER BEING BASED ON A COMBINATION OF AT LEAST ONE USER-CONFIGURABLE PARAMETER AND AT LEAST ONE NON-USER-CONFIGURABLE PARAMETER
740
IN RESPONSE TO AUTHENTICATION SERVER AUTHENTICATING THE DEVICE IDENTIFIER, ESTABLISHING A SPN BETWEEN THE TMC AND A DEVICE (SUCH AS, FOR EXAMPLE, FIELD SECURITY DEVICE, NETWORK DEVICE, ETC.)

FIG. 7

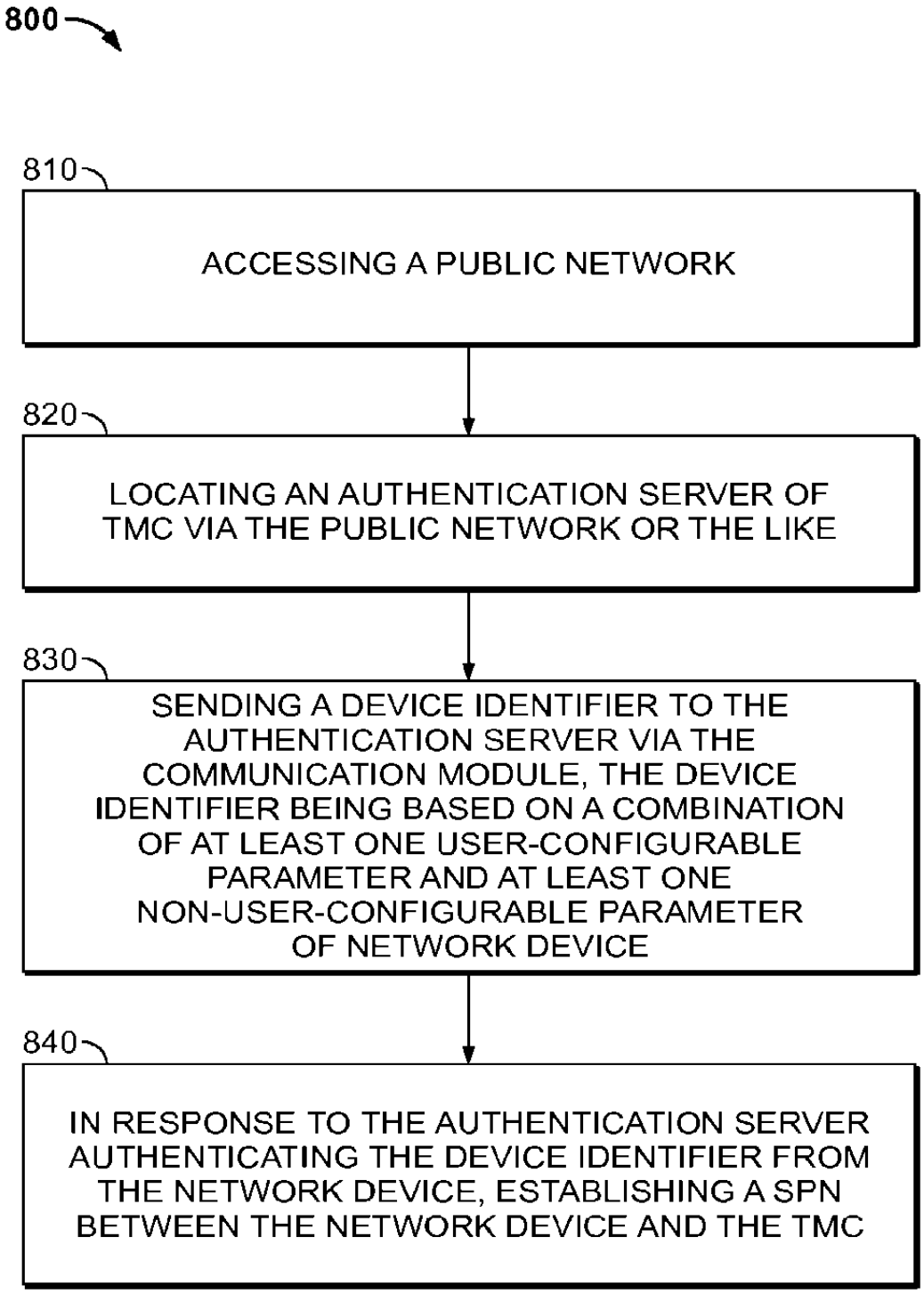

800
810
ACCESSING A PUBLIC NETWORK
820
LOCATING AN AUTHENTICATION SERVER OF TMC VIA THE PUBLIC NETWORK OR THE LIKE
830
SENDING A DEVICE IDENTIFIER TO THE AUTHENTICATION SERVER VIA THE COMMUNICATION MODULE, THE DEVICE IDENTIFIER BEING BASED ON A COMBINATION OF AT LEAST ONE USER-CONFIGURABLE PARAMETER AND AT LEAST ONE NON-USER-CONFIGURABLE PARAMETER OF NETWORK DEVICE
840
IN RESPONSE TO THE AUTHENTICATION SERVER AUTHENTICATING THE DEVICE IDENTIFIER FROM THE NETWORK DEVICE, ESTABLISHING A SPN BETWEEN THE NETWORK DEVICE AND THE TMC

FIG. 8

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2009/044467

A. CLASSIFICATION OF SUBJECT MATTER
INV. H04L29/06 H04L29/08 G08G1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
H04L G06F G08G H04W H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 670 188 A (CIT ALCATEL [FR]) 14 June 2006 (2006-06-14) paragraphs [0011], [0033] - [0042], [0053]; figures 1,2	1-33
X	WO 2005/104686 A (IPASS INC [US]; ELGRESSY MOSHE [IL]; BOB KENNETH [US]) 10 November 2005 (2005-11-10) paragraphs [0024], [0025], [0027], [0029] - [0033], [0037], [0042], [0073] - [0080]; figures 1A,4	1-33

-/--

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 October 2009	03/11/2009

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Ghomrasseni, Z

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2009/044467

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ECONOLITE: "Econolite and Uniloc Partner to Bring Unmatched Infrastructure Security to Advanced Traffic Control Networks with Launch of StrongPoint"[Online] 4 March 2008 (2008-03-04), XP002550941 Retrieved from the Internet: URL:http://www.econolite.com/docs/press/20080304_Econolite_StrongPoint.pdf> [retrieved on 2009-10-12] the whole document	1-33
A	WO 2007/060516 A (LO TEDDY YEUNG MAN [CN]) 31 May 2007 (2007-05-31) paragraphs [0039], [0043] - [0050], [0057] - [0059], [0067], [0068]; figures 1,2,6	1-33
A	ANGHA ET AL: "Securing Transportation Network Infrastructure With Patented Technology Of Device Loching -Developed vy UNILOC USA"[Online] 24 October 2006 (2006-10-24), XP002550942 Retrieved from the Internet: URL:http://www.dksassociates.com/admin/paperfile/ITS%20World%20Paper%20Submission_Uniloc%20_2_.pdf> [retrieved on 2009-10-13] the whole document	1-33

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2009/044467

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1670188 A	14-06-2006	CN 1787533 A	14-06-2006
		US 2006130135 A1	15-06-2006
WO 2005104686 A	10-11-2005	EP 1741045 A2	10-01-2007
		US 2006265446 A1	23-11-2006
WO 2007060516 A	31-05-2007	EP 1917654 A2	07-05-2008

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 December 2009 (30.12.2009)

PCT

(10) International Publication Number
WO 2009/158525 A2

(51) International Patent Classification:
G06Q 30/00 (2006.01) *G06Q 10/00* (2006.01)
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2009/048704

(22) International Filing Date:
25 June 2009 (25.06.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
61/075,690 25 June 2008 (25.06.2008) US

(71) Applicant *(for all designated States except US)*: **UNILOC USA, INC.** [US/US]; 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **ETCHEGOYEN, Craig, S.** [US/US]; Uniloc Usa, Inc., 2151 Michelson Drive, Suite 100, Irvine, CA 92612 (US).

(74) Agent: **PAIK, John, L.**; Connolly Bove Lodge & Hutz, LLP, P.O. Box 2207, Wilmington, DE 19899 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH,

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR MONITORING EFFICACY OF ONLINE ADVERTISING

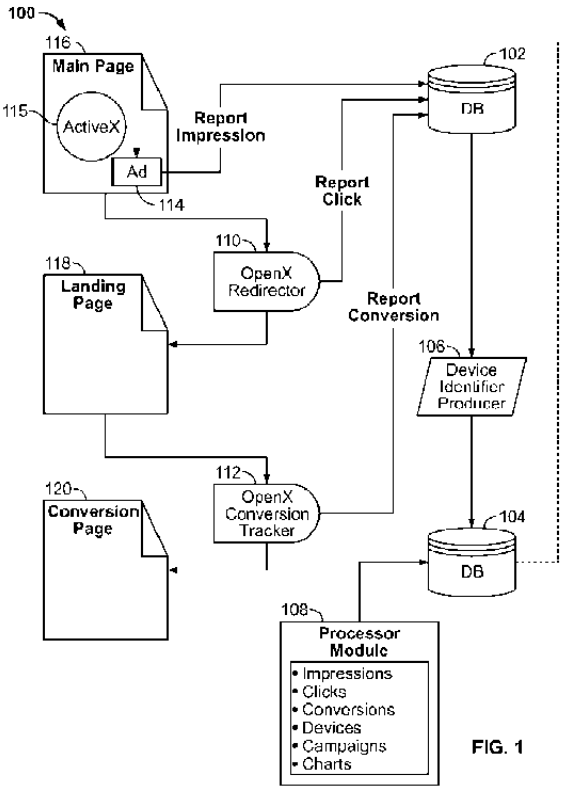


FIG. 1

(57) Abstract: Systems and methods are provided for determining the efficacy of online advertising campaigns. In one approach, the method involves, in response to a user of a network device accessing a web page on which an advertisement is displayed, caching an impression in a first database, and receiving device data regarding the network device. The device data may comprise a combination of user-configurable and non-user-configurable machine parameters. A unique device identifier for the network device may be generated based on the machine parameters. The generated device identifier may be utilized to gain insight into which online ads users look at and which products and/or services they tend to buy.

WO 2009/158525 A2

GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

SYSTEM AND METHOD FOR MONITORING EFFICACY OF ONLINE ADVERTISING

Background of the Invention

Field of the Invention

[0001] The present invention is directed toward systems for monitoring and analyzing the source of device clicks, and related methods.

Description of the Related Art

[0002] Currently, there are limited ways to analyze the effectiveness of online advertising campaigns. Existing approaches to monitoring which ads are viewed or clicked on typically involve utilizing removable cookies and jpeg tracking. Such approaches provide limited information regarding which online ads people tend to look at or what they tend to buy.

[0003] Moreover, such approaches do not provide an effective way to monitor and assess the efficacy of online advertising campaigns by tracking the impression (i.e., when a user sees an ad), clicking, and conversion (i.e., the user bought the product or subscribed to the service shown in the ad) behaviors associated with each given network device (e.g., laptop computer, tablet computer, desktop computer, personal digital assistant, mobile phone or device, etc.). Accordingly, it would be desirable to provide a system and method for generating and assigning unique device identifiers to the network devices, and analyzing the impression, clicking, and conversion behaviors associated with the device identifiers.

Summary of the Invention

[0004] The following presents a simplified summary of one or more embodiments in order to provide a basic understanding of such embodiments. This summary is not an extensive overview of all contemplated embodiments, and is intended to neither identify key or critical elements of all embodiments nor delineate the scope of any or all embodiments. Its sole purpose is to present some concepts of one or more embodiments in a simplified form as a prelude to the more detailed description that is presented later.

[0005] In accordance with one or more embodiments and corresponding disclosure thereof, there is provided a system and method for checking device identifier information to

detect click-fraud (i.e., the act of clicking ads without any intention of purchasing the product or service, thereby making an ad campaign appear more effective than it actually is) or the like.

[0006] In accordance with an aspect of the embodiments described herein, there is provided a system and method for generating precise analytics and thereby gaining insight into the behaviors of consumers (i.e., what they look at and what they tend to buy) who are presented with online ads information that was previously only available via removable cookies and jpeg tracking.

[0007] In accordance with one or more embodiments and corresponding disclosure thereof, various aspects are described in connection with techniques for determining efficacy of online advertising. For example, the method may involve, in response to a user of a network device accessing a web page on which an advertisement is displayed, caching an impression in a first database and receiving device data regarding the network device. The method may involve, in response to the user clicking on the advertisement, caching a user click in the first database and directing the user from the web page to a landing page. The method may involve: generating a device identifier for the network device based on the machine parameters; associating the cached impression and the cached user click with the device identifier; calculating device-specific cached impressions and device-specific cached clicks associated with the device identifier; and displaying information regarding at least one of the device-specific cached impressions and the device-specific cached clicks.

[0008] In related aspects, method may further involve, in response to the user making a purchase on the landing page, caching a conversion in the first database and directing the user from the landing page to a conversion page. The method may also involve: associating the cached conversion with the device identifier; calculating device-specific cached conversions associated with the device identifier; and displaying the device-specific cached conversions.

[0009] To the accomplishment of the foregoing and related ends, the one or more embodiments comprise the features hereinafter fully described and particularly pointed out in the claims. The following description and the annexed drawings set forth in detail certain illustrative aspects of the one or more embodiments. These aspects are indicative, however, of but a few of the various ways in which the principles of various embodiments may be employed and the described embodiments are intended to include all such aspects and their equivalents.

APPLE EXHIBIT 1102, Page 773 of 1048

Brief Description of the Drawings

[0010] Figure 1 provides a block diagram of an exemplary system for monitoring and analyzing impressions, clicks, and conversions.

[0011] Figure 2 illustrates an exemplary advertisement that may be shown on a web page.

[0012] Figure 3 shows an exemplary advertising landing page.

[0013] Figure 4 illustrates components of an exemplary device identifier.

[0014] Figure 5 illustrates an exemplary report generated by the system of Figure 1.

[0015] Figure 6 illustrates an alternative exemplary report generated by the system of Figure 1.

[0016] Figures 7A-C illustrate one embodiment of an apparatus for determining the efficacy of online advertising.

Detailed Description

[0017] The present invention addresses the need for a system and method for assessing the effectiveness of online advertising campaigns by analyzing the impressions, clicks, and conversions associated with the unique device identifiers generated for each network device.

Gathering Ad and User Device Data:

[0018] Figure 1 illustrates an exemplary embodiment of a system 100 for monitoring and assessing the efficacy of online advertising, that generally comprises a first database 102, a second database 104, a device identifier producer 106, a processor module 108, a redirector 110 (e.g., an OpenX redirector), and a conversion tracker 112 (e.g., an OpenX conversion tracker). These components of the system 100 may be located on a single machine or server. In the alternative, or in addition, the components may reside on multiple machines/servers.

[0019] The first database 102 may comprise a cache database in operative communication with the device identifier producer 106, the redirector 110, and the conversion tracker 112. The redirector 110 may be in operative communication with a web page 116 (i.e., an initial publisher page), an advertiser landing page 118, and the first database 102. The redirector 110 may be developed to ensure that device analytics are collected and stored in one or more databases on an advertising campaign by campaign basis.

APPLE EXHIBIT 1102, Page 774 of 1048

[0020] The web page 116 may comprise an advertisement 114, such as, for example, the ad 114 shown in Figure 2. The advertisement 114 may be provided to the web page 116 by the second database 104. In the alternative, or in addition, the advertisement 114 may originate from another source. The conversion tracker 112 may be in operative communication with the landing page 118, an advertiser conversion page 120, and the first database 102. The second database 104 may comprise an OpenX database in operative communication with the device identifier producer 106 and the processor module 108.

[0021] In one embodiment, in response to a user of a network device accessing the web page 116 on which the advertisement 114 is displayed, the first database 102 may receive and cache an impression. This may be achieved via an ActiveX control 115 or the like (e.g., Java applet) embedded in the web page 116. Further, the ActiveX control 115 may, alone or in conjunction with other applets installed on the network device, send device data regarding the network device to the first database 102 to be stored/cached. The device data may comprise user-configurable and/or non-user-configurable machine parameters, which may be used by the device identifier producer 106 to generate a unique device identifier for the network device, as explained in further detail below.

[0022] In response to the user clicking on the advertisement 114 on the web page 116, the redirector 110 may cache a user click in the first database 102, and may direct the user from the web page 116 to a landing page 118, such as, for example, the landing page 118 shown in Figure 3. In response to the user making a purchase on the landing page 118, the conversion tracker 112 may cache a conversion in the first database, and may direct the user from the landing page 118 to a conversion page 120. The first database 102 may optionally store the device data and/or information regarding impressions, clicks, and conversions, etc. as comma separated values (CSV), lists, or files.

[0023] The device identifier producer 106 may receive or access the device data (i.e., the user-configurable and/or non-user-configurable machine parameters) stored on the first database 102, and utilize such data to generate a device identifier for the network device. The producer 106 may comprise an application or applet residing on a machine/server with the first database 102. In the alternative, or in addition, the producer 106 or components thereof may be located on machine/server that is separate from, but in operative communication with, the first database 102 and/or second database 104. In another embodiment, the producer 106 or

components thereof may be located on the network device, and may generate and send the device identifier to the first database 102 and/or the second database 104.

[0024] In another embodiment, in response to the user accessing the web page 116 on which the advertisement 114 is displayed, the first database 102 may update an impression count for the advertisement 114. Similarly, in response to the user clicking on the advertisement 114 on the web page 116, the first database may update a click count for the advertisement 114.

[0025] In yet another embodiment, the system 100 does not include the conversion tracker 112; rather, the redirector 110 may be in operative communication with the web page 116, the landing page 118, the conversion page 120, and the first database 102. The redirector 110 may take on the role of the conversion tracker 112 in addition to its own role in system 100.

Device Identifiers:

[0026] The device identifier may be generated from machine parameters of the network device, such as, for example, hard disk volume name, user name, device name, user password, hard disk initialization date, etc. The machine parameters may relate to the platform on which the web browser runs, such as, for example, CPU number, or unique parameters associated with the firmware in use. The machine parameters may also include system configuration information, such as amount of memory, type of processor, software or operating system serial number, etc. The device identifier generated from the machine parameters may include the network device's Internet Protocol (IP) address and/or other geo-location code to add another layer of specificity to the network device's unique identifier. In the alternative, or in addition, the device identifier may comprise a randomly generated and assigned number that is unique for the network device.

[0027] It is noted that an application running on the network device or otherwise having access to the network device's hardware and file system may generate a unique device identifier using a process that operates on data indicative of the network device's configuration and hardware. It is also noted that an application or module (e.g. device identifier producer 106 in system 100 of Figure 1) running on a server or the like, in communication with the network device, may receive device data regarding the network device and generate the device identifier for the network device.

APPLE EXHIBIT 1102, Page 776 of 1048

[0028] The device identifier may be generated using a combination of user-configurable and non-user-configurable machine parameters as input to a process that results in the device identifier, which may be expressed in digital data as a binary number. Each machine parameter may include data determined by a hardware component, software component, or data component specific to the device that the unique identifier pertains to. Machine parameters may be selected based on the target device system configuration such that the resulting device identifier has a very high probability (e.g., greater than 99.999%) of being unique to the target device. In addition, the machine parameters may be selected such that the device identifier includes at least a stable unique portion up to and including the entire identifier that has a very high probability of remaining unchanged during normal operation of the target device. Thus, the resulting device identifier should be highly specific, unique, reproducible and stable as a result of properly selecting the machine parameters.

[0029] The device identifier producer 106 may comprise an application that operates on the collected parameters with one or more algorithms to generate the device identifier. This process may include at least one irreversible transformation, such as, for example, a cryptographic hash function, such that the input machine parameters cannot be derived from the resulting device identifier. Each device identifier, to a very high degree of certainty, cannot be generated except by the suitably configured application operating on or otherwise having access to the same field security device on which the device identifier was first generated. Conversely, each identifier, again to a very high degree of certainty, can be successfully reproduced by the suitably configured application operating on or otherwise having access to the same field security device on which the identifier was first generated.

[0030] In one embodiment, the ActiveX control 115 or the like may operate by performing or initiating a system scan to determine a present configuration of the field security device. The producer 106 may then select the machine parameters to be used as input for generating the unique device identifier. Selection of parameters may vary depending on the system configuration. Once the parameters are selected, the producer 106 may generate the identifier.

[0031] Further, generating the device identifier may also be described as generating a device fingerprint and may entail the sampling of physical, non-user configurable properties, as well as a variety of additional parameters, such as uniquely generated hashes and time sensitive values. Physical device parameters available for sampling may include, for example, unique manufacturer characteristics, carbon and silicone degradation and small device failures.

APPLE EXHIBIT 1102, Page 777 of 1048

[0032] The process of measuring carbon and silicone degradation may be accomplished by measuring a chip's ability to process complex mathematical computations, and its ability to respond to intensive time variable computations. These processes measure how fast electricity travels through the carbon. Using variable offsets to compensate for factors such as heat and additional stresses placed on a chip during the sampling process allows for each and every benchmark to reproduce the expected values. During a standard operating lifetime, the process of passing electricity through the various switches causes a computer chip to degrade. These degradations manifest as gradually slower speeds that extend the processing time required to compute various benchmarking algorithms.

[0033] In addition to the chip benchmarking and degradation measurements, the process for generating a device identifier may include measuring physical, non-user-configurable characteristics of disk drives and solid state memory devices. Each data storage device has a large variety of damage and unusable data sectors that are nearly unique to each physical unit. The ability to measure and compare values for damaged sectors and data storage failures provides a method for identifying storage devices.

[0034] Device parameter sampling, damage measurement and chip benchmarking make up just a part of device fingerprinting technologies described herein. These tools may be further extended by the use of complex encryption algorithms to convolute the device identifier values during transmission and comparisons. Such encryption processes may be used in conjunction with random sampling and key generations.

[0035] The device identifier may be generated by utilizing machine parameters associated with one or more of the following: machine model; machine serial number; machine copyright; machine ROM version; machine bus speed; machine details; machine manufacturer; machine ROM release date; machine ROM size; machine UUID; and machine service tag.

[0036] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: CPU ID; CPU model; CPU details; CPU actual speed; CPU family; CPU manufacturer; CPU voltage; and CPU external clock.

[0037] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: memory model; memory slots; memory total; and memory details.

[0038] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: video model; video details; display model; display details; audio model; and audio details.

[0039] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: network model; network address; Bluetooth address; Blackbox model; Blackbox serial; Blackbox details; Blackbox damage map; Blackbox volume name; NetStore details; and NetStore volume name.

[0040] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: optical model; optical serial; optical details; keyboard model; keyboard details; mouse model; mouse details; printer details; and scanner details.

[0041] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: baseboard manufacturer; baseboard product name; baseboard version; baseboard serial number; and baseboard asset tag.

[0042] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: chassis manufacturer; chassis type; chassis version; and chassis serial number.

[0043] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: IDE controller; SATA controller; RAID controller; and SCSI controller.

[0044] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: port connector designator; port connector type; port connector port type; and system slot type.

[0045] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: cache level; cache size; cache max size; cache SRAM type; and cache error correction type.

[0046] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: fan; PCMCIA; modem; portable battery; tape drive; USB controller; and USB hub.

APPLE EXHIBIT 1102, Page 779 of 1048

[0047] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: device model; device model IMEI; device model IMSI; and device model LCD.

[0048] The device identifier may also be generated by utilizing machine parameters associated with one or more of the following: wireless 802.11; webcam; game controller; silicone serial; and PCI controller.

[0049] With reference to Figure 4, in one exemplary embodiment, a device identifier 130 may include two components – namely, a variable key portion 132 and a system key portion 134. The variable key portion 132 may be generated by reference to a variable platform parameter, such as via reference to system time information, although other parameters which are variable may be utilized in other embodiments. The system key portion 134 may include the above described parameters expected to be unique to the network device, such as, for example, hard disk volume name, user name, computer name, user password, hard disk initialization date, or combinations thereof. Portions 132 and/or 134 may be combined with the IP address and/or other platform parameters of the network device.

[0050] It is further noted that the device identifiers or machine fingerprints generated by the device identifier producer 106 or the like uniquely identify a given network device when compared to other computers, machines, or devices. The device identifier generated for a given device may be tolerant enough to account for minor changes to a device, such as normal wear and tear or minor changes to the configuration of the device. Such built in tolerances to a given device identifier may allow the device hardware to be upgraded or modified over time without affecting the ability to recognize or differentiate the given device.

Analytics:

[0051] With reference once again to Figure 1, the system 100 may comprise the second database 104 that is in operative communication with the device identifier producer 106, the processor module 108, and optionally with a display unit (not shown). The device identifier generated by the producer 106 may be stored in the second database 104, which may comprise an ad database, such as, for example, an OpenX database or the like. Other data that may be cached in the first database 102 (e.g., the cached impression, the cached user click, cached

conversion, impressions counts, click counts, conversions counts, etc.) may be stored in the second database 104.

[0052] The processor module 108 may analyze the data in the second database 104 in any number of ways and may be used to generate an analytics database or the like. The processor module 108 may comprise an analytics engine, data mining tool, or the like to collect, process, and display statistical information relating, but not limited, to: impressions; clicks; conversions; devices; advertising campaigns (i.e., the actual ads that were clicked on by the user); click through rates (CTR); effective costs per 1000 impressions (ECPM); click fraud information, etc.

[0053] Data regarding impressions, clicks, and/or conversions may be augmented with the device identifier or digital fingerprint of the respective network devices to get a greater understanding about the individuals viewing specific ads. For example, in the embodiment shown in Figure 5, the processor module 108 may associate the cached impression(s), the cached user click(s), and the cached conversion(s) with the device identifier, and may calculate total cached impressions, total cached clicks, and total cached conversions associated with each unique device identifier. The processor module 108 may instruct the display unit to display one or more of the total cached impressions, the total cached clicks, and the total cached conversions for each unique identifier on an analytics page (i.e., a page that displays device statistics for the advertising campaigns).

[0054] Data regarding conversions and tracking may be presented in any known way, including tables, charts, etc. For example, the data may be organized by advertising campaigns, or specification ads, total impressions, total clicks, CTRs, revenues, ECPM, etc. The data may further be sorted by genuine unique IDs (i.e., device identifiers of the network devices), and for each unique ID by advertising campaign, impressions, clicks, and conversions. The data may be extracted from a specific period, advertising campaign, or any other criteria. The display unit may display such information as readable graphs and charts, and may incorporate various color schemes, as well as animation.

[0055] With reference to Figure 6, the data stored in the second database 104 may be processed to remove redundant data and/or impressions and clicks resulting from click-fraud. By generating unique signatures (i.e., the device identifiers) for network devices, it is possible to monitor the behavior and impression/clicking/conversion patterns of the network device

APPLE EXHIBIT 1102, Page 781 of 1048

users, and thereby account for suspicious behavior (e.g., click-fraud). For example, in one approach, the processor module 108 may be adapted to discount or ignore an unusually high number of ad impressions and/or clicks for a given device identifier. The criteria for filtering out or identifying odd or out-of-the-ordinary impression/click behavior may be based on the number of impressions and/or clicks in a defined period. For example, the processor module 108 may be adapted to ignore clicks from a network device if there are more than three clicks associated with the network device's device identifier in a one day period.

[0056] By assessing which impressions and/or clicks are associated with which device identifiers, the processor module 108 may determine the total number of unique impressions and unique clicks (i.e., the impressions and clicks that remain after one removes the redundant or suspicious impressions and clicks associated with certain device identifiers), as shown in Figure 6. The unique CTR may be calculated for each ad by dividing the unique clicks by the unique impressions. Accordingly, the cost-per-impression becomes cost-per-genuine-unique-impression, while cost-per-click becomes cost-per-genuine-unique-click.

[0057] In another embodiment, the processor module 108 may associate the cached impression and the cached user click with the device identifier, and may calculate the total cached impressions and total cached clicks associated with the device identifier. For each unique device identifier, the processor module 108 may divide the total cached clicks by the total cached impressions to calculate a device-specific click-through rate. In the alternative, or in addition, the processor module 108 may divide the click count by the impression count to calculate a general click-through rate. The processor module 108 may instruct the display unit to display the device-specific click-through rate and/or the general click-through rate. In the alternative, or in addition, the display unit may display the difference between the device-specific click-through rate and the general click-through rate as a number, graphically, or combinations thereof.

[0058] In yet another embodiment, the analytics performed by the processor module 108 with the information in the second database 104 may incorporate geo-location cross-referencing to monitor and compare the behaviors of consumers in different locations.

[0059] According to related aspects, the cached/collected data regarding ad impressions, ad clicks, click through rates, revenues, ECPM, ad campaigns, device data, device identifiers, and/or geo-location data, and combinations thereof may be organized and presented in any

number of ways (e.g., charts, graphs, line items, folders, etc.). It is noted that the analytics and information presented in Figures 5 and 6 are merely exemplary, and that the techniques described herein relate more generally to analyzing and organizing data regarding ad campaigns based at least in part on the device identifiers.

[0060] In accordance with one or more embodiments and corresponding disclosure thereof, various aspects are described in connection with identifying a device that is associated with too many ad campaigns, ad impressions, ad clicks, conversions, and/or products sold, or combinations thereof, that is, a number of ad campaigns, ad impressions, ad clicks, etc. greater than normal for a given item (e.g., ad clicks from a given device) or exceeding a defined upper limit (e.g., ad impressions and/or clicks for a given ad campaign from a given device that exceed a defined number, such as ten ad clicks). It is noted that once a device identifier is generated for a given device, it is possible to track or monitor which ads device users are viewing, clicking, and/or converting into actual sales/transactions of products or services.

[0061] According to related aspects, geo-location data or codes may be collected from devices. The collected geo-location data may optionally be incorporated into or utilized in generating the unique device identifiers for the devices. In one exemplary approach, the geo-location data comprise IP addresses, information, or the like. The collected geo-location data about the devices may be used to deliver local ads (i.e., ads from stores and establishments located near the device user) and/or geo-targeted/located ads (i.e., ads that are aimed or targeted at people in a particular geographic location).

[0062] In accordance with one or more aspects of the embodiments described herein, there are provided devices and apparatuses for determining the efficacy of online advertising. With reference to Figures 7A-C, there is provided an exemplary apparatus 700 that may be configured as either a computing device, or as a processor or similar device for use within a computing device. As illustrated in Figure 7A, apparatus 700 may comprise a means 720 for, in response to a user of a network device accessing a web page on which an advertisement is displayed, caching an impression in a first database and receiving device data regarding the network device. The device data may comprise a combination of at least one user-configurable machine parameter and at least one non-user-configurable machine parameter of the network device.

APPLE EXHIBIT 1102, Page 783 of 1048

[0063] Apparatus 700 may comprise a means 722 for, in response to the user clicking on the advertisement, caching a user click in the first database and directing the user from the web page to a landing page. Apparatus 700 may comprise a means 724 for generating a device identifier for the network device based on the machine parameters. Apparatus 700 may comprise a means 726 for associating the cached impression and the cached user click with the device identifier. Apparatus 700 may comprise a means 728 calculating device-specific cached impressions and device-specific cached clicks associated with the device identifier. Apparatus 700 may comprise a means 730 for displaying information regarding at least one of the device-specific cached impressions and the device-specific cached clicks.

[0064] With reference to Figure 7B, apparatus 700 may comprise a means 740 for, in response to the user making a purchase on the landing page, caching a conversion in the first database and directing the user from the landing page to a conversion page. Apparatus 700 may comprise: a means 742 for associating the cached conversion with the device identifier; a means 744 for calculating device-specific cached conversions associated with the device identifier; a means 746 for displaying the device-specific cached conversion; and a means 748 for storing the device identifier and data regarding the cached conversion in a second database.

[0065] With reference to Figure 7C, apparatus 700 may comprise a means 750 for storing the device identifier and data regarding at least one of the cached impression and the cached user click in a second database. Apparatus 700 may comprise a means 760 for calculating a device-specific click-through rate. Calculating the device-specific click-through rate may involve dividing the device-specific cached clicks by the device-specific cached impressions.

[0066] Apparatus 700 may comprise a means 770 for calculating a general click-through rate. The first database may update a total impression count for the advertisement, in response to the user of the network device accessing the web page on which the advertisement is displayed. The first database may update a total click count for the advertisement, in response to the user clicking on the advertisement. Calculating the general click-through rate may involve dividing the total click count by the total impression count. At least one of the device-specific click-through rate and general click-through rate may be displayed on a display module.

[0067] Apparatus 700 may comprise a means 780 for calculating a device-specific conversion rate. Calculating the device-specific conversion rate may involve dividing the

APPLE EXHIBIT 1102, Page 784 of 1048

device-specific cached conversions by the device-specific cached impressions or the device-specific cached clicks.

[0068] Apparatus 700 may comprise a means 790 for calculating a general conversion rate. The first database may update a total impression count for the advertisement, in response to the user of the network device accessing the web page on which the advertisement is displayed. The first database may update a total click count for the advertisement, in response to the user clicking on the advertisement. The first database may update a total conversion count for the advertisement, in response to the user making the purchase on the landing page. Calculating the general conversion rate may involve dividing the total conversion count by the total impression count or the total click count. At least one of the device-specific conversion rate and general conversion rate may be displayed on a display module.

[0069] Apparatus 700 may optionally include a processor module 706 having at least one processor, in the case of apparatus 700 configured as computing device, rather than as a processor. Processor 706, in such case, may be in operative communication with means 720-790, and components thereof, via a bus 702 or similar communication coupling. Processor 706 may effect initiation and scheduling of the processes or functions performed by means 720-790, and components thereof.

[0070] Apparatus 700 may include a transceiver/communication module 704 for communicating with one or more network devices. A stand alone receiver and/or stand alone transmitter may be used in lieu of or in conjunction with communication module 704.

[0071] Apparatus 700 may optionally include a means for storing information, such as, for example, a memory device/module 708. Computer readable medium or memory device/module 708 may be operatively coupled to the other components of apparatus 700 via bus 702 or the like. The computer readable medium or memory device 708 may be adapted to store computer readable instructions and data for effecting the processes and behavior of means 720-790, and components thereof, or processor 706 (in the case of apparatus 700 configured as a computing device) or the methods disclosed herein.

[0072] In related aspects, the memory module 708 may optionally include executable code for the processor module 706 to: (a) in response to a user of a network device accessing a web page on which an advertisement is displayed, (i) caching an impression in a first database and (ii) receiving device data regarding the network device; (b) in response to the user clicking on

APPLE EXHIBIT 1102, Page 785 of 1048

the advertisement, (i) caching a user click in the first database and (ii) directing the user from the web page to a landing page; (c) generating a device identifier for the network device based on the machine parameters; (d) associating the cached impression and the cached user click with the device identifier; (e) calculating device-specific cached impressions and device-specific cached clicks associated with the device identifier; and (f) displaying information regarding at least one of the device-specific cached impressions and the device-specific cached clicks. One or more of steps (a)-(f) may be performed by processor module 706 in lieu of or in conjunction with the means 720-790 described above.

[0073] It is noted that one or more of the techniques and methodologies described herein may be performed by embedded applications, platforms, or systems. The methods described herein may be performed by a general-purpose computer system and/or an embedded application or component of a special-purpose apparatus (e.g., traffic controller, traffic signal, surveillance cameras, sensors, detectors, vehicles, vehicle navigation systems, mobile phones, personal digital assistants, etc.). In one embodiment, the special-purpose device comprises an embedded platform running an embedded Linux operating system (OS) or the like. For example, the unique device identifier or fingerprint for the special-purpose device may be created by collecting and using one or more of the following information: machine model; processor model; processor details; processor speed; memory model; memory total; network model of each Ethernet interface; network MAC address of each Ethernet interface; BlackBox model (e.g., any Flash device); BlackBox serial (e.g., using Dallas Silicone Serial DS-2401 chipset or the like); OS install date; nonce value; nonce time of day; any other predefined hardware information stored (optionally encrypted) in EEPROM; and any variations/combinations thereof.

[0074] While the present invention has been illustrated and described with particularity in terms of preferred embodiments, it should be understood that no limitation of the scope of the invention is intended thereby. Features of any of the foregoing methods and devices may be substituted or added into the others, as will be apparent to those of skill in the art. It should also be understood that variations of the particular embodiments described herein incorporating the principles of the present invention will occur to those of ordinary skill in the art and yet be within the scope of the invention.

[0075] As used in this application, the terms "component," "module," "system," and the like are intended to refer to a computer-related entity, either hardware, firmware, a

combination of hardware and software, software, or software in execution. For example, a component can be, but is not limited to being, a process running on a processor, a processor, an object, an executable, a thread of execution, a program, and/or a computer. By way of illustration, both an application running on a computing device and the computing device can be a component. One or more components can reside within a process and/or thread of execution and a component can be localized on one computer and/or distributed between two or more computers. In addition, these components can execute from various computer readable media having various data structures stored thereon. The components can communicate by way of local and/or remote processes such as in accordance with a signal having one or more data packets (e.g., data from one component interacting with another component in a local system, distributed system, and/or across a network such as the Internet with other systems by way of the signal).

[0076] It is understood that the specific order or hierarchy of steps in the processes disclosed herein is an example of exemplary approaches. Based upon design preferences, it is understood that the specific order or hierarchy of steps in the processes may be rearranged while remaining within the scope of the present disclosure. The accompanying method claims present elements of the various steps in sample order, and are not meant to be limited to the specific order or hierarchy presented.

[0077] Moreover, various aspects or features described herein can be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. For example, computer-readable media can include but are not limited to magnetic storage devices (e.g., hard disk, floppy disk, magnetic strips, etc.), optical discs (e.g., compact disc (CD), digital versatile disc (DVD), etc.), smart cards, and flash memory devices (e.g., Erasable Programmable Read Only Memory (EPROM), card, stick, key drive, etc.). Additionally, various storage media described herein can represent one or more devices and/or other machine-readable media for storing information. The term "machine-readable medium" can include, without being limited to, wireless channels and various other media capable of storing, containing, and/or carrying instruction(s) and/or data.

[0078] Those skilled in the art will further appreciate that the various illustrative logical blocks, modules, circuits, methods and algorithms described in connection with the examples

APPLE EXHIBIT 1102, Page 787 of 1048

disclosed herein may be implemented as electronic hardware, computer software, or combinations of both. To clearly illustrate this interchangeability of hardware and software, various illustrative components, blocks, modules, circuits, methods and algorithms have been described above generally in terms of their functionality. Whether such functionality is implemented as hardware or software depends upon the particular application and design constraints imposed on the overall system. Skilled artisans may implement the described functionality in varying ways for each particular application, but such implementation decisions should not be interpreted as causing a departure from the scope of the present invention.

APPLE EXHIBIT 1102, Page 788 of 1048

WHAT IS CLAIMED IS:

1. A method for determining efficacy of online advertising, comprising:

in response to a user of a network device accessing a web page on which an advertisement is displayed, (a) caching an impression in a first database and (b) receiving device data regarding the network device, the device data comprising a combination of user-configurable and non-user-configurable machine parameters;

in response to the user clicking on the advertisement, (a) caching a user click in the first database and (b) directing the user from the web page to a landing page;

generating a device identifier for the network device based on the machine parameters;

associating the cached impression and the cached user click with the device identifier;

calculating device-specific cached impressions and device-specific cached clicks associated with the device identifier; and

displaying information regarding at least one of the device-specific cached impressions and the device-specific cached clicks.

2. The method of Claim 1, further comprising storing the device identifier and data regarding at least one of the cached impression and the cached user click in a second database.

3. The method of Claim 1, further comprising calculating a device-specific click-through rate.

4. The method of Claim 3, wherein calculating the device-specific click-through rate comprises dividing the device-specific cached clicks by the device-specific cached impressions.

5. The method of Claim 3, further comprising calculating a general click-through rate.

6. The method of Claim 5, wherein:

the first database updates a total impression count for the advertisement, in response to the user of the network device accessing the web page on which the advertisement is displayed;

the first database updates a total click count for the advertisement, in response to the user clicking on the advertisement; and

the step of calculating the general click-through rate comprises dividing the total click count by the total impression count.

7. The method of Claim 6, further comprising displaying at least one of the device-specific click-through rate and general click-through rate.

8. The method of Claim 1, further comprising:

in response to the user making a purchase on the landing page, (a) caching a conversion in the first database and (b) directing the user from the landing page to a conversion page;

associating the cached conversion with the device identifier;

calculating device-specific cached conversions associated with the device identifier; and

displaying the device-specific cached conversions.

9. The method of Claim 8, further comprising storing the device identifier and data regarding the cached conversion in a second database.

10. The method of Claim 8, further comprising calculating a device-specific conversion rate.

11. The method of Claim 10, wherein calculating the device-specific conversion rate comprises dividing the device-specific cached conversions by one of the device-specific cached impressions and the device-specific cached clicks.

12. The method of Claim 10, further comprising calculating a general conversion rate.

APPLE EXHIBIT 1102, Page 790 of 1048

13. The method of Claim 12, wherein:

the first database updates a total impression count for the advertisement, in response to the user of the network device accessing the web page on which the advertisement is displayed;

the first database updates a total click count for the advertisement, in response to the user clicking on the advertisement;

the first database updates a total conversion count for the advertisement, in response to the user making the purchase on the landing page; and

the step of calculating the general conversion rate comprises dividing the total conversion count by one of the total impression count and the total click count.

14. The method of Claim 13, further comprising displaying at least one of the device-specific conversion rate and general conversion rate.

15. A system for determining efficacy of online advertising, comprising:

a first database adapted to, in response to a user of a network device accessing a web page on which an advertisement is displayed, (a) cache an impression and (b) cache device data regarding the network device, the device data comprising a combination of user-configurable and non-user-configurable machine parameters;

a redirector in operative communication with the first database and adapted to, in response to the user clicking on the advertisement, (a) report a user click to the first database, and (b) direct the user from the web page to a landing page;

a conversion tracker in operative communication with the first database and adapted to, in response to the user making a purchase on the landing page, (a) report a conversion to the first database, and (b) direct the user from the landing page to a conversion page; and

a device identifier producer in operative communication with the first server and adapted to generate a device identifier for the network device based on the machine parameters.

APPLE EXHIBIT 1102, Page 791 of 1048

16. The system of Claim 15, further comprising:

a second database adapted to receive (a) the device identifier from the device identifier producer and (b) information regarding any impressions, any user clicks, and any conversions associated with the device identifier from the first database;

a display unit; and

a processor module in operative communication with the second database and the display unit, the processor module being adapted to:

calculate total impressions, total clicks, and total conversions for each unique device identifier; and

instruct the display unit to display at least one of the total impressions, the total clicks, and the total conversions for each unique device identifier.

17. The system of Claim 16, wherein the second server provides the advertisement displayed on the web page.

APPLE EXHIBIT 1102, Page 792 of 1048

18. An apparatus for determining efficacy of online advertising, comprising:

a communication module;

a display module;

at least one processor operatively coupled to the communication module and the display module; and

a memory module operatively coupled to the at least one processor and comprising executable code for the at least one processor to:

in response to a user of a network device accessing a web page on which an advertisement is displayed, (a) cache an impression in a first database and (b) receive device data regarding the network device, the device data comprising a combination of user-configurable and non-user-configurable machine parameters;

in response to the user clicking on the advertisement, (a) cache a user click in the first database and (b) direct the user from the web page to a landing page;

generate a device identifier for the network device based on the machine parameters;

associate the cached impression and the cached user click with the device identifier;

calculate device-specific cached impressions and device-specific cached clicks associated with the device identifier; and

instruct the display module to display information regarding at least one of the device-specific cached impressions and the device-specific cached clicks.

19. The apparatus of Claim 18, wherein the at least one processor instructs the communication nodule to send the device identifier and data regarding at least one of the cached impression and the cached user click to a second database.

APPLE EXHIBIT 1102, Page 793 of 1048

20. The apparatus of Claim 18, wherein the at least one processor:

in response to the user making a purchase on the landing page, (a) caches a conversion in the first database and (b) directs the user from the landing page to a conversion page;

associates the cached conversion with the device identifier;

calculates device-specific cached conversions associated with the device identifier; and

instruct the display module to display information regarding the device-specific cached conversions.

APPLE EXHIBIT 1102, Page 794 of 1048

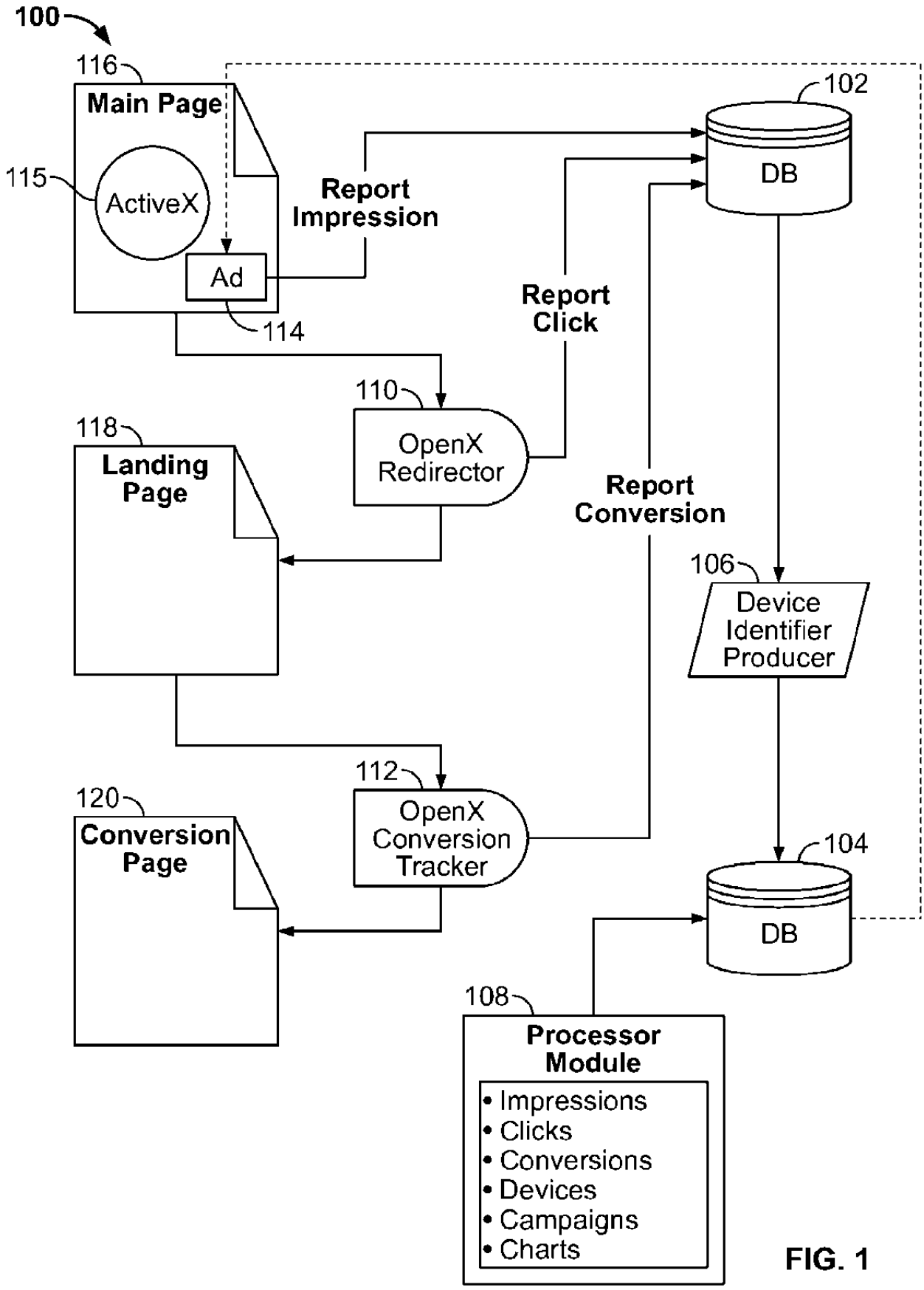

100
116
Main Page
115
ActiveX
Ad
114
Report Impression
102
DB
Report Click
110
OpenX Redirector
118
Landing Page
Report Conversion
106
Device Identifier Producer
112
OpenX Conversion Tracker
120
Conversion Page
104
DB
108
Processor Module
• Impressions
• Clicks
• Conversions
• Devices
• Campaigns
• Charts

FIG. 1

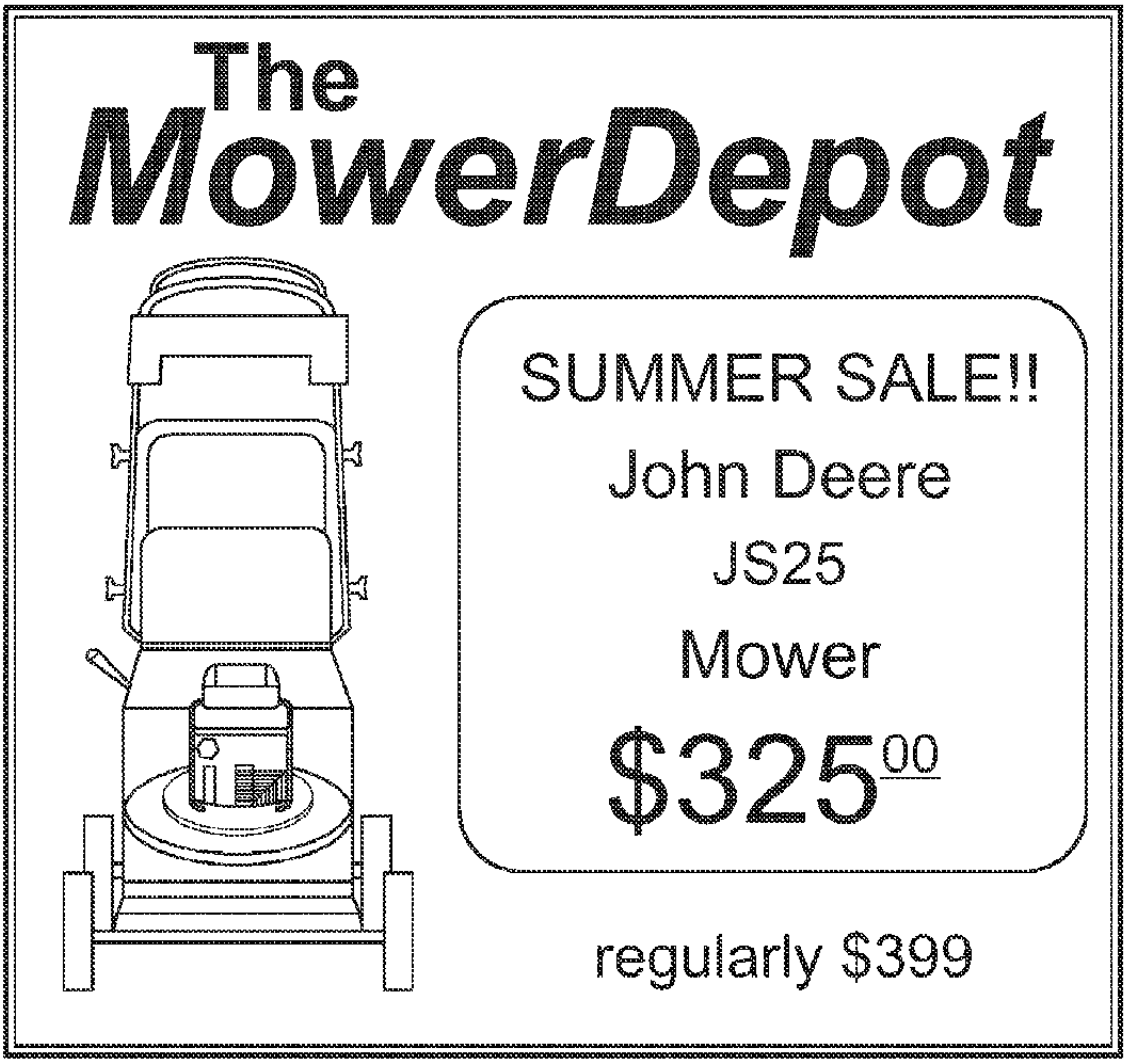

The
MowerDepot
SUMMER SALE!!
John Deere
JS25
Mower
$325.00
regularly $399

FIG. 2

The MowerDepot
Walk-Behind Mowers
Riding Mowers
Parts & Accessories
Special Offers
Store Locations
SUMMER SALE $325.00
John Deere
JS25 Walk-Behind Mower
Add To Cart >>
Key Features
Briggs & Stratton 190 cc engine is manufactured to John Deere standards... more
Discharge-mulch-bag... more
Durable wheels are made of high-impact plastic ... more
One-lever wheel adjuster quickly changes cut-height from 1.25-in. to 4-in. ... more
Conveniently located controls easy to operate ... more
Handlebar is easily adjusted or folded down ... more
Price: $399.00

FIG. 3

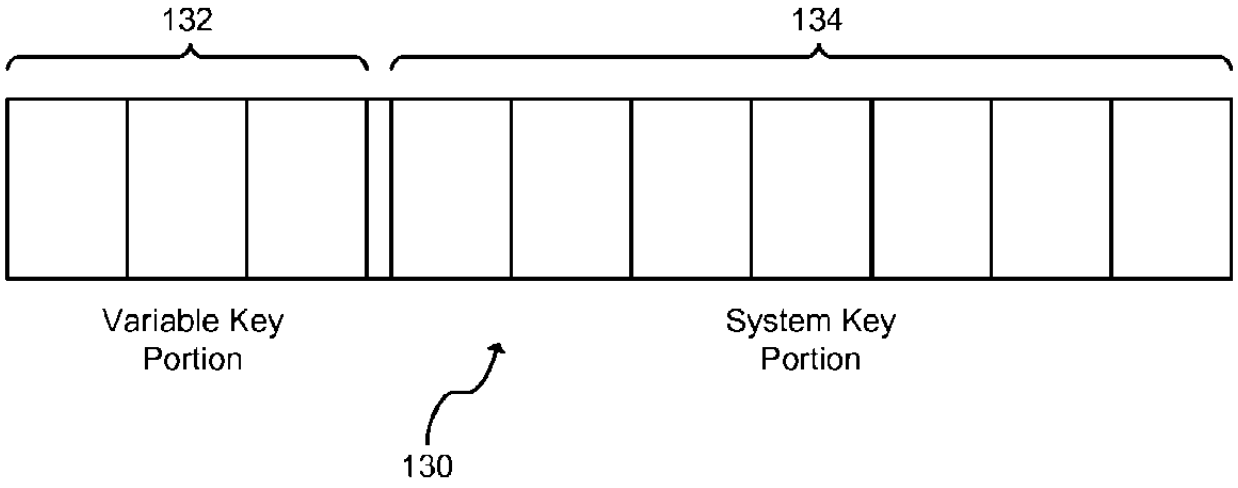


FIG. 4

Statistics | Reports | Inventory | Settings — administrator [70.169.248.194] Help Logout

Advertisers & Campaigns | Publishers & Zones | Global history

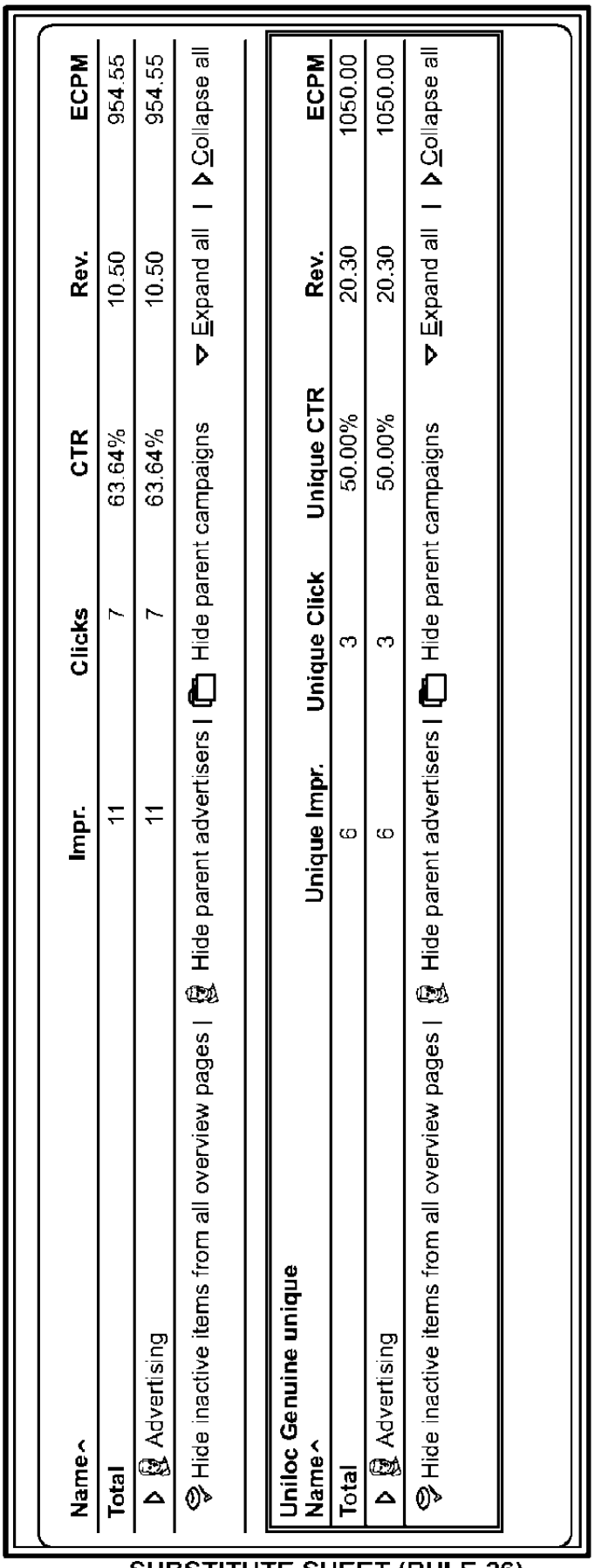
Today From 2008-04-21 To 2008-04-21

Name	Impr.	Clicks	CTR	Rev.	ECPM
Total	11	7	63.64%	10.50	954.55
▷ Advertising	11	7	63.64%	10.50	954.55

Hide inactive items from all overview pages | Hide parent advertisers | Hide parent campaigns — Expand all | Collapse all

Geniune Unique ID	Campaign	Imps.	Clicks	Conv.
B1B3B0B23FBEB5BAB1B0B2B9B...	John Deere	1	1	1
B1B3B0B23FBEB5BAB1B0B2B9B...	Aero	1	0	0
B4BDB0B538BABCBCB1BAB3B6B...	John Deere	1	0	0
B4BDB0B538BABCBCB1BAB3B6B...	Aero	1	0	0
B4BEB2B432B6B5BAB1BEB3B1B...	John Deere	1	2	2
B4BEB2B432B6B5BAB1BEB3B1B...	Aero	1	0	0
B7BAB7BE32BFB3B5BEB0B5B4B...	John Deere	1	0	0
B7BAB7BE32BFB3B5BEB0B5B4B...	Aero	1	0	0
B9BEB3B539B5B8B3B6B0B0BAB...	John Deere	1	13	2
B9BEB3B539B5B8B3B6B0B0BAB...	Aero	1	5	0

FIG. 5



Name^	Impr.	Clicks	CTR	Rev.	ECPM
Total	11	7	63.64%	10.50	954.55
▷ Advertising	11	7	63.64%	10.50	954.55

Hide inactive items from all overview pages | Hide parent advertisers | Hide parent campaigns ▽ Expand all | ▷ Collapse all

Uniloc Genuine unique Name^	Unique Impr.	Unique Click	Unique CTR	Rev.	ECPM
Total	6	3	50.00%	20.30	1050.00
▷ Advertising	6	3	50.00%	20.30	1050.00

Hide inactive items from all overview pages | Hide parent advertisers | Hide parent campaigns ▽ Expand all | ▷ Collapse all

FIG. 6

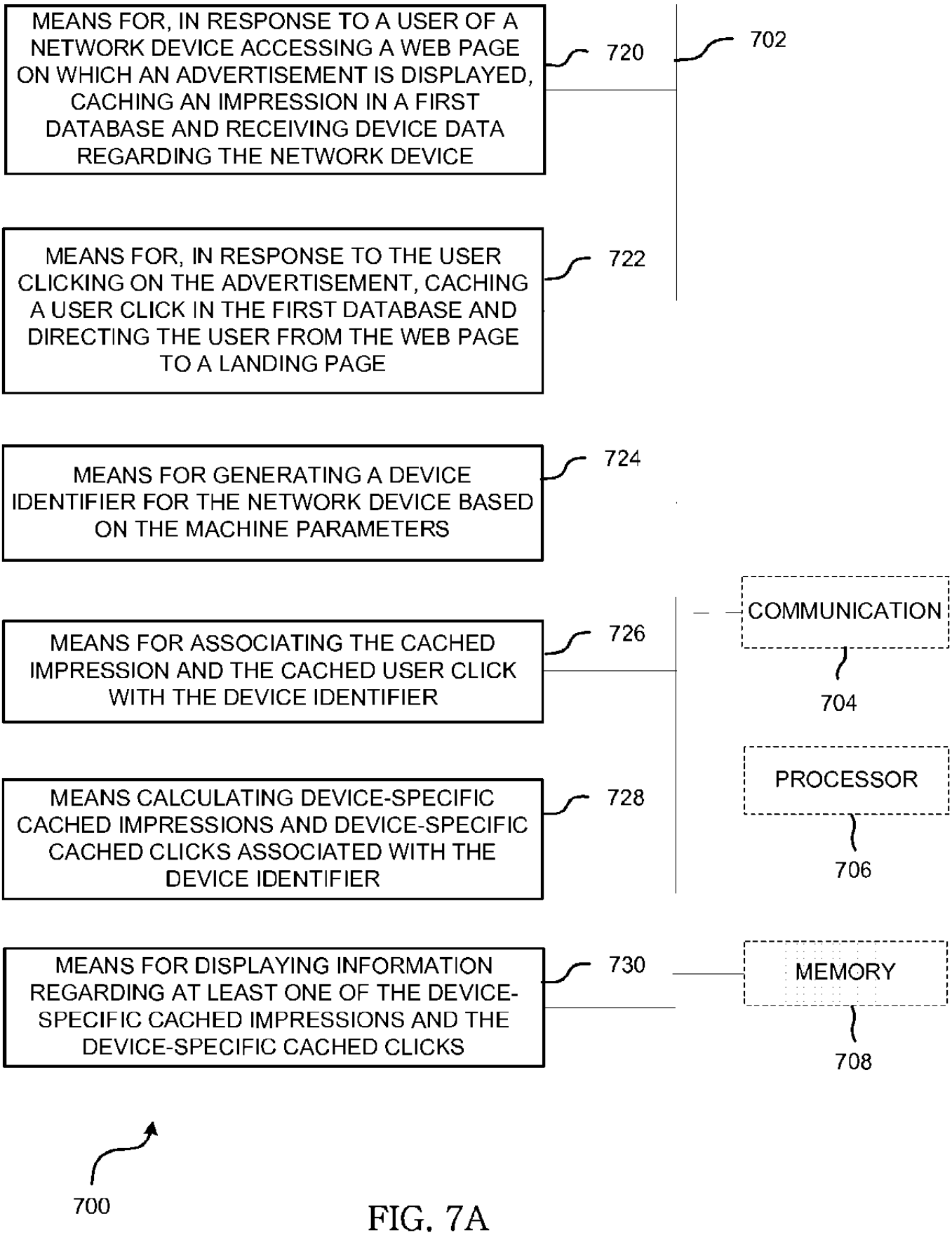

MEANS FOR, IN RESPONSE TO A USER OF A NETWORK DEVICE ACCESSING A WEB PAGE ON WHICH AN ADVERTISEMENT IS DISPLAYED, CACHING AN IMPRESSION IN A FIRST DATABASE AND RECEIVING DEVICE DATA REGARDING THE NETWORK DEVICE
720
702
MEANS FOR, IN RESPONSE TO THE USER CLICKING ON THE ADVERTISEMENT, CACHING A USER CLICK IN THE FIRST DATABASE AND DIRECTING THE USER FROM THE WEB PAGE TO A LANDING PAGE
722
MEANS FOR GENERATING A DEVICE IDENTIFIER FOR THE NETWORK DEVICE BASED ON THE MACHINE PARAMETERS
724
MEANS FOR ASSOCIATING THE CACHED IMPRESSION AND THE CACHED USER CLICK WITH THE DEVICE IDENTIFIER
726
COMMUNICATION
704
MEANS CALCULATING DEVICE-SPECIFIC CACHED IMPRESSIONS AND DEVICE-SPECIFIC CACHED CLICKS ASSOCIATED WITH THE DEVICE IDENTIFIER
728
PROCESSOR
706
MEANS FOR DISPLAYING INFORMATION REGARDING AT LEAST ONE OF THE DEVICE-SPECIFIC CACHED IMPRESSIONS AND THE DEVICE-SPECIFIC CACHED CLICKS
730
MEMORY
708
700

FIG. 7A

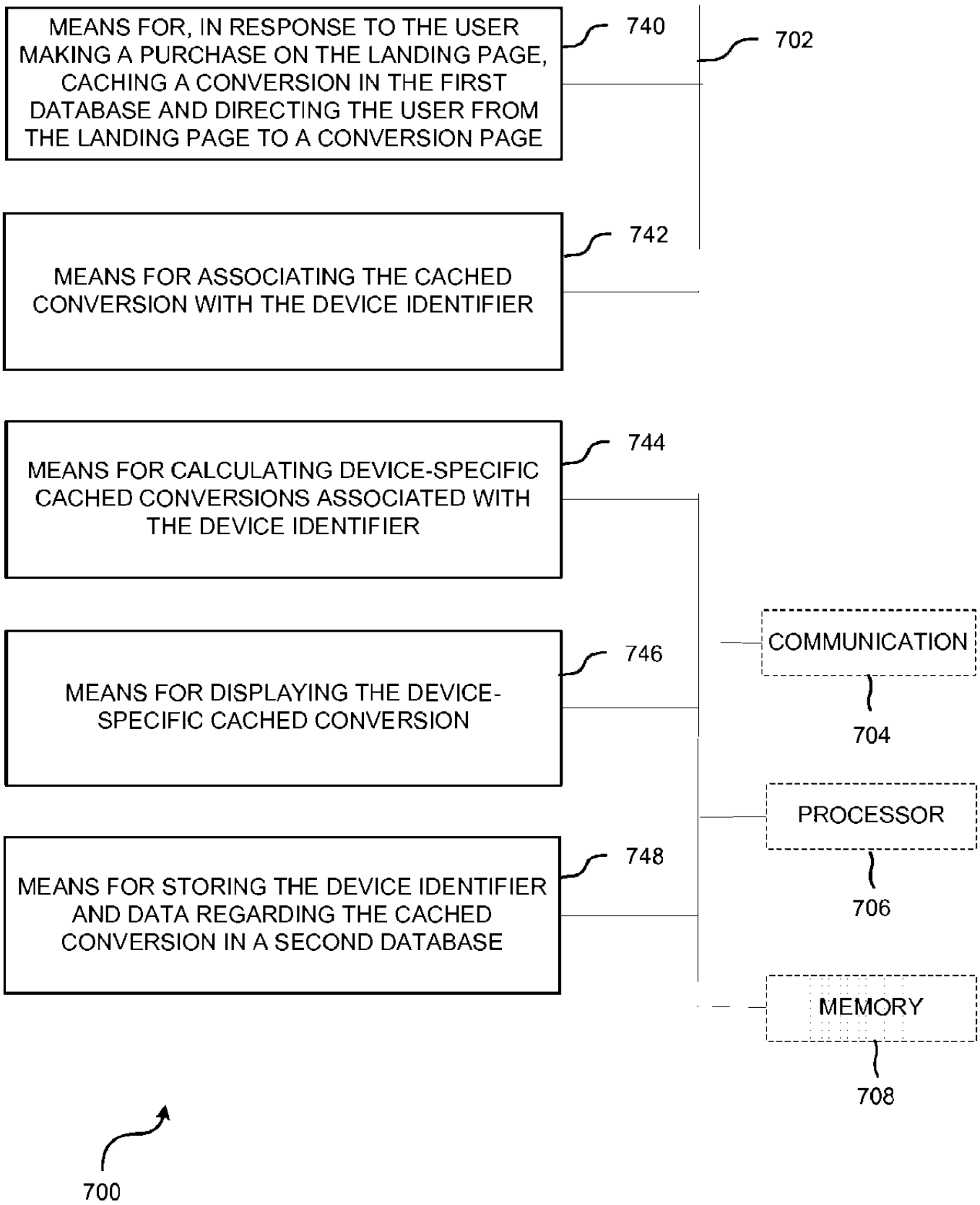

MEANS FOR, IN RESPONSE TO THE USER MAKING A PURCHASE ON THE LANDING PAGE, CACHING A CONVERSION IN THE FIRST DATABASE AND DIRECTING THE USER FROM THE LANDING PAGE TO A CONVERSION PAGE
740
702
MEANS FOR ASSOCIATING THE CACHED CONVERSION WITH THE DEVICE IDENTIFIER
742
MEANS FOR CALCULATING DEVICE-SPECIFIC CACHED CONVERSIONS ASSOCIATED WITH THE DEVICE IDENTIFIER
744
MEANS FOR DISPLAYING THE DEVICE-SPECIFIC CACHED CONVERSION
746
MEANS FOR STORING THE DEVICE IDENTIFIER AND DATA REGARDING THE CACHED CONVERSION IN A SECOND DATABASE
748
COMMUNICATION
704
PROCESSOR
706
MEMORY
708
700

FIG. 7B

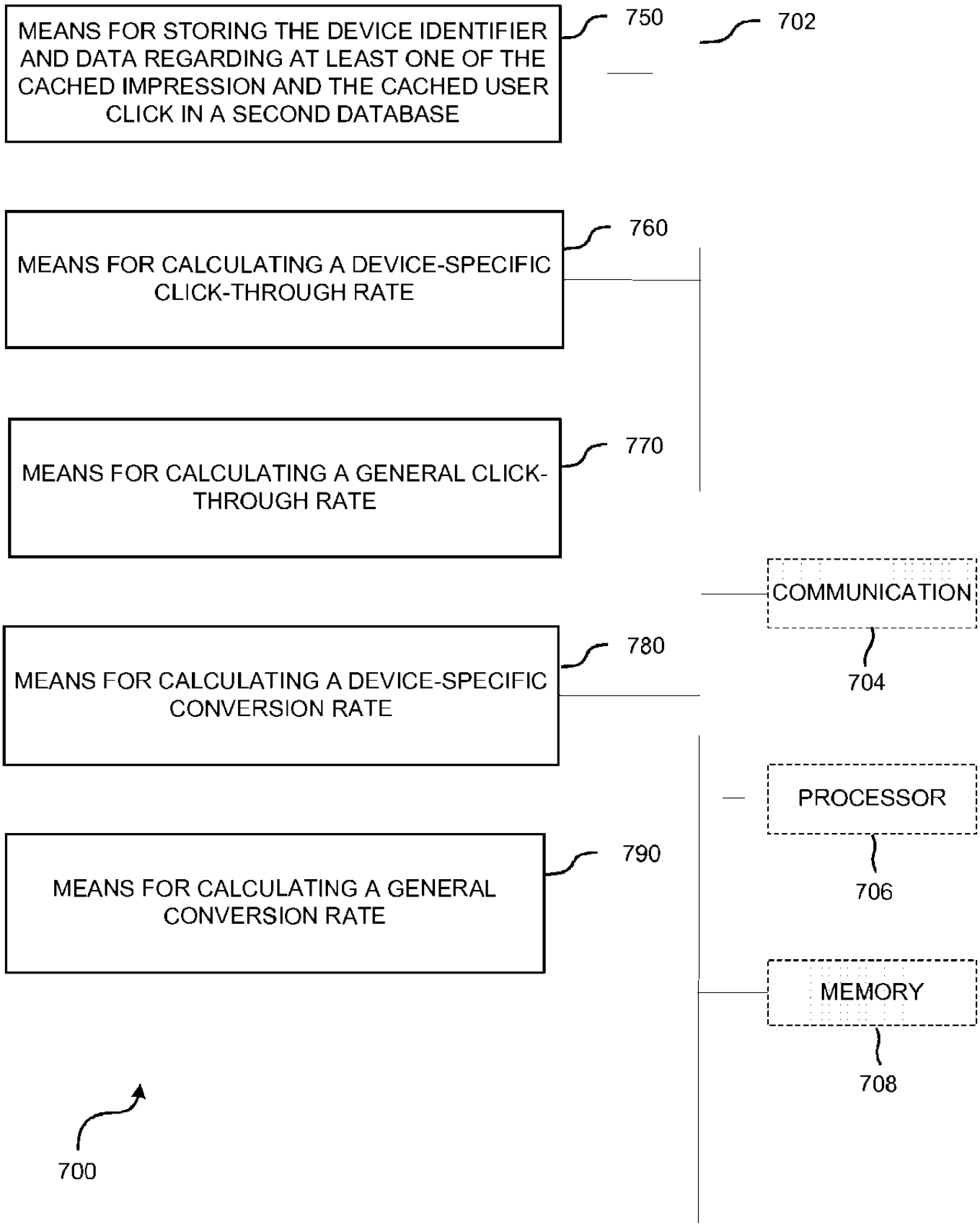

MEANS FOR STORING THE DEVICE IDENTIFIER AND DATA REGARDING AT LEAST ONE OF THE CACHED IMPRESSION AND THE CACHED USER CLICK IN A SECOND DATABASE
750
702
MEANS FOR CALCULATING A DEVICE-SPECIFIC CLICK-THROUGH RATE
760
MEANS FOR CALCULATING A GENERAL CLICK-THROUGH RATE
770
COMMUNICATION
704
MEANS FOR CALCULATING A DEVICE-SPECIFIC CONVERSION RATE
780
PROCESSOR
706
MEANS FOR CALCULATING A GENERAL CONVERSION RATE
790
MEMORY
708
700

FIG. 7C

Electronic Acknowledgement Receipt

EFS ID:	8519547
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick
Filer Authorized By:	
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	28-SEP-2010
Filing Date:	18-JUN-2010
Time Stamp:	20:09:47
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	IDS_transmittal.pdf	25378 d16ab7eee18f47d85983a09155879f9fef1ca63874	no	2

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Filed (SB/08)	sb0008ab.pdf	90819 32e307f98be98a8043f6f1105b7af9907c56bd827	no	5
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	NPL Documents	Angha.pdf	339963 cdf8a39bbb6140a90d85cdb5a15be65bc29d69550	no	7
Warnings:					
Information:					
4	Foreign Reference	AU678985_Uniloc_Corp_Pty_Ltd.pdf	2027449 dbebf6dbb63e23d388e10cbb3c80f206ca58c2a1	no	55
Warnings:					
Information:					
5	NPL Documents	Econolite.pdf	224348 bcacd43b0e73cd389401f0779f915e66a916fb7d0	no	3
Warnings:					
Information:					
6	Foreign Reference	EP1637958A2_Microsoft.pdf	2137091 3353a304829734faef4700d31adeba5e5a62aed	no	35
Warnings:					
Information:					
7	Foreign Reference	EP1637961_Microsoft.pdf	566487 4252d20fae568513f31ba657c07897337c0ecded	no	9
Warnings:					
Information:					
8	Foreign Reference	EP1670188_Alcatel.pdf	1049669 cab1887fd0ac588020fbfbe58aa6709fb1acf9e2	no	17
Warnings:					
Information:					
9	NPL Documents	Williams.pdf	1378703 dd738b47d1b03b35a4e90a6e0fcdd4b785dbf9cf	no	35
Warnings:					
Information:					
10	Foreign Reference	WO0067095_Trymedia_Systems.pdf	2775085 7b41b8f5a60938427d125fc3558ed0dc3a1ad1a7	no	90

Warnings:					
Information:					
11	Foreign Reference	WO9220022_Digital_Equipment_Corp.pdf	4692190 4de2b6e313c85c054867f729f60af5dc93da5fBd	no	137
Warnings:					
Information:					
12	Foreign Reference	WO9301550_Infologic_Software.pdf	2228694 0296906a9a07986e9c9979c261c5aae01d2ab146	no	64
Warnings:					
Information:					
13	Foreign Reference	WO9535533_Megalode_Corp.pdf	2165118 8c11a6aad1779296b038c3bb63e66a923fc5f919	no	58
Warnings:					
Information:					
14	Foreign Reference	WO2005104686_IPASS_Inc.pdf	1803937 cbaf1239853b4cb2f0f376767f6d23fd3f6ea3047	no	40
Warnings:					
Information:					
15	Foreign Reference	WO2007060516_Lo.pdf	1094275 f409fa24fbb3ad5393239c32a356d5fa93fbc087e	no	26
Warnings:					
Information:					
16	Foreign Reference	WO2008013504_Starhub.pdf	962755 c559c8a5000c9b592bc096e6b0af78dbd59c5aa2	no	22
Warnings:					
Information:					
17	Foreign Reference	WO2008157639_Uniloc_Corp.pdf	67681 b9cfe1f4cdec5bf3be0be49992770c1d524d03b9	no	1
Warnings:					
Information:					
18	Foreign Reference	WO2009039504_Uniloc_Corp.pdf	947424 90338d96bb1778307f29fd30abdf4cb9ba5f3a15	no	23
Warnings:					
Information:					
19	Foreign Reference	WO2009065135_Uniloc_Corp.pdf	1594083 20d60b6ecae1caf54361019dde160613a988e55b	no	31

Warnings:					
Information:					
20	Foreign Reference	WO2009076232_Uniloc_Corp.pdf	1332716 e131af53d3064cba9bfa1a12b5500bc52168e0d3	no	32
Warnings:					
Information:					
21	Foreign Reference	WO2009105702_Etchegoyen.pdf	1970132 239ff1abe6c9e1000204e5721680bc52e90c1ad3	no	44
Warnings:					
Information:					
22	Foreign Reference	WO2009143115_Etchegoyen.pdf	1804267 1b389e03ddc0590e1b6881582c996287878f4c58	no	38
Warnings:					
Information:					
23	Foreign Reference	WO2009158525_Etchegoyen.pdf	1433798 0e87dda222bb6c71e10c586074d631778a1aed86	no	34
Warnings:					
Information:					
		Total Files Size (in bytes):	32712062		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. no.:	12/818,906	Conf. no.	8831
Applicant:	Craig S. Etchegoyen	Art Unit:	2192
Filed:	June 18, 2010	Examiner:	not yet assigned

Title: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant hereby submits, without admission of prior art effect thereof, form(s) PTO/SB/08 pursuant to the duty of disclosure requirements of 37 CFR §§ 1.56, 1.97 and 1.98.

Applicant has listed publication dates on the attached form(s) PTO/SB/08 based on information presently available to the undersigned. However, the listed publication dates should not be construed as an admission that the information was actually published on the date indicated.

It is respectfully requested that the Examiner initial and return a copy of the enclosed forms PTO/SB/08, and to indicate in the official file wrapper of this patent application that the documents have been considered.

This Information Disclosure Statement is being filed within three months of the U.S. filing date or before the mailing date of a first Office Action on the merits, therefore no statement under 37 CFR § 1.97(e) or fee is required.

Respectfully Submitted,

/Sean D. Burdick/

Sean D. Burdick
Reg. No. 51,513

Uniloc USA, Inc.
2151 Michelson Drive, Suite 100
Irvine, CA 92612
(949) 825-5527

PTO/SB/422 (11-10)

Certification and Request to Participate in FY 2011 Pilot Program Concerning Public Submission of Peer Reviewed Prior Art	
First Named Inventor: Craig S. Etchegoyen	Application Number (if known): 12/818,906
Title of Invention: Remote Update of Computers Based on Physical Device Recognition	

Part I – Request to Participate in Pilot

Applicant provides express written consent under 35 USC 122 (c) for the USPTO to accept prior art references and comments by third party submitter, Peer To Patent, to be considered during the examination of the above identified application participating in the Peer Reviewed Prior Art Pilot Project. No other consents are provided herein.

Subject to the conditions listed below in Part II, Applicant hereby requests that the requirements of 37 CFR 1.99(d) and 37 CFR 1.291(b) be waived by the USPTO for the above-identified application participating in this pilot.

Part II – Additional Conditions Necessary For Waiver

1. **Applicant certifies the following:**

a. the above-identified application has not itself been published by the USPTO more than one month prior to the filing of this Request;

b. the above-identified application contains claimed subject matter believed to be classifiable in at least one of the classes/subclasses listed at http://www.uspto.gov/patents/init_events/class_subclasses_FY2011pilot.jsp;

c. Applicant agrees to full participation in the pilot program; and

d. (including this request) Applicant or Applicant's assignee has not submitted more than 25 applications for participation in the Peer Reviewed Prior Art Pilot Project.

2. **Through submission of this form, Applicant provides consent to the following:**

a. the submission of up to six patents or publications by the third party submitter (Peer To Patent) and comments describing the relevance of the document to the disclosed invention; and

b. the filing of the submission no later than eighteen (18) weeks after publication of the above-identified Application.

Note: Applicant shall not be required to submit an Information Disclosure Statement under 37 CFR 1.97 and 1.98 to ensure that the references submitted by third party submitter (Peer To Patent) are considered by the examiner. No fees are required in making this request.

Signature /Sean D. Burdick/	Date December 28, 2010
Name (Printed/Typed) Sean D. Burdick	Registration Number 51,513
Telephone No. 949-825-5527	E-mail sean.burdick@unilocusa.com

Note: *Signature of all the inventors or assignees of record of the entire interest or their representative(s) are required in accordance with 37 CFR 1.33 and 11.18. Please see 37 CFR 1.4(d) for the form of the signature. If necessary, submit multiple forms for more than one signature, see below*.*

☐ *Total of _____ signed forms are submitted.

SUBMIT COMPLETED FORM by EFS-Web, or by FAX to 571-273-8300.

Electronic Acknowledgement Receipt

EFS ID:	9127653
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick/Amanda Ivey
Filer Authorized By:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	28-DEC-2010
Filing Date:	18-JUN-2010
Time Stamp:	18:28:38
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Peer Review Pilot	AD-037_Peer_Pilot_Program_sb0422.pdf	38764 80063fafcf6e48b9098f9d72c321a2c7792e68d6	no	1

Warnings:	
Information:	

Total Files Size (in bytes):	38764

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National 5tage of an International Application under 35 U.5.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the U5PTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037

CONFIRMATION NO. 8831

96051
Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612

PUBLICATION NOTICE

OC000000045214045

Title: Remote Update of Computers Based on Physical Device Recognition

Publication No.US-2010-0333081-A1
Publication Date:12/30/2010

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Office of Public Records. The Office of Public Records can be reached by telephone at (703) 308-9726 or (800) 972-6382, by facsimile at (703) 305-8759, by mail addressed to the United States Patent and Trademark Office, Office of Public Records, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently http://pair.uspto.gov/. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Managment, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

Doc Code: PEER.SB423
Document Description: Third Party Submission – Peer Review

PTO/SB/423 (03-11)

THIRD-PARTY SUBMISSION OF PATENTS OR PUBLICATIONS UNDER PEER REVIEW PILOT PROGRAM

Complete if Known	
Application Number	12/818,906
Filing Date	06-18-2010
First Named Inventor	Craig Stephen Etchegoyen
Art Unit	2192
Examiner Name	-
Attorney Docket Number	UN-NP-AD-037

DO NOT SUBMIT THIS FORM VIA EFS-WEB

A signed certification and request to participate in the Peer Review Pilot Program (Form PTO/SB/422) was filed on (insert date): 12-28-2010.

NOTE: A submission received by the Office prior to the filing of a signed certification and request to participate in the Peer Review Pilot Program (Form PTO/SB/422) will be returned to the Third-Party Submitter and will NOT be placed into the Application File.

U.S. PATENT DOCUMENTS

Cite No.	Check Box If Comments Attached	Document Number Number-Kind Code[1] (If known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Examiner Initials*
1	☐	US- 6327617	12-04-2001	Philip Fawcett	
2	☐	US- 6880086	04-12-2005	Joseph Kidder	
3	☐	US-			
4	☐	US-			
5	☐	US-			
6	☐	US-			

FOREIGN PATENT DOCUMENTS

Cite No.	Check Box If Comments Attached	Document Number Country Code[2]-Number[3]-Kind Code[4] (If known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Examiner Initials*
1	☐				
2	☐				
3	☐				
4	☐				
5	☐				
6	☐				

[1] See Kind Codes of USPTO patent documents at www.uspto.gov/patents/ebc/kindcodesum.jsp or MPEP 901.04(a).
[2] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3).
[3] For Japanese patent documents, the indication of the year of the Emperor must precede the serial number of the patent document.
[4] Kind of document by the appropriate symbols, as indicated on the document under WIPO Standard ST.16, if possible.
*EXAMINER: Initial each citation that is considered. Draw a line through each citation that is not considered. Include a copy of this form with next communication to applicant.

NON PATENT LITERATURE DOCUMENTS			
Cite No.	Check Box If Comments Attached	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.) date, page(s), volume-issue number(s), publisher, city and/or country where published.	Examiner Initials*
1	☐		
2	☐		
3	☐		
4	☐		
5	☐		
6	☐		

My signature below certifies that I am authorized to make the enclosed submission as part of the Peer Review Pilot Program.

Signature of Third-Party Submitter	s/		Date 4-11-2011
Name (Printed/Typed)	Andrea Casillas	Reg. No., if applicable	Telephone Number 212-431-2368
Address	185 W. Broadway New York, NY 10013		

Examiner Signature		Date Considered	

*EXAMINER: Initial each citation that is considered. Draw a line through each citation that is not considered. Include a copy of this form with next communication to applicant.

PLUS Search Results for S/N 12818906, Searched Wed Aug 10 12:45:55 EDT 2011

The Patent Linguistics Utility System (PLUS) is a USPTO automated search system for U.S. Patents from 1971 to the present PLUS is a query-by-example search system which produces a list of patents that are most closely related linguistically to the application searched. This search was prepared by the staff of the Scientific and Technical Information Center, SIRA.

5655152 99
6098099 98
6839564 97
7146389 97
7555303 97
7610387 97
7904622 97
20020161769 97
20040044698 97
20050125459 97
20070043793 97
20070276962 97
20070288548 97
20090300066 97
20100199016 97
20100333081 97
5983241 79
5995999 79
6178519 79
5528757 78
5675752 78
5699517 78
5745904 78
5796966 78
5809543 78
5809527 78
5884301 78
5974409 78
5987376 78
5999947 78
6041362 78
6044399 78
6061799 78
6061799 78
6094721 78
6138120 78
6151708 78
6157953 78
6199762 78
6202085 78
6253188 78
6301612 78
6311209 78
6317754 78
6330560 78
6336115 78
6442549 78
6560636 78
6789255 78
6954737 78

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

96051 7590 08/12/2011

Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
08/12/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	12/818,906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	Art Unit
	QING CHEN	2191

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *18 June 2010*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *18 June 2010* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20100928*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This is the initial Office action based on the application filed on June 18, 2010.

2. **Claims 1-20** are pending.

3. This application is participating in the Peer Reviewed Prior Art Pilot Project.

Claim Objections

4. **Claims 2, 3, and 5** are objected to because of the following informalities:

- Claims 2 and 3 recite the limitation "the unique device identifier." It should read -- the unique device *identifiers* --.
- Claim 5 recites the limitation "the unique identifiers." It should read -- the unique *device* identifiers --.

Appropriate correction is required.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. **Claims 18-20** are rejected under 35 U.S.C. 102(b) as being anticipated by US 6,467,088 (hereinafter "alSafadi").

As per Claim 18, alSafadi discloses:

A method for remote update of a program (col. 6 lines 44-46, "A processor-implemented method for controlling the reconfiguration of an electronic device ..."), **comprising:**

- **collecting, at an update server, unique identifiers from at least one of an audit server and client device** (col. 4 lines 39-47, "... the reconfiguration manager 10 [update server] obtains [collects] information regarding the hardware and software configuration of device X, i.e., electronic device 12 [client device] of FIG. 1 ... In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier [unique identifiers] of the electronic device.");
- **analyzing the unique identifiers** (col. 5 lines 8-12, "If step 112 indicates [analyzing] that the set is not empty, a particular set of upgrade configuration is selected in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X.");
- **determining an updated program configuration for the client device based on the analyzed unique identifiers** (col. 5 lines 8-12, "If step 112 indicates that the set is not empty, a particular set of upgrade configuration is selected [determined] in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X."); **and**
- **delivering the updated program configuration to the client device** (col. 5 lines 16-18, "The reconfiguration manager or other server associated therewith then downloads [delivers] the upgrade to device X in step 120.").

As per Claim 19, the rejection of Claim 18 is incorporated; and alSafadi further discloses:

- **wherein the determining step comprises the update server comparing each analyzed unique identifier to known identifiers stored in a database to determine whether a match exists** (col. 2 lines 37-41, "The reconfiguration manager then compares the needed and currently implemented components with previously-stored lists of known acceptable and unacceptable configurations for the electronic device (emphasis added).").

As per Claim 20, the rejection of Claim 19 is incorporated; and alSafadi further discloses:

- **wherein the determining step further comprises the update server generating the updated program configuration as data representing all matches yielded by the comparing step** (col. 4 lines 62-64, "In step 104 of FIG. 2, the reconfiguration manager 10 generates a potential upgrade configuration that will satisfy the received request (emphasis added).").

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. **Claims 1-17** are rejected under 35 U.S.C. 103(a) as being unpatentable over alSafadi in view of US 6,880,086 (hereinafter "Kidder").

As per Claim 1, alSafadi discloses:

A system for remotely updating a program configuration (col. 7 lines 46 and 47, "An apparatus for controlling the reconfiguration of an electronic device …"), **comprising:**

- **a client device** (Figure 1: 12) **configured to execute a computer program to perform a remote update, the client device comprising:**

- **a first transceiver configured to send the unique device identifiers to at least one server via Internet** (col. 4 lines 42-47, "This information is generally included as part of the request 20 sent by the device 12 to the reconfiguration manager 10. In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier [unique device identifiers] of the electronic device (emphasis added)."; col. 6 lines 1-3, "The network 214 may represent a global computer communications network such as the Internet …"); **and**

- **an update server** (Figure 1: 10) **configured to collect the unique device identifiers from at least one client device** (col. 4 lines 39-47, "… the reconfiguration manager 10 obtains [collects] information regarding the hardware and software configuration of device X, i.e., electronic device 12 of FIG. 1 … In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier [unique device identifiers] of the electronic device."), **the update server comprising:**

- **a second processor coupled to memory and configured to analyze the unique device identifiers at the update server, and to determine based on the analyzed unique device identifiers an updated program configuration** (col. 5 lines 8-12, "If step 112 indicates [analyzes] that the set is not empty, a particular set of upgrade configuration is selected

[determined] in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X."); **and**

- **a second transceiver configured to deliver via the Internet data representing the updated program configuration to the client device for storage therein** (col. 5 lines 16-18, "The reconfiguration manager or other server associated therewith then downloads [delivers] the upgrade to device X in step 120.").

alSafadi does not explicitly disclose:

- **a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates unique device identifiers for the client device, the unique device identifiers based at least in part on the determined machine parameters.**

However, Kidder discloses:

- **a first processor coupled to memory storing a computer program which, when executed by the processor (i) performs physical device recognition on a client device to determine machine parameters** (col. 61 lines 9-13, "Also within modular system services is a Master Control Driver (MCD) that learns the physical characteristics of the particular computer system on which it is running, in this instance, computer system 10."), **and (ii) generates unique device identifiers for the client device, the unique device identifiers based at least in part on the determined machine parameters** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates unique device identifiers for the client device, the unique device identifiers based at least in part on the determined machine parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 2, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the unique device identifier comprises a hash code.**

However, Kidder discloses:

- **wherein unique identifier comprises a hash code** (col. 88 lines 62-64, "To avoid versioning errors, instead of assigning a version number, a signature is "machine generated" based on the content of the software component."; col. 89 lines 15-17, "The Sha-1 utility is a secure hash algorithm that uses the contents of a software component to generate a signature that is 20 bytes in length.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the unique device identifier comprises a hash code. The modification would be obvious

because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 3, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.**

However, Kidder discloses:

- **wherein a computer program when executed implements at least one irreversible transformation such that machine parameters cannot be derived from a unique device identifier** (col. 89 lines 5-7, "In one embodiment, the signatures are generated using the "Sha-1" cryptography utility (often called the "sha1sum").").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier. The modification would be obvious because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 4, the rejection of Claim 3 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the at least one irreversible transformation comprises a cryptographic hash function.**

However, Kidder discloses:

- **wherein at least one irreversible transformation comprises a cryptographic hash function** (col. 89 lines 5-7, "In one embodiment, the signatures are generated using the "Sha-1" cryptography utility (often called the "sha1sum").").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the at least one irreversible transformation comprises a cryptographic hash function. The modification would be obvious because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 5, the rejection of Claim 1 is incorporated; and alSafadi further discloses:

- **wherein the unique identifiers further comprise software identifiers** (col. 4 lines 15-19, "The request in the illustrative embodiment also includes a list of the components currently in the device, i.e., version 1.1 of component A, version 2.0 of component C and version 2.3 of component B.").

alSafadi does not explicitly disclose:

- **wherein the unique identifiers further comprise geo-location identifiers.**

However, Kidder discloses:

- **wherein unique identifiers further comprise geo-location identifiers** (col. 20 lines 1-4, "To configure a network device, the administrator begins by selecting (step 874, FIG. 3g) a particular network device to configure, for example, the network device corresponding to IP address 192.168.9.202 (FIG. 4f).").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the unique identifiers further comprise geo-location identifiers. The modification would be obvious because one of ordinary skill in the art would be motivated to access the client device using the Internet.

As per Claim 6, the rejection of Claim 5 is incorporated; and alSafadi does not explicitly disclose:

- **wherein at least one of the geo-location identifiers comprises an Internet Protocol address of the client device.**

However, Kidder discloses:

- **wherein at least one of the geo-location identifiers comprises an Internet Protocol address of a client device** (col. 20 lines 1-4, "To configure a network device, the administrator begins by selecting (step 874, FIG. 3g) a particular network device to configure, for example, the network device corresponding to IP address 192.168.9.202 (FIG. 4f).").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include

wherein at least one of the geo-location identifiers comprises an Internet Protocol address of the client device. The modification would be obvious because one of ordinary skill in the art would be motivated to access the client device using the Internet.

As per Claim 7, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model number) pertaining to the computer system.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: a machine

model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 8, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model) pertaining to the CPU.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU

model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 9, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model) pertaining to the optical drive.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model. The

modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 10, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., manufacturer) pertaining to the baseboard/motherboard.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag. The modification would be obvious because one of ordinary skill in the art

would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 11, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., manufacturer) pertaining to the chassis.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 12, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the IDE controller.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 13, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the port connector.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 14, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., size) pertaining to the cache.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 15, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub** (col. 61

lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the USB drive.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 16, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that

the physical inventory of the computer system includes information pertaining to the device model.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 17, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.")**.**

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the PCI controller.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

Conclusion

9. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

10. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Anna Deng/

Primary Examiner, Art Unit 2191

Notice of References Cited	Application/Control No. 12/818,906	Applicant(s)/Patent Under Reexamination ETCHEGOYEN, CRAIG STEPHE	
	Examiner QING CHEN	Art Unit 2191	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2003/0014745 A1	01-2003	Mah et al.	717/170
*	B	US-2003/0195995 A1	10-2003	Tabbara, Bassam	709/313
*	C	US-2005/0034115 A1	02-2005	Carter et al.	717/173
*	D	US-2005/0262498 A1	11-2005	Ferguson et al.	717/172
*	E	US-2007/0169087 A1	07-2007	Fadell, Anthony M.	717/168
*	F	US-5,155,847	10-1992	Kirouac et al.	709/221
*	G	US-6,327,617 B1	12-2001	Fawcett, Philip E.	709/219
*	H	US-6,467,088 B1	10-2002	alSafadi et al.	717/173
*	I	US-6,880,086 B2	04-2005	Kidder et al.	713/191
*	J	US-7,200,237 B2	04-2007	Zhang et al.	381/60
*	K	US-7,577,948 B2	08-2009	Zomaya et al.	717/168
*	L	US-7,676,804 B2	03-2010	Ferguson et al.	717/173
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	**Art Unit**
	QING CHEN	2191

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/11/2011								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	Art Unit
	QING CHEN	2191

SEARCHED			
Class	Subclass	Date	Examiner

SEARCH NOTES		
Search Notes	Date	Examiner
EAST Search (US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB)	8/10/2011	/QC/
717/168-178 (limited classification search using keywords)	8/10/2011	/QC/
EAST Inventor Name Search and Assignee Search (US-PGPUB; USPAT)	8/10/2011	/QC/
NPL Search (ACM)	8/10/2011	/QC/
PLUS Search	8/10/2011	/QC/
PALM Inventor Name Search	8/10/2011	/QC/

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	52	CRAIG near ETCHEGOYEN.in.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:03
S2	1	S1 and (remote near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:04
S3	75	("20060072444" | "20080086423" | "20080320607" | "5414269" | "5754763" | "6044471" | "7188241" | "20040030912" | "20060265337" | "20070168288" | "20070219917" | "6233567" | "6294793" | "6330670" | "7203966" | "7327280" | "7337147" | "20010034712" | "20020082997" | "20040187018" | "20090138975" | "5440635" | "5490216" | "5745879" | "6009401" | "6158005" | "6449645" | "6920567" | "7272728" | "7319987" | "0000000" | "20020161718" | "20040143746" | "20070198422" | "20070282615" | "5291598" | "5418854" | "5666415" | "6230199" | "6785825" | "7032110" | "7206765" | "20030065918" | "20030172035" | "20040024860" | "20040059929" | "20060282511" | "20070203846" | "20080065552" | "4658093" | "4796220" | "6243468" | "6976009" | "7069595" | "20020019814" | "20050172280" | "20080147556" | "20080228578" | "5925127" | "7463945" | "4351982" | "4704610" | "5210795" | "6536005" | "6859793" | "7085741" | "7343297" | "7653899" | "20050108173" | "20050138155" | "20060095454" | "20060161914" | "20090083730" | "5790664" | "5974150" | "7069440").PN.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:05
S5	1	S3 and (remote near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:06
S6	2	S3 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:06
S7	2	("6327617" "6880086").pn.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:08
S8	14	uniloc.as.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:35
S9	0	S8 and (remote$2 near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2011/08/10 11:35
S10	49	("5655152" "6098099" "6839564" "7146389" "7555303" "7610387" "7904622" "20020161769" "20040044698" "20050125459" "20070043793" "20070276962" "20070288548" "20090300066"	US-PGPUB; USPAT	OR	OFF	2011/08/10 13:03

		"20100199016" "20100333081" "5983241" "5995999" "6178519" "5528757" "5675752" "5699517" "5745904" "5796966" "5809543" "5809527" "5884301" "5974409" "5987376" "5999947" "6041362" "6044399" "6061799" "6061799" "6094721" "6138120" "6151708" "6157953" "6199762" "6202085" "6253188" "6301612" "6311209" "6317754" "6330560" "6336115" "6442549" "6560636" "6789255" "6954737").pn.				
S11	6879	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:09
S12	6879	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:30
S13	599	S12 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:30
S14	47	S13 and (device near3 identifier)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:30
S15	43	S14 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:31
S16	1	"5155847".pn.	US-PGPUB; USPAT	OR	OFF	2011/08/10 13:39
S17	41	("5155847" | "5253344" | "5327560" | "5497490" | "5634075" | "5822531" | "5898872" | "5918194" | "5933026" | "6058455" | "6065068" | "6167408" | "6301707" | "6385668").PN. OR ("6467088").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2011/08/10 13:41
S18	7	S17 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:41
S19	17108	(remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO;	OR	OFF	2011/08/10 13:44

			DERWENT; IBM_TDB			
S20	749	S19 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:44
S21	109	S20 and (hash$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:44
S22	75	S21 and ((updat$4 or upgrad$4) near3 server)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:44
S23	74	S22 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:45
S24	52	S23 and configuration	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:45
S25	41	S23 and (device near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 13:47
S26	50	S12 and (hardware near3 profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 15:57
S27	49	S26 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 15:57
S28	11	S27 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 15:57
S29	20	S12 and (hardware near3 parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO;	OR	OFF	2011/08/10 16:07

			DERWENT; IBM_TDB			
S30	22	S12 and (machine near3 parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:16
S31	5	S12 and (determin$4 with (machine near3 parameter))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:21
S32	20	(hardware near3 profile) with identifier	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:23
S33	18	S32 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:23
S34	2	(device near identifier) with (hardware near profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:25
S35	0	(device near identifier) with (hardware near parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:28
S36	2	S29 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/08/10 16:29

8/ 11/ 2011 4:01:22 PM
C:\ Users\ qchen\ Documents\ EAST\ Workspaces\ docket\ 12818906.wsp

ACM DIGITAL LIBRARY Us Patent & Trademark Office

SIGN IN SIGN UP

remotely updating device SEARCH

Searching for: remotely updating device (start a new search)

Found **881** within *The ACM Guide to Computing Literature* (Bibliographic citations from major publishers in computing)

Limit your search to Publications from ACM and Affiliated Organizations (Full-Text collection: **312,030** items)

REFINE YOUR SEARCH

Refine by Keywords
remotely updating devi
Discovered Terms

Refine by People
Names
Institutions
Authors
Editors
Reviewers

Refine by Publications
Publication Year
Publication Names
ACM Publications
All Publications
Content Formats
Publishers

Refine by Conferences
Sponsors
Events
Proceeding Series

ADVANCED SEARCH
Advanced Search

FEEDBACK
Please provide us with feedback

Found **881** of **1,697,209**

Search Results Related Journals Related Magazines Related SIGs Related Conferences

Results 1 - 20 of 881 Sort by relevance in expanded for

Result page: **1** 2 3 4 5 6 7 8 9 10 next

1 MeshUp: reliably evolving a living lab

Johnathan Ishmael, Fabian Hugelshofer, Paul Smith, Nicholas J.P. Race

September 2010 **WiNTECH '10:** Proceedings of the fifth ACM international workshop on **Wireless network testbeds** experimental evaluation and characterization

Publisher: ACM Request Permissions

Full text available: Pdf (438.47 KB)

Bibliometrics: Downloads (6 Weeks): 3, Downloads (12 Months): 31, Downloads (Overall): 31, Citation Count: 0

Several benefits can be derived from having a user community associated with an experimental wireless netwo such as access to real user's network traffic. However, to ensure continued use of the network, it must provide acceptable levels of service. ...

Keywords: failsafe, mesh, software update, wireless

2 ELMR: efficient lightweight mobile records

Arvind Kumar, Jay Chen, Michael Paik, Lakshminarayanan Subramanian

August 2009 **MobiHeld '09:** Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds

Publisher: ACM

Full text available: Pdf (277.87 KB)

Bibliometrics: Downloads (6 Weeks): 0, Downloads (12 Months): 25, Downloads (Overall): 74, Citation Count: 0

In this paper we describe Efficient Lightweight Mobile Records (ELMR), a system that provides a practical protoc for accessing and updating database records remotely from low-end mobile devices using the 140-byte SMS channel.

Keywords: cell phones, compression, healthcare, user interface

3 ELMR: lightweight mobile health records

Arvind Kumar, Amey Purandare, Jay Chen, Arthur Meacham, Lakshminarayanan Subramanian

June 2009 **SIGMOD '09:** Proceedings of the 35th SIGMOD international conference on Management of da

Publisher: ACM

Full text available: Pdf (405.87 KB)

Bibliometrics: Downloads (6 Weeks): 5, Downloads (12 Months): 80, Downloads (Overall): 277, Citation Count: 0

Cell phones are increasingly being used as common clients for a wide suite of distributed, database-centric healthcare applications in developing regions. This is particularly true for rural developing regions where the bu the healthcare is handled ...

Keywords: cell phones, rural healthcare, user interface

4 Remote Service Deployment on Programmable Switches with the IETF SNMP Script MIB

Jürgen Quittek, Cornelia Kappler

October 1999 **DSOM '99:** Proceedings of the 10th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Active Technologies for Network and Service Management

Publisher: Springer-Verlag

Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count:

Some approaches to network programmability require control processes for network devices. Such control processes are executed at the device or at a control node locally connected to the device. This paper discusses management of these control processes, ...

5 Secure automotive on-board protocols: a case of over-the-air firmware updates

Muhammad Sabir Idrees, Hendrik Schweppe, Yves Roudier, Marko Wolf, Dirk Scheuermann, Olaf Henniger

March 2011 **Nets4Cars/ Nets4Trains'11:** Proceedings of the Third international conference on Communication technologies for vehicles

Publisher: Springer-Verlag

Full text available: Publisher Site

Bibliometrics: Downloads (6 Weeks): n/a. Downloads (12 Months): n/a. Downloads (Overall): n/a. Citation Count:

The software running on electronic devices is regularly updated, these days. A vehicle consists of many such devices, but is operated in a completely different manner than consumer devices. Update operations are safety critical in the automotive domain. ...
Keywords: over the air firmware updates, security architectures, security protocols, software functionality

6 Cimbiosys: a platform for content-based partial replication
Venugopalan Ramasubramanian, Thomas L. Rodeheffer, Douglas B. Terry, Meg Walraed-Sullivan, Ted Wobber, Catherine C. Marshall, Amin Vahdat

April 2009 **NSDI '09:** Proceedings of the 6th USENIX symposium on Networked systems design and implementa
Publisher: USENIX Association

Bibliometrics: Downloads (6 Weeks): n/a. Downloads (12 Months): n/a. Downloads (Overall): n/a. Citation Count:

Increasingly people manage and share information across a wide variety of computing devices from cell phones Internet services. Selective replication of content is essential because devices, especially portable ones, have limited resources for storage ...

7 What goes where?: designing interactive large public display applications for mobile device interaction
Nima Kaviani, Matthias Finke, Sidney Fels, Rodger Lea, Hua Wang

November 2009 **ICIMCS '09:** Proceedings of the First International Conference on Internet Multimedia Computing Service

Publisher: ACM Request Permissions

Full text available: Pdf (1.83 MB)

Bibliometrics: Downloads (6 Weeks): 1. Downloads (12 Months): 62, Downloads (Overall): 85, Citation Count: 0

In this paper we describe our current research in the field of interactive large public displays with mobile device interaction. The core of our research is based on a user interface concept that takes advantage of input and out capabilities of interactive ...
Keywords: conflict, dual display, engagement, large public displays, mobile interaction

8 NodeMD: diagnosing node-level faults in remote wireless sensor systems
Veljko Krunic, Eric Trumpler, Richard Han

June 2007 **MobiSys '07:** Proceedings of the 5th international conference on Mobile systems, applications and services

Publisher: ACM Request Permissions

Full text available: Pdf (1.87 MB)

Bibliometrics: Downloads (6 Weeks): 5. Downloads (12 Months): 40, Downloads (Overall): 396, Citation Count: 12

Software failures in wireless sensor systems are notoriously difficult to debug. Resource constraints in wireless deployments substantially restrict visibility into the root causes of node-level system and application faults. At t same time, the high ...

Keywords: deployment, diagnosis, software fault, wireless sensor networks

9 A dynamic operating system for sensor nodes
Chih-Chieh Han, Ram Kumar, Roy Shea, Eddie Kohler, Mani Srivastava

June 2005 **MobiSys '05:** Proceedings of the 3rd international conference on Mobile systems, applications, a services

Publisher: ACM Request Permissions

Full text available: Pdf (418.07 KB)

Bibliometrics: Downloads (6 Weeks): 16. Downloads (12 Months): 159. Downloads (Overall): 1956, Citation Count

Sensor network nodes exhibit characteristics of both embedded systems and general-purpose systems. They m use little energy and be robust to environmental conditions, while also providing common services that make it to write applications. In ...

10 Ubiquitous device personalization and use: The next generation of IP multimedia communications
Ron Shacham, Henning Schulzrinne, Srisakul Thakolsri, Wolfgang Kellerer

May 2007 **Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)** , Vo 3 Issue 2

Publisher: ACM Request Permissions

Full text available: Pdf (301.74 KB)

Bibliometrics: Downloads (6 Weeks): 2. Downloads (12 Months): 81, Downloads (Overall): 1094, Citation Count: 1

Service usage in emerging ubiquitous environments includes seamless and personalized usage of public and pri devices discovered in the vicinity of a user. In our work, we describe an architecture for device discovery, devic configuration, and the ...

Keywords: Internet multimedia, Location-based services, mobile communications, ubiquitous computing

11 Class notes: don't be a WIMP: (http://www.not-for-wimps.org)
Johannes Itten, Dirk Reiners
August 2008 **SIGGRAPH '08:** SIGGRAPH 2008 classes
Publisher: ACM Request Permissions
Full text available: Flv (93:37 MIN), Mov (93:28 MIN), PDF (6.40 MB)
Bibliometrics: Downloads (6 Weeks): 14, Downloads (12 Months): 155, Downloads (Overall): 807, Citation Count:

Virtual and augmented reality have been around for a long time, but for most people they are movie fantasies. few people outside a few research labs have worked with or experienced these systems for themselves. On the other hand, interactive 3D ...

12 OMA DM-based remote software fault management for mobile devices
Joon-Myung Kang, Hong-Taek Ju, Mi-Jung Choi, James Won-Ki Hong, Jun-Gu Kim
November 2009 **International Journal of Network Management**, Volume 19 Issue 6
Publisher: John Wiley & Sons, Inc.
Full text available: Pdf (1.10 MB)
Bibliometrics: Downloads (6 Weeks): 1, Downloads (12 Months): 6, Downloads (Overall): 16, Citation Count: 0

Mobile devices (e.g. mobile handsets or PDAs) have gained much functionality and intelligence with the growth mobile network technologies and the increased use of mobile services. As a consequence, mobile devices have become more complex and many ...

13 OmniStore: Automating data management in a personal system comprising several portable devices
Alexandros Karypidis, Spyros Lalis
October 2007 **Pervasive and Mobile Computing**, Volume 3 Issue 5
Publisher: Elsevier Science Publishers B. V.
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count:

The ever growing amount of data generated and consumed on the move using portable devices gives rise to se data management issues. The fact that each person may own and is likely to carry several such devices further aggravates this problem. On ...

Keywords: Distributed systems, File management, Mobile computing, Personal area networks, Pervasive and ubiquitous computing

14 Distributed interface bits: dynamic dialogue composition from ambient computing resources
Anthony Savidis, Constantine Stephanidis
May 2005 **Personal and Ubiquitous Computing**, Volume 9 Issue 3
Publisher: Springer-Verlag
Full text available: Pdf (1.70 MB)
Bibliometrics: Downloads (6 Weeks): 13, Downloads (12 Months): 41, Downloads (Overall): 527, Citation Count: 9

This paper discusses a particular issue in the context of disappearing computing, namely, user mobility. Mobile users may carry with them a variety of wireless gadgets while being immersed in a physical environment encompassing numerous computing devices. ...

Keywords: Abstract dialogue elements, Adaptive interaction, Ambient dialogues, Dynamic user interface composition, Wearable interfaces

15 Digital rights management using a master control device
Imad M. Abbadi
December 2007 **ASIAN'07:** Proceedings of the 12th Asian computing science conference on Advances in computer science: computer and network security
Publisher: Springer-Verlag
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count:

This paper focuses on the problem of preventing the illegal copying of digital content whilst allowing content mobility within a single user domain. This paper proposes a novel solution for binding a domain to a single own Domain owners are authenticated ...

16 Multimedia room bridge adapter for seamless interoperability between heterogeneous home network dev
Sung He Park, Myung Jin Lee, Soon Ju Kang
January 2008 **MG '06:** Proceedings of the 15th ACM Mardi Gras conference: From lightweight mash-ups to lambda grids: Understanding the spectrum of distributed computing requirements, applications, tools, infrastructures, interoperability, and the incremental adoption of key capabilities
Publisher: ACM
Full text available: PDF (349.62 KB)
Bibliometrics: Downloads (6 Weeks): 2, Downloads (12 Months): 29, Downloads (Overall): 281, Citation Count: 1

A home network is a typical ubiquitous computing network that consists of various consumer devices and servic environments. Home networks are requiring increasingly more complicated services, such as multimedia home theater and the monitoring and controlling ...
Keywords: distributed network, home network, multimedia room bridge adapter, seamless interoperability

17 CompUTE: a runtime infrastructure for device composition
Jakob E. Bardram, Christina Fuglsang, Simon C. Pedersen
May 2010 **AVI '10:** Proceedings of the International Conference on Advanced Visual Interfaces
Publisher: ACM Request Permissions
Full text available: Pdf (382.65 KB)
Bibliometrics: Downloads (6 Weeks): 7, Downloads (12 Months): 63, Downloads (Overall): 63, Citation Count: 0

In this paper, we present the idea of a composite device, which is one device made up of a composition of seve separate devices working together in concert. We present the CompUTE architecture as a runtime infrastructure device composition. ...
Keywords: application view re-direction, co-located cooperation, composite device, control re-direction, distrib clipboard

18 Authorised domain management using location based services
Imad Abbadi
September 2007 **Mobility '07:** Proceedings of the 4th international conference on mobile technology, applications, systems and the 1st international symposium on Computer human interaction in mobile technolog
Publisher: ACM
Full text available: PDF (406.94 KB)
Bibliometrics: Downloads (6 Weeks): 7, Downloads (12 Months): 19, Downloads (Overall): 132, Citation Count: 4

This paper focuses on creating a secure domain consisting of all devices owned by a single owner. This domain allows secure content sharing between devices in each domain and prevents the illegal copying of content to devices outside the domain. This ...
Keywords: DRM, access control, authorised domain, copyright protection, location based services, trusted computing

19 honeyM: a framework for implementing virtual honeyclients for mobile devices
T. J. O'Connor, Ben Sangster
March 2010 **WiSec '10:** Proceedings of the third ACM conference on Wireless network security
Publisher: ACM Request Permissions
Full text available: Pdf (664.81 KB)
Bibliometrics: Downloads (6 Weeks): 31, Downloads (12 Months): 241, Downloads (Overall): 351, Citation Count:

This paper presents honeyM, a framework for deploying virtual mobile device honeyclients. Honeyclients provid ability discover early warnings about novel attacks and ex- ploitations and are typically deployed to protect wire infrastructure. In ...

Keywords: bluetooth, gps, information assurance, mobile device security, wifi, wireless security

20 Configuring and managing a large-scale monitoring network: solving real world challenges for ultra-low-powered and long-range wireless mesh networks
Christophe Duqas
July 2005 **International Journal of Network Management**, Volume 15 Issue 4
Publisher: John Wiley & Sons, Inc.
Full text available: Pdf (287.88 KB)
Bibliometrics: Downloads (6 Weeks): 1, Downloads (12 Months): 4, Downloads (Overall): 499, Citation Count: 1

In creating wireless networking solutions suitable for deployment in harsh, unpredictable and widespread environments, we were confronted with a series of problems as yet unsolved by commercially available technologies. The purpose of this article is ...

Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2011 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-4351982	09/28/1982	Miller et al.	
		US-4658093	04/14/1987	Hellman	
		US-4704610	11/03/1987	Smith et al.	
		US-4796220	01/03/1989	Wolfe	
		US-5210795	05/11/1993	Lipner et al.	
		US-5291598	03/01/1994	Grundy	
		US-5414269	05/09/1995	Takahashi	
		US-5418854	05/23/1995	Kaufman et al.	
		US-5440635	08/08/1995	Bellovin et al.	
		US-5490216	02/06/1996	Richardson, III	
		US-5666415	09/09/1997	Kaufman	
		US-5745879	04/28/1998	Wyman	
		US-5754763	05/19/1998	Bereiter	
		US-5790664	08/04/1998	Coley et al.	
		US-5925127	07/20/1999	Ahmad	
		US-5974150	10/26/1999	Kaish et al.	
		US-6009401	12/28/1999	Horstmann	
		US-6044471	03/28/2000	Colvin	
		US-6158005	12/05/2000	Bharathan et al.	
		US-6230199	05/08/2001	Revashetti et al.	
		US-6233567	05/15/2001	Cohen	
		US-6243468	06/05/2001	Pearce et al.	
		US-6294793	09/25/2001	Brunfeld et al.	
		US-6330670	12/11/2001	England et al.	

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 2 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-6449645	09/10/2002	Nash	
		US-6536005	03/18/2003	Augarten	
		US-6785825	08/31/2004	Colvin	
		US-6859793	02/22/2005	Lambiase	
		US-6920567	07/19/2005	Doherty et al.	
		US-6976009	12/13/2005	Tadayon et al.	
		US-7032110	04/18/2006	Su et al.	
		US-7069440	06/27/2006	Aull	
		US-7069595	06/27/2006	Cognigni et al.	
		US-7085741	08/01/2006	Lao et al.	
		US-7188241	03/06/2007	Cronce et al.	
		US-7203966	04/10/2007	Abburi et al.	
		US-7206765	04/17/2007	Gilliam et al.	
		US-7272728	09/18/2007	Pierson et al.	
		US-7319987	01/15/2008	Hoffman et al.	
		US-7337147	02/26/2008	Chen et al.	
		US-7343297	03/11/2008	Bergler et al.	
		US-7327280	02/05/2008	Bachelder et al.	
		US-7463945	12/09/2008	Kiesel et al.	
		US-7653899	01/26/2010	Lindahi et al.	
		US-20010034712	10/25/2001	Colvin	
		US-20010044782	11/22/2001	Hughes et al.	
		US-20020019814	2/14/2002	Ganesan	
		US-20020082997	6/27/2002	Kobata et al.	

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 3 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-20020161718	10/31/2002	Coley et al.	
		US-20030065918	04/03/2003	Willey	
		US-20030172035	09/11/2003	Cronce et al.	
		US-20040024860	02/05/2004	Sato et al.	
		US-20040030912	02/12/2004	Merkle et al.	
		US-20040059929	03/25/2004	Rodgers et al.	
		US-20040143746	07/22/2004	Ligeti et al.	
		US-20040187018	09/23/2004	Owen et al.	
		US-20050108173	05/19/2005	Stefik et al.	
		US-20050138155	06/23/2005	Lewis	
		US-20050172280	08/04/2005	Ziegler et al.	
		US-20060072444	040/6/2006	Engel et al.	
		US-20060095454	05/04/2006	Shankar et al.	
		US-20060265337	11/23/2006	Wesinger, Jr.	
		US-20060161914	07/20/2006	Morrison et al.	
		US-20060282511	12/14/2006	Takano et al.	
		US-20070168288	07/19/2007	Bozeman	
		US-20070198422	08/23/2007	Prahlad et al.	
		US-20070203846	08/30/2007	Kavuri et al.	
		US-20070219917	09/20/2007	Liu et al.	
		US-20070282615	12/06/2007	Hamilton et al.	
		US-20080065552	03/13/2008	Elazar et al.	
		US-20080086423	04/10/2008	Waites	
		US-20080147556	06/19/2008	Smith et al.	

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 4 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-20080228578	09/18/2008	Mashinsky	
		US-20080320607	12/25/2008	Richardson	
		US-20090083730	03/26/2009	Richardson	
		US- 20090138975	05/28/2009	Richardson	

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document Country Code – Number – Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		WO 9220022	11/12/1992	Digital Equip. Corp.		
		WO 9301550	1/21/1993	Infologic Software		
		WO 9535533	12/28/1995	Megalode Corp.		
		AU 678985	6/19/1997	Uniloc Corp. Pty Ltd		
		WO 0067095	11/9/2000	Trymedia Systems		
		WO 2005104686	11/10/2005	IPASS Inc.		
		EP 1637961	3/22/2006	Microsoft Corp.		
		EP 1637958	3/22/2006	Microsoft Corp.		
		EP 1670188	6/14/2006	Alcatel		
		WO2007060516	5/31/2007	Lo		
		WO2008013504	1/31/2008	Starhub Ltd		
		WO2008157639	12/24/2008	Uniloc Corp.		
		WO2009039504	3/26/2009	Uniloc Corp.		
		WO2009065135	5/22/2009	Uniloc Corp.		
		WO2009076232	6/9/2009	Uniloc Corp.		
		WO2009105702	8/27/2009	Etchegoyen		
		WO2009143115	11/26/2009	Etchegoyen		

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 5 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document (Country Code – Number – Kind Code)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		2009158525	12/30/2009	Uniloc USA, Inc.		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date page(s), volume-issue number(s), publisher, city and/or country where published.	T
		WILLIAMS, R., "A Painless Guide to CRC Error Detection Algorithms", Ver. 3, Aug. 19, 1993.	
		ANGHA, F. et al., "Securing Transportation Network Infrastructure with Patented Technology of Device Locking – Developed By Uniloc USA", *avail. at:* http://www.dksassociates.com/admin/paperfile/ITS%20World%20Paper%20Submission_Uniloc%20_2_.pdf, Oct. 24, 2006.	
		ECONOLITE, "Econolite and Uniloc Partner to Bring Unmatched Infrastructure Security to Advanced Traffic Control Networks with Launch of Strongpoint", *avail. at: http://www.econolite.com/docs/press/20080304_Econolite_StrongPoint.pdf*, Mar. 4, 2008.	

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 8831

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
12/818,906	06/18/2010	717	2191	UN-NP-AD-037

APPLICANTS
Craig Stephen Etchegoyen, Irvine, CA; OK /QC/ 08/10/2011

**** CONTINUING DATA ****************************
This appln claims benefit of 61/220,092 06/24/2009 OK /QC/ 08/10/2011

**** FOREIGN APPLICATIONS *************************** None /QC/ 08/10/2011

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED **** ** SMALL ENTITY **
07/01/2010

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /QING CHEN/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		CA	6	20	2

ADDRESS

Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612
UNITED STATES

TITLE

Remote Update of Computers Based on Physical Device Recognition

FILING FEE RECEIVED		
462	FEES: Authority has been given in Paper No._____ to charge/credit DEPOSIT ACCOUNT No._____ for following:	☐ All Fees ☐ 1.16 Fees (Filing) ☐ 1.17 Fees (Processing Ext. of time) ☐ 1.18 Fees (Issue) ☐ Other _____ ☐ Credit

ACM DIGITAL LIBRARY Us Patent & Trademark Office

SIGN IN SIGN UP

remotely updating program configura

Searching for: remotely updating program configuration (start a new search)

Found **590** within *The ACM Guide to Computing Literature* (Bibliographic citations from major publishers in computing)

Limit your search to Publications from ACM and Affiliated Organizations (Full-Text collection: **312,030** items)

REFINE YOUR SEARCH

▾ Refine by Keywords
remotely updating prog
Discovered Terms

▾ Refine by People
Names
Institutions
Authors
Editors
Reviewers

▾ Refine by Publications
Publication Year
Publication Names
ACM Publications
All Publications
Content Formats
Publishers

▾ Refine by Conferences
Sponsors
Events
Proceeding Series

ADVANCED SEARCH
Advanced Search

FEEDBACK
Please provide us with feedback

Found **590** of **1,697,209**

Search Results Related Journals Related Magazines Related SIGs Related Conferences

Results 1 - 20 of 590 Sort by relevance in expanded for

Result page: **1** 2 3 4 5 6 7 8 9 10 next

1 Vigilante: End-to-end containment of Internet worm epidemics

Manuel Costa, Jon Crowcroft, Miguel Castro, Antony Rowstron, Lidong Zhou, Lintao Zhang, Paul Barham

December 2008 **Transactions on Computer Systems (TOCS)** . Volume 26 Issue 4

Publisher: ACM Request Permissions

Full text available: Pdf (1.99 MB)

Bibliometrics: Downloads (6 Weeks): 20, Downloads (12 Months): 163, Downloads (Overall): 704, Citation Count:

Worm containment must be automatic because worms can spread too fast for humans to respond. Recent work proposed network-level techniques to automate worm containment; these techniques have limitations because there is no information about the vulnerabilities ...

Keywords: Worm containment, dynamic data-flow analysis, program analysis, self-certifying alerts, vulnerabil condition slicing

2 Communications of the ACM: Volume 54 Issue 5

May 2011 Communications of the ACM

Publisher: ACM

Full text available: Digital Edition , PDF (7.71 MB)

Bibliometrics: Downloads (6 Weeks): 388, Downloads (12 Months): 388, Downloads (Overall): 388, Citation Count

3 Fast detection of communication patterns in distributed executions

Thomas Kunz, Michiel F. H. Seuren

November 1997 **CASCON '97:** Proceedings of the 1997 conference of the Centre for Advanced Studies on Collabora research

Publisher: IBM Press

Full text available: Pdf (4.21 MB)

Bibliometrics: Downloads (6 Weeks): 17, Downloads (12 Months): 177, Downloads (Overall): 6087, Citation Count

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time diagr are often used to obtain a better understanding of the execution of the application. The visualization tool we us Poet, an event ...

4 Web customization using behavior-based remote executing agents

Eugene Hung, Joseph Pasquale

May 2004 **WWW '04:** Proceedings of the 13th international conference on World Wide Web

Publisher: ACM

Full text available: Pdf (128.60 KB)

Bibliometrics: Downloads (6 Weeks): 1, Downloads (12 Months): 12, Downloads (Overall): 551, Citation Count: 1

ReAgents are remotely executing agents that customize Web browsing for non-standard clients. A reAgent is essentially a one-shot" mobile agent that acts as an extension of a client dynamically launched by the client to on its behalf at a remote more ...

Keywords: dynamic deployment, remote agents, web customization

5 Two executable mobility design patterns: mfold and mmap

Zara Field, Rick Dewar, Phil Trinder, Andre Rauber Du Bois

October 2006 **PLoP '06:** Proceedings of the 2006 conference on Pattern languages of programs

Publisher: ACM

Full text available: PDF (428.21 KB)

Bibliometrics: Downloads (6 Weeks): 3, Downloads (12 Months): 8, Downloads (Overall): 56, Citation Count: 0

We present two mobility design patterns, mfold and mmap. The patterns are equipped with corresponding coordination specifications. mobility skeletons. implemented on top of a host object-orientated mobile code language. Voyager. The mobility skeletons ...
Keywords: agents. code mobility, design patterns

6 Secure automotive on-board protocols: a case of over-the-air firmware updates
Muhammad Sabir Idrees, Hendrik Schweppe, Yves Roudier, Marko Wolf, Dirk Scheuermann, Olaf Henniger
March 2011 **Nets4Cars/ Nets4Trains'11:** Proceedings of the Third international conference on Communication technologies for vehicles
Publisher: Springer-Verlag
Full text available: Publisher Site
Bibliometrics: Downloads (6 Weeks): n/a. Downloads (12 Months): n/a. Downloads (Overall): n/a. Citation Count:

The software running on electronic devices is regularly updated, these days. A vehicle consists of many such devices, but is operated in a completely different manner than consumer devices. Update operations are safety critical in the automotive domain. ...
Keywords: over the air firmware updates, security architectures, security protocols, software functionality

7 Proceedings of the 14th international conference on Principles of distributed systems
Chenyang Lu, Toshimitsu Masuzawa, Mohamed Mosbah
December 2010 **OPODIS'10:** Proceedings of the 14th international conference on Principles of distribut systems
Publisher: Springer-Verlag
Bibliometrics: Downloads (6 Weeks): n/a. Downloads (12 Months): n/a. Downloads (Overall): n/a. Citation Count:

8 Charles W. Bachman interview: September 25-26, 2004, Tucson, Arizona
Thomas Haigh, Charles W. Bachman
January 2006 Oral History interviews
Publisher: ACM
Full text available: Pdf (974.87 KB)
Bibliometrics: Downloads (6 Weeks): 4. Downloads (12 Months): 290. Downloads (Overall): 2114. Citation Count:

Charles W. Bachman reviews his career. Born during 1924 in Kansas, Bachman attended high school in East Lansing, Michigan before joining the Army Anti Aircraft Artillery Corp, with which he spent two years in the Southwest Pacific Theater, during ...

9 Nixos: A purely functional linux distribution
Eelco Dolstra, Andres LÖh, Nicolas Pierron
November 2010 **Journal of Functional Programming** . Volume 20 Issue 5-6
Publisher: Cambridge University Press
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a. Downloads (Overall): n/a, Citation Count:

Existing package and system configuration management tools suffer from an imperative model, where system administration actions such as package upgrades or changes to system configuration files are stateful: they destructively update the state ...

10 Communications of the ACM: Volume 53 Issue 12
December 2010 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition . PDF (6.57 MB)
Bibliometrics: Downloads (6 Weeks): 82. Downloads (12 Months): 82. Downloads (Overall): 82. Citation Count: 0

11 Communications of the ACM: Volume 54 Issue 3
March 2011 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition . PDF (11.32 MB)
Bibliometrics: Downloads (6 Weeks): 190. Downloads (12 Months): 190. Downloads (Overall): 190. Citation Count

12 Communications of the ACM: Volume 53 Issue 6
June 2010 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition . Pdf
Bibliometrics: Downloads (6 Weeks): 44. Downloads (12 Months): 44. Downloads (Overall): 44. Citation Count: 0

13 Communications of the ACM: Volume 52 Issue 12
December 2009 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition , Pdf (7.37 MB)
Bibliometrics: Downloads (6 Weeks): 27, Downloads (12 Months): 27, Downloads (Overall): 27, Citation Count: 0

14 Communications of the ACM: Volume 51 Issue 12
December 2008 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition , Pdf (6.91 MB)
Bibliometrics: Downloads (6 Weeks): 20, Downloads (12 Months): 20, Downloads (Overall): 3752, Citation Count:

15 DART: directed automated random testing
Patrice Godefroid, Nils Klarlund, Koushik Sen
June 2005 **PLDI '05:** Proceedings of the 2005 ACM SIGPLAN conference on Programming language design and implementation
Publisher: ACM Request Permissions
Full text available: Pdf (163.84 KB)
Bibliometrics: Downloads (6 Weeks): 58, Downloads (12 Months): 503, Downloads (Overall): 2502, Citation Count

We present a new tool, named DART, for automatically testing software that combines three main techniques: *automated* extraction of the interface of a program with its external environment using static source-code parsi (2) automatic generation ...

Keywords: automated test generation, interfaces, program verification, random testing, software testing

Also published in:
June 2005 **SIGPLAN Notices** Volume 40 Issue 6

16 Performance evaluation of the Orca shared-object system
Henri E. Bal, Raoul Bhoedjang, Rutger Hofman, Ceriel Jacobs, Koen Langendoen, Tim Rühl, M. Frans Kaashoek
February 1998 **Transactions on Computer Systems (TOCS)** , Volume 16 Issue 1
Publisher: ACM Request Permissions
Full text available: Pdf (179.39 KB)
Bibliometrics: Downloads (6 Weeks): 1, Downloads (12 Months): 38, Downloads (Overall): 458, Citation Count: 37

Orca is a portable, object-based distributed shared memory (DSM) system. This article studies and evaluates th design choices made in the Orca system and compares Orca with other DSMs. The article gives a quantitative analysis of Orca's coherence protocol ...

Keywords: distributed shared memory, parallel processing, portability

17 Communications of the ACM: Volume 52 Issue 5
May 2009 Communications of the ACM
Publisher: ACM
Full text available: Digital Edition , Pdf (7.31 MB)
Bibliometrics: Downloads (6 Weeks): 22, Downloads (12 Months): 22, Downloads (Overall): 1559, Citation Count:

18 Why Order Matters: Turing Equivalence in Automated Systems Administration
November 2002 **LISA '02:** Proceedings of the 16th USENIX conference on System administration
Publisher: USENIX Association
Bibliometrics: Downloads (6 Weeks): n/a, Downloads (12 Months): n/a, Downloads (Overall): n/a, Citation Count:

Hosts in a well-architected enterprise infrastructure are self-administered: they perform their own maintenance upgrades. By definition, self-administered hosts execute self-modifying code. They do not behave according to simple state machine rules, ...

19 Frontmatter (TOC, Letters, Philosophy of computer science, Interviewers needed, Taking software requirements creation from folklore to analysis, SW components and product lines: from business to syst and technology, Software engineering survey)
September 2005 **SIGSOFT Software Engineering Notes** , Volume 30 Issue 5
Publisher: ACM
Full text available: Pdf (1.98 MB)

Bibliometrics: Downloads (6 Weeks): 9. Downloads (12 Months): 113. Downloads (Overall): 3628. Citation Count:

20 NixOS: a purely functional Linux distribution
Eelco Dolstra, Andres Löh
September 2008 **ICFP '08**: Proceeding of the 13th ACM SIGPLAN international conference on Functional programm
Publisher: ACM Request Permissions
Full text available: Fly (27:0 MIN), Mp3 (27:0 MIN), Pdf (403.95 KB)
Bibliometrics: Downloads (6 Weeks): 6. Downloads (12 Months): 34. Downloads (Overall): 231. Citation Count: 6

Existing package and system configuration management tools suffer from an imperative model, where system administration actions such as upgrading packages or changes to system configuration files are stateful: they destructively update the state ...
Keywords: NixOS, nix, package management, purely functional deployment model, purely functional language software deployment, system configuration management

Also published in:
September 2008 **SIGPLAN Notices** Volume 43 Issue 9

Result page: **1** 2 3 4 5 6 7 8 9 10 next

The ACM Digital Library is published by the Association for Computing Machinery. Copyright © 2011 ACM, Inc.
Terms of Usage Privacy Policy Code of Ethics Contact Us

Useful downloads: Adobe Acrobat QuickTime Windows Media Player Real Player

Doc Code: PEER.SB423
Document Description: Third Party Submission – Peer Review

PTO/SB/423 (03-11)

THIRD-PARTY SUBMISSION OF PATENTS OR PUBLICATIONS UNDER PEER REVIEW PILOT PROGRAM

Complete If Known	
Application Number	12/818,906
Filing Date	06-18-2010
First Named Inventor	Craig Stephen Etchegoyen
Art Unit	2192
Examiner Name	-
Attorney Docket Number	UN-NP-AD-037

DO NOT SUBMIT THIS FORM VIA EFS-WEB

A signed certification and request to participate in the Peer Review Pilot Program (Form PTO/SB/422) was filed on (insert date): 12-28-2010

NOTE: A submission received by the Office prior to the filing of a signed certification and request to participate in the Peer Review Pilot Program (Form PTO/SB/422) will be returned to the Third-Party Submitter and will NOT be placed into the Application File.

U.S. PATENT DOCUMENTS

Cite No.	Check Box If Comments Attached	Document Number Number-Kind Code[1] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Examiner Initials*
1	☐	US- 6327617	12-04-2001	Philip Fawcett	
2	☐	US- 6880086	04-12-2005	Joseph Kidder	
3	☐	US-			
4	☐	US-			
5	☐	US-			
6	☐	US-			

FOREIGN PATENT DOCUMENTS

Cite No.	Check Box If Comments Attached	Document Number Country Code[2]-Number[3]-Kind Code[4] (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Examiner Initials*
1	☐				
2	☐				
3	☐				
4	☐				
5	☐				
6	☐				

[1] See Kind Codes of USPTO patent documents at www.uspto.gov/patents/abc/kindcodesum.jsp or MPEP 901.04(a).
[2] Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3).
[3] For Japanese patent documents, the indication of the year of the Emperor must precede the serial number of the patent document.
[4] Kind of document by the appropriate symbols, as indicated on the document under WIPO Standard ST.16, if possible.
*EXAMINER: Initial each citation that is considered. Draw a line through each citation that is not considered. Include a copy of this form with next communication to applicant.

NON PATENT LITERATURE DOCUMENTS			
Cite No.	Check Box If Comments Attached	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.) date, page(s), volume-issue number(s), publisher, city and/or country where published.	Examiner Initials*
1	☐		
2	☐		
3	☐		
4	☐		
5	☐		
6	☐		

My signature below certifies that I am authorized to make the enclosed submission as part of the Peer Review Pilot Program.

Signature of Third-Party Submitter	s/		Date 4-11-2011
Name (Printed/Typed)	Andrea Casillas	Reg. No., if applicable	Telephone Number 212-431-2368
Address	185 W. Broadway New York, NY 10013		

Examiner Signature		Date Considered	8/12/2011

*EXAMINER: Initial each citation that is considered. Draw a line through each citation that is not considered. Include a copy of this form with next communication to applicant.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. no.:	12/818,906	Conf. no.	8831
Applicant:	Craig S. Etchegoyen	Art Unit:	2191
Filed:	June 18, 2010	Examiner:	Qing Chen

Title: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

RESPONSE TO OFFICE ACTION

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir,

In response to the Office Action mailed August 12, 2011, please amend the present application as follows:

Amendments to the Specification begin on page 2.

Amendments to the Claims are shown in the listing of claims beginning on page 3.

Remarks begin on page 9.

IN THE SPECIFICATION:

Please add a heading for the "Summary" section between paragraphs [0003] and [0004] of the specification, as follows:

SUMMARY OF THE INVENTION

Please replace the paragraph beginning on p. 7 at line 10 with the following amended paragraph:

[0029] In yet further related aspects, the memory module 508 may optionally include executable code for the processor module ~~504~~ 506 configured to: (a) collect unique identifiers from at least one of an audit server and client device; (b) analyze the collected unique identifiers; (c) determine an updated program configuration for the client device; and (d) deliver the updated program configuration to the client device(s). One or more of steps (a)-(d) may be performed by a processor module in lieu of or in conjunction with the means described above.

IN THE CLAIMS:

1. (currently amended) A system for remotely updating a program configuration, comprising a client device and an update server wherein:

(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device , the client device comprising:

a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of the software that the user of the client device is entitled to use, ~~and~~ (ii) generates a unique device identifier[[s]] for the client device, the unique device identifier[[s]] based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the software identifier being unique to a particular copy of the software and to the particular user of the software; and

a first transceiver configured to send the unique device and software identifiers to ~~at least one~~ an update server via the Internet; and

(b) the ~~an~~ update server is configured to ~~collect~~ receive the unique device and software identifiers from ~~at least one~~ the client device, the update server comprising:

a second processor coupled to memory and configured to analyze the unique device and software identifiers at the update server, and to determine based on the analyzed unique device and software identifiers an updated program configuration if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier; and

a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

2. (original) The system of claim 1 wherein the unique device identifier comprises a hash code.

3. (original) The system of claim 1 wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.

4. (original) The system of claim 3 wherein the at least one irreversible transformation comprises a cryptographic hash function.

5. (currently amended) The system of claim 1 wherein the unique device identifier further comprises ~~software identifiers and~~ one or more geo-location ~~identifiers~~ codes.

6. (currently amended) The system of claim 5 wherein at least one of the geo-location ~~identifiers~~ codes comprises an Internet Protocol address of the client device.

7. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

8. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.

9. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model.

10. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.

11. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.

12. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.13.

13. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.

14. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type.

15. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.

16. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.

17. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.

18-20. (canceled)

21. (new) A client device configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a processor;

APPLE EXHIBIT 1102, Page 869 of 1048

memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for software on the client device, the software identifier being unique to a particular copy of the software and to the particular user of the software; and

a transceiver configured to (i) send the unique device and software identifiers to an update server via the Internet, and (ii) receive from the update server an updated program configuration if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier.

22. (new) An update server configured to execute a computer program to receive a unique device identifier and a unique software identifier from a client device, the update server comprising:

a processor;

memory coupled to the processor and storing the computer program which, when executed by the processor, analyzes the unique device and software identifiers at the update server, and determines based on the analyzed unique device and software identifiers an updated program configuration if a user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier; and

a transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

REMARKS

Applicant thanks Examiner Chen for a thorough review of the application papers and his opinion on patentability.

Claims 1-17 and 21-22 are pending in the application. Claims 21 and 22 are new. Applicant respectfully requests reconsideration of all pending claims.

Response to Claim Objections

Dependent claims 2, 3, and 5 were objected to for informalities. Applicant has amended claim 1 to recite "device identifier" in singular form. This obviates the objections to claims 2 and 3. Applicant has amended claim 5 according to Examiner Qing's suggestion.

Response to Rejections Under 35 USC §102

Claims 18-20 were rejected under 35 USC §102(b) as being anticipated by U.S. Patent 6,467,088 ("*AlSafadi*"). Applicant has canceled these claims.

Response to Rejections Under 35 USC §103

Claims 1-17 were rejected under 35 USC §103 as being unpatentable over *AlSafadi* in view of U.S. Patent 6,880,086 ("*Kidder*"). Applicant respectfully traverses.

The present invention ("*Etchegoyen*") and the *AlSafadi* and *Kidder* references all disclose inventions in the general field of remote updating of software. Upon closer inspection, however, there are patentable differences between *Etchegoyen* and the references of record. Generally, *Etchegoyen* is distinctive because it generates, from multiple machine parameters, a unique device identifier for a client device to determine, among other things, whether the client device is *licensed* to receive a software upgrade.

AlSafadi generally teaches a reconfiguration manager (or upgrade server) that receives an upgrade request from a client device. The upgrade request contains (i) identification of a desired software upgrade, and (ii) a list of hardware and software components presently installed on the client device. The reconfiguration server compares the list of components to known "good" configurations that will support the requested upgrade. If the list of components satisfies a good configuration, then the upgraded version of the software is downloaded to the client. If the list of

components does not satisfy a good configuration, then the reconfiguration manager either denies the request, or informs the client that the requested upgrade is unknown, or gives the client an option to download all components needed to implement the desired upgrade. *AlSafadi* at 4:12 to 5:40.

Importantly, *AlSafadi* doesn't teach generating from the machine parameters of the client device, a unique device identifier for the client device. In *AlSafadi*, the client computer's software and hardware component identifiers are predetermined, and are either included with the upgrade request, or they are obtained by cross-referencing a serial number of the client device to configuration data stored in a database. *Id.* at 4:40-47. There is no test in *AlSafadi* to determine whether the client is licensed to receive the upgrade.

Kidder teaches the use of software "signatures" to promote hot upgrades of software components, whereby an early release of a software product can be upgraded to a later release. *Kidder's* method involves generating a signature for each component of a first release of a software product, and when an upgrade request is received from a client device, comparing each signature to the signatures in a second release of the product. During operation of the product, each signature that matches continues to run and is not upgraded, and each signature that doesn't match is ignored in favor of its corresponding upgraded component. *Kidder* at 3:49-65.

Importantly, *Kidder's* method does not generate a composite device identifier for the client device. *Kidder* generates individual signatures for each component in the client device that it surveys, to facilitate a component-by-component comparison. There is no test in *Kidder* to determine whether the client is licensed to receive the upgrade.

Etchegoyen, on the other hand, teaches a security-related upgrading method. The *Etchegoyen* invention collects machine parameters and other identifiers from the client device to determine its identity and thus whether it is licensed to receive a software upgrade. Neither *AlSafadi* nor *Kidder* is concerned with collecting machine parameters for the purpose of uniquely identifying the client device, to determine whether a software upgrade for that particular client device is authorized.

To make these distinctions more apparent in the claims, applicant files amendments herewith.

Claim 1

Claim 1 has been amended herein, and now recites:

> *a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters <u>including account information for a user of the client device and features of the software that the user of the client device is entitled to use</u>, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) <u>collects a unique software identifier for the software on the client device, the software identifier being unique to a particular copy of the software and to the particular user of the software</u>; and ...*
>
> *a second processor coupled to memory and configured to analyze the unique device <u>and software</u> identifiers at the update server, and to determine based on the analyzed unique device <u>and software</u> identifiers an updated program configuration <u>if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier</u> ...*

The Office Action on page 6 admits that *AlSafadi* doesn't teach the first processor element as claimed, but cites to *Kidder* as teaching physical device recognition, and generating a unique device identifier based on the functionality of *Kidder's* Master Control Driver (MCD) 38. Applicant respectfully disagrees.

In response to an upgrade request, when *Kidder* surveys a client device, MCD 38 assigns a physical identification number (PID) to each internal component of the client device that is being surveyed, so that each <u>*individual*</u> component can be identified. Using the PIDs, *Kidder* takes an "inventory" of the client device so that MCD 38 can determine its configuration and thereby determine which cards need an upgraded driver. *Kidder* at 61:9-41; 62:14-19. There is no teaching to generate, from surveyed machine parameters, a composite device identifier that uniquely identifies the client device. Applicant submits that claim 1 should be allowed for this reason alone.

In addition, there is no teaching in either *AlSafadi* or *Kidder* to determine "an updated program configuration if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier". Applicant submits that claim 1 should be allowed for this reason alone.

For the reasons presented above, applicant submits that claim 1, as amended, is patentable over the cited references. Specifically, neither *AlSafadi* nor *Kidder* teaches (i) collecting a unique software identifier from a client device, (ii) generating from the machine parameters of the client device, a unique device identifier for the client device, and (iii) determining an updated program configuration for the client device if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier.

Claims 2 - 17

Claims 2 - 17 each depend from claim 1. Applicant reasserts here the foregoing arguments in favor of claim 1, and requests that claims 2 - 17 be allowed based on dependency.

Claim 2, in particular, should be allowed over *AlSafadi* in view of *Kidder* for reciting "wherein the unique identifier comprises a hash code". In rejecting claim 2, the Office Action cites to *Kidder's* use of the Sha-1 utility at 88:62-64. But the hashed signature as taught in *Kidder* comprises a SHA-1 hash for each of the software components present in the device requesting an upgrade. Nowhere does *Kidder* teach or suggest hashing a unique identifier that is generated from multiple such components.

Claim 3 should be allowed over *AlSafadi* in view of *Kidder* for similar reasons. The Office Action interprets the Sha-1 utility as used in *Kidder* as anticipating the "irreversible transformation" feature recited in claim 3. But again, there is no teaching in *Kidder* to derive a unique device identifier from multiple machine parameters. Therefore, the irreversible transformation of Sha-1 as proposed in *Kidder* is not implemented on a "device identifier" as that term is defined in the present specification.

Support for Claim Amendments and New Claims

Support for the amendments to claim 1 may be found in the original specification, *e.g.* U.S. Application Pub. 2010/0333081 as follows:

- pars. 0006, 0027 ("performing a remote update of a program configuration on the client device");
- par. 0034 ("including account information for a user of the client device and features of software that the user of the client device is entitled to use");

- pars. 0019, etc. (generating a unique identifier based on machine parameters);
- pars. 0058-0069 ("collects a unique software identifier for the software on the client device, the software identifier being unique to a particular copy of the software and to the particular user of the software");
- pars. 0062-0063 ("sending the unique device identifier and software identifiers to an update server via the Internet"); and
- pars. 0019, 0027, 0036, 0058-0060; FIG. 1 it. 105, 107, 109; FIG. 2 it. 250 ("if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier").

New claim 21 recites similar limitations as in claim 1, but is written from the perspective of a client device that executes software comprising a method according to the invention. Applicant submits that claim 21 is allowable over the references of record for the same reasons presented above in support of claim 1.

Support for claim 21 may be found in the original specification, *e.g.* U.S. Application Pub. 2010/0333081 par. 0007.

New claim 22 recites similar limitations as in claim 1, but is written from the perspective of an update server that executes software comprising a method according to the invention. Applicant submits that claim 22 is allowable over the references of record for the same reasons presented above in support of claim 1.

Support for claim 22 may be found in the original specification, *e.g.* U.S. Application Pub. 2010/0333081 par. 0010.

/

/

/

/

/

/

Conclusion

In view of all of the above, applicant believes that all pending claims are in condition for allowance and earnestly requests that these claims be passed to issuance. If the Examiner believes that a telephone conversation would help to expedite prosecution, please call the undersigned attorney at the number below.

Respectfully Submitted,

Sean D. Burdick
Reg. No. 51,513

Uniloc USA, Inc.
2151 Michelson Drive, Suite 100
Irvine, CA 92612
(949) 825-5527

Electronic Patent Application Fee Transmittal

Application Number:	12818906
Filing Date:	18-Jun-2010
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Filer:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Submission- Information Disclosure Stmt	1806	1	180	180
		Total in USD ($)		**180**

Electronic Acknowledgement Receipt

EFS ID:	11387806
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick
Filer Authorized By:	
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	14-NOV-2011
Filing Date:	18-JUN-2010
Time Stamp:	20:23:47
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$180
RAM confirmation Number	7706
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Transmittal Letter	AT-037_IDS_Transmittal.pdf	30029 48bbb0da2d95130799a0b5d5153c4fb563fd7aaf	no	2
Warnings:					
Information:					
2	Information Disclosure Statement (IDS) Form (SB08)	AD-037_IDS_list.pdf	30536 32973ef585c5140c45f708807ec8f70e1446d750	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Foreign Reference	EP1096406_Madonion_OY.pdf	115522 3c4cad551ace9f10554c51baff152d3258f29892	no	4
Warnings:					
Information:					
4	Foreign Reference	WO01090892_Everdream.pdf	5596143 13a4fee08f1f7f520fffaf95c84dbcf97435451a	no	37
Warnings:					
Information:					
5	Non Patent Literature	XP002439673.pdf	2318458 a8973a678fc161a489b9ae38580ee31c6ce0f3f7	no	69
Warnings:					
Information:					
6	Non Patent Literature	RFC_1321_XP002337165.pdf	614309 a27595ae7817528a7902561d0a8068b7b937998b	no	21
Warnings:					
Information:					
7	Non Patent Literature	XP002604710.pdf	112965 b25cd933c2b3161b7cf0624769e31b02e0194a83	no	2
Warnings:					
Information:					
8	Non Patent Literature	XP002603488.pdf	324904 ac39dbd53e422150fe261bd5ba732442910f3eed	no	3
Warnings:					
Information:					
9	Amendment/Req. Reconsideration-After Non-Final Reject	response_OA_08-12-2011.pdf	82811 67aa6983a06e677f8f53321d126ccd097be635dc	no	14

Warnings:					
Information:					
10	Fee Worksheet (SB06)	fee-info.pdf	30067 08cd2a96f138baf7bbfc3cd6bb3660523 1d3 5211	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	9255744	

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National 5tage of an International Application under 35 U.5.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the U5PTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. no.:	12/818,906	Conf. no.	8831
Applicant:	Craig S. Etchegoyen	Art Unit:	2191
Filed:	June 18, 2010	Examiner:	Qing Chen

Title: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

Applicant hereby submits, without admission of prior art effect thereof, form(s) PTO/SB/08 pursuant to the duty of disclosure requirements of 37 CFR §§ I.56, 1.97 and I.98.

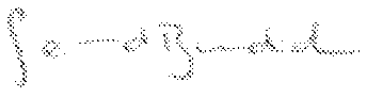
Applicant has listed publication dates on the attached form(s) PTO/SB/08 based on information presently available to the undersigned. However, the listed publication dates should not be construed as an admission that the information was actually published on the date indicated.

It is respectfully requested that the Examiner initial and return a copy of the enclosed forms PTO/SB/08, and to indicate in the official file wrapper of this patent application that the documents have been considered.

Applicant submits concurrently herewith the fee set forth in § 1.17(p).

APPLE EXHIBIT 1102, Page 883 of 1048

Respectfully Submitted,



Sean D. Burdick
Reg. No. 51,513

Uniloc USA, Inc.
2151 Michelson Drive, Suite 100
Irvine, CA 92612
(949) 825-5527

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2191
Examiner Name	Qing Chen
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-6,324,519	11/27/2001	Eldering, Charles A.	
		US-2005/0055269	03/10/2005	Roetter et al.	
		US-2007/0072676	03/29/2007	Baluja, Shumeet	
		US-2008/0167943	07/10/2008	O'Neil et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document Country Code – Number – Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		EP 1 096 406	5/2/2001	Madonion Oy		
		WO 2001/090892	11/29/2001	Everdream Inc.		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date page(s), volume-issue number(s), publisher, city and/or country where published.	T
		Microsoft Corporation, "Operations Guide: Microsoft Systems Management Server 2003," 2003, Internet Citation retrieved on June 27, 2007. XP 002439673	
		Rivest, R. "RFC 1321 - The MD5 Message Digest Algorithm," April 1992, Retrieved from the Internet on July 21, 2005.	
		Wikipedia: "Software Extension," May 28, 2009, Internet Article retrieved on October 11, 2010. XP002604710	
		H. Williams, et al., "Web Database Applications with PHP & MySQL", Chapter 1, "Database Applications and the Web", ISBN 0-596-00041-3, O'Reilly & Associates, Inc., March 2002, avail. at: http://docstore.mik.ua/orelly/webprog/webdb/ch01_01.htm. XP002603488	

Examiner Signature		Date Considered	

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 096 406 A3

(12) **EUROPEAN PATENT APPLICATION**

(88) Date of publication A3:
07.05.2003 Bulletin 2003/19

(43) Date of publication A2:
02.05.2001 Bulletin 2001/18

(21) Application number: **00309137.8**

(22) Date of filing: **17.10.2000**

(51) Int Cl.7: **G06F 17/60**, G06F 11/22, G06F 9/445

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **29.10.1999 GB 9925545**

(71) Applicant: **Madonion OY**
02200 Espoo (FI)

(72) Inventor: **Mäki, Marcus**
00230 Espoo (FI)

(74) Representative: **Lind, Robert**
Marks & Clerk,
Nash Court,
Oxford Business Park South
Oxford OX4 2RU (GB)

(54) **Computer upgrading**

(57) A method of providing computer upgrade information to a computer user and comprising examining the hardware and/or software components in a client computer 1, 4-7. The results of the examination are sent from the client computer 1, 4-7 to a central computer 10 over the Internet 2. On the basis of the received results, the central computer 10 determines one or more upgrades which would improve a performance criterion or criteria of the client computer 1, 4-7. An identification of the upgrade(s) is sent from the central computer 10 to the client computer 1, 4-7 over the Internet 2 together with an option to purchase the identified upgrade(s).

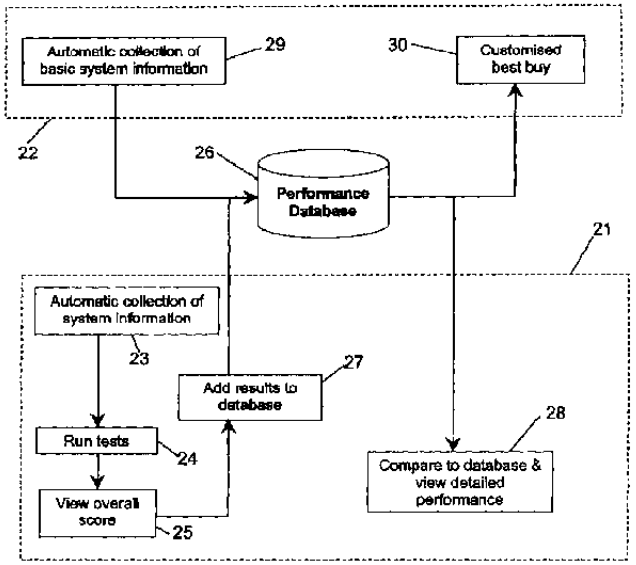


Figure 2

EP 1 096 406 A3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 30 9137

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	US 5 867 714 A (TERRELL MICHAEL R ET AL) 2 February 1999 (1999-02-02) * abstract * * figures 1,3 * * column 2, line 35-40 * * column 3 * * column 4, line 6-8,40-42 * * column 6-11 * * column 12, line 30-42,49-62 * * column 13, line 16-29 * * claims 1,2,7-9,15,16 * ---	1-13	G06F17/60 G06F11/22 G06F9/445
X	US 5 287 505 A (CALVERT NATHANIEL ET AL) 15 February 1994 (1994-02-15) * abstract * * figures 1,2 * * column 2, line 1-22 * * column 3, line 57-62 * * column 4, line 13-16,30-56 * * column 5, line 4-8,45-50 * * column 6, line 48-51 * * column 7, line 19-33 * * column 9, line 11-16 * * column 10, line 7-10,26-34 * * column 11, line 20-49 * * claims 1,3,5,6,8-10 * ---	1-13	TECHNICAL FIELDS SEARCHED (Int.Cl.7) G06F
A	US 5 845 077 A (FAWCETT PHILIP E) 1 December 1998 (1998-12-01) * abstract * * column 1, line 24-29 * * column 2, line 24-67 * * column 3, line 3-6,10-14 * * column 5, line 34-46 * * column 6, line 12-32 * * column 7, line 3-11,23-56 * * column 10, line 11-25 * * claims 1,3-8,10,12,14,17,22-24 * --- -/--	1-13	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	11 March 2003	Lutz, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

APPLE EXHIBIT 1102, Page 887 of 1048

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 00 30 9137

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropr ate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
A	US 5 553 235 A (ROSS JOSEPH C ET AL) 3 September 1996 (1996-09-03) * abstract * * claims 1,10-12 * -----	1-13	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	11 March 2003	Lutz, A

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

APPLE EXHIBIT 1102, Page 888 of 1048

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 00 30 9137

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

11-03-2003

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 5867714	A	02-02-1999	NONE		
US 5287505	A	15-02-1994	DE	68920462 D1	23-02-1995
			DE	68920462 T2	13-07-1995
			EP	0333620 A2	20-09-1989
			JP	1243135 A	27-09-1989
			JP	1916596 C	23-03-1995
			JP	6044242 B	08-06-1994
US 5845077	A	01-12-1998	US	6073214 A	06-06-2000
			US	6327617 B1	04-12-2001
			US	2002016956 A1	07-02-2002
US 5553235	A	03-09-1996	US	5684945 A	04-11-1997

EPO FORM P045

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

APPLE EXHIBIT 1102, Page 889 of 1048

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2001 (29.11.2001)

PCT

(10) International Publication Number
WO 01/90892 A1

(51) International Patent Classification[7]: G06F 9/445, 1/00

(21) International Application Number: PCT/US01/15720

(22) International Filing Date: 14 May 2001 (14.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/580,931 25 May 2000 (25.05.2000) US

(71) Applicant *(for all designated States except US)*: **EVERDREAM, INC.** [US/US]; 6591 Dumbarton Circle, Fremont, CA 94555 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **MCCALEB, Jed** [US/US]; 1663 Fulton, San Francisco, CA 94117 (US). **RIVE, Russell** [CA/US]; 2433 Greer Road, Palo Alto, CA 94303 (US).

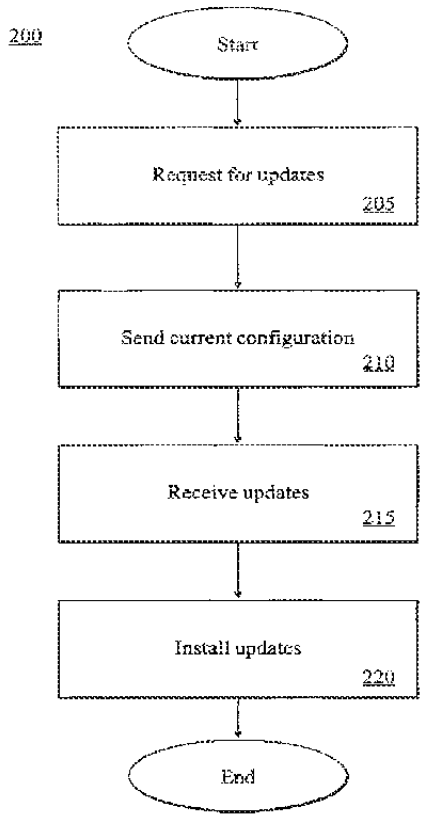
(74) Agents: **MALLIE, Michael, J.** et al.; Blakely, Sokoloff, Taylor & Zafman LLP, 7th Floor, 12400 Wilshire Boulevard, Los Angeles, CA 90025 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European

[Continued on next page]

(54) Title: INTELLIGENT PATCH CHECKER



(57) Abstract: A method for remotely updating software in a plurality of computer systems is disclosed. In one embodiment, a request for an upgrade is sent to a server system connected in a network. The upgrade is for a software application installed in a client connected in the network. The request is sent from the client system. The request comprises a unique identification associated with the client system. The unique identification is recognized by the server system as belonging to the client system. At least one instruction is received from the server system in response to the request for the upgrade. The at least one instruction directs the client system to collect application information about the software application installed on the client system. The application information about the software application is sent to the server system. The server system performs a comparison between the application information about the software application and the most-updated upgrade package for the software application. The most-updated upgrade package for the software application is received by the client system automatically when the comparison indicates that the most-updated upgrade package has not been installed on the client system.

WO 01/90892 A1

patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
with international search report

before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTELLIGENT PATCH CHECKER

FIELD OF THE INVENTION

The present invention relates generally to field of remote support for computer systems. More specifically, the present invention is directed to a method and an apparatus for updating software in a plurality of computer systems.

BACKGROUND

Personal computers have become an important part of the information age. The use of the personal computers has expanded beyond the traditional university campus and large office environments. Today, many small businesses and residences have at least one personal computer running a wide range of applications sold by many different software vendors.

As the applications become easier to use, the personal computers are no longer considered the tool for only the technical users. The user community has expanded and the personal computers are being viewed more as the tools to run the applications. Most users are interested in dealing with the applications and usually have no clue when something goes wrong with their personal computers. When the user is unable to use the application on the user's personal computer, the usual action is to take the personal computer to a local personal computer repair shop.

Since there are many different brands of personal computers such as, for example, IBM, Compaq, Gateway, Dell, etc., it is usually the case that each personal computer from a different brand may have a different set up. For example, the IBM personal computer may use a different video adapter from the Dell personal computer, among others. As such, to have a problem corrected, the user usually has to bring the personal computer into the repair shop so that the technician can isolate the problem.

APPLE EXHIBIT 1102, Page 892 of 1048

One of the most common problems of application failure is incompatibility. The incompatibility may be related to the hardware or to the other applications in the same personal computer system. For example, the user may have installed a new application that is incompatible with the existing application when running together. The user may have installed a new hardware adapter that is incompatible with the existing application without installing a necessary update. Often the identification of the incompatibility occurs at a most unfortunate time such as, for example, prior to the user having an opportunity to save the work in progress. This experience is frustrating, time consuming and can be costly for the user.

SUMMARY OF THE INVENTION

A method for remotely updating software in a plurality of computer systems is disclosed. In one embodiment, a request for an upgrade is sent to a server system connected in a network. The upgrade is for a software application installed in a client system connected in the network. The request is sent from the client system. The request comprises a unique identification associated with the client system. The unique identification is recognized by the server system as belonging to the client system. At least one instruction is received from the server system in response to the request for the upgrade. The server system has knowledge of the software application installed on the client system. The at least one instruction directs the client system to collect application information about the software application installed on the client system. The server system has no knowledge whether most-updated upgrade packages available for the software application have been installed on the client system. The application information about the software application is sent to the server system. The server system performs a comparison between the application information about the software application and the most-updated upgrade package for the software application. The most-updated upgrade

APPLE EXHIBIT 1102, Page 893 of 1048

package for the software application is stored in a part database. The most-updated upgrade package for the software application is received by the client system automatically when the comparison indicates that the most-updated upgrade package has not been installed on the client system.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example in the following drawings in which like references indicate similar elements. The following drawings disclose various embodiments of the present invention for purposes of illustration only and are not intended to limit the scope of the invention.

Figure 1 is a network diagram illustrating one embodiment of components connected in a network that can be used with the method of the present invention.

Figure 2 is a flow diagram illustrating one embodiment of an update process.

Figure 3 is another flow diagram illustrating one embodiment of the update process.

Figure 4 is an exemplary tool bar that can be used with one method of the present invention.

Figure 5 is an exemplary diagram illustrating a relationship between a server connection point, a customer data base and a part data base.

Figure 6 is an exemplary diagram illustrating a communication protocol between a client system and the server through the Internet network

Figure 7 illustrates one embodiment of a computer-readable medium containing various sets of instructions, code sequences, configuration information, and other data used by a computer or other processing device.

APPLE EXHIBIT 1102, Page 894 of 1048

DETAILED DESCRIPTION

A method and apparatus for remotely updating software in a plurality of computer systems is disclosed. In the following description, for purposes of explanation, specific nomenclature is set forth to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention.

Some portions of the detailed descriptions that follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of operations leading to a desired result. The operations are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussion, it is appreciated that throughout the description, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within

the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computer selectively activated or reconfigured by a computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, and magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), EPROMs, EEPROMs, magnetic or optical cards, or any type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general-purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method operations. The required structure for a variety of these systems will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

In one embodiment, the method disclosed in the present invention allows for better remote support of users of client systems in the network. A server provides update information to multiple client systems connected in a network. When necessary, the updates are retrieved from a central depository, sent to the appropriate client systems and automatically update the applications. In one embodiment, the client systems are IBM-compatible personal computers running in the Window environment such as, for example, Windows 98, Windows 2000, etc. The server and the client systems are connected in a network such as, for example, the Internet. By keeping the client systems updated, remote support can be efficiently performed to

APPLE EXHIBIT 1102, Page 896 of 1048

minimize the down time of the client systems. Each client system comprises of multiple installed software packages. The software packages may have been previously installed on the client system prior to delivery to a user. The software may include, for example, application software, device drivers, etc.

Figure 1 illustrates an exemplary embodiment of the update network. A server 105 maintains a client database 125 to keep track of the client systems 110, 115. For example, whenever a client system 110 or 115 communicates with the server 105, the server 105 already knows about the installed software on that client system 110, 115. The server 105 also maintains a part database 120 containing software patches and software updates to help keeping the client systems 110 and 115 up to date. The client database 125 allows the server 105 to know about the configuration of the client systems 110 and 115. The client database 125 and the part database 120 may be in the same database server or in separate database servers connected in the network 130. Alternatively, the client database 125 and the part database 120 may be in the same system as the server 105. In one embodiment, the server 105 serves as a central point for receiving update requests from the client systems 110 and 115 and for retrieving information from the databases 125 and 120 to satisfy the update requests.

Figure 2 is a flow diagram 200 illustrating one embodiment of an update method. At block 205, an update request is generated by the client system 110, 115 and sent to the server 105. The update is performed on a periodic basis, such as, for example, every 24 hours. Alternatively, the update may be performed at any time by the user sending an update request to the server 105 on the network. The server 105 knows each client system 110, 115 by a unique identification associated with the client system 110, 115.

In one embodiment, the server 105 accesses a client database 125 containing information about the client system 110, 115. The client database 125 may include information, such as, for example, installed software packages on the client system 110, 115, the operating system installed on the client system 110, 115, etc. However, what the server 105 may not know is whether

these installed software packages are up to date. For example, the user of the client system 110, 115 may have changed the configuration parameters of the software packages, or the user may not have requested for an update for an extended length of time due to the client system 110, 115 not being connected to the network 130.

In one embodiment, the client system 110, 115 may need to do a self-check and send its current software configuration to the server 105. A self-check may be done by the server 105 directing the client system 110, 115 specifically what to check for and the information to be collected from the client system 110,115. This information is the sent to the server 105, as shown in block 210. Based on this information, the server 105 checks its part database 120 and determines the updates that the client system 110, 115 needs. The updates are sent from the server 105 to the client system 110, 115, as shown in block 215. The updates may be sent with instructions from the server 105 that tells the client system 110, 115what to do to have the updates installed, as shown in block 220.

Figure 3 is another flow diagram illustrating one embodiment of an update method 300. In one embodiment, a utility program executed by the client system 110, 115 communicates with the server 105 for information to check on the client system 110, 115. The execution of this utility program may be initiated by the user or it may be automatic. The utility program is herein referred to as a patch checker. The patch checker initiates the request to have the applications verified for any necessary updates. The request is sent to the server 105 along with the unique identification number of the client system 110, 115. The server 105 uses the client identification number to check against the client database 125 for authentication. In one embodiment, the database contains configuration information about the client system 110, 115. The server 105 retrieves the configuration information for the client system 110, 115, generates a script file and sends the script file to the client system 110, 115, as shown in block 305. In one embodiment, the script file contains commands that tell the client system 110, 115 the functions to perform. For example, the

APPLE EXHIBIT 1102, Page 898 of 1048

commands may direct the client system 110, 115 to perform self-check functions. The self-check functions may have the following parameters:

- 'v: filename' get the file's version
- 'm: filename' get the file's modified date
- 'd: driveletter' get amount of free disk space
- 'r: keyname' get the value of the specified registry key
- 's: filename' get the size of the file.

In one embodiment, the commands are executed by the client system 110, 115 to collect information pertinent to the applications currently installed on the client system 110, 115.

The script file may contain a list of parts that the server 105 thinks the client system 110, 115 has and that the server 105 wants the client system 110, 115 to check. The parts may be the names of the applications and the server 105 may want the client system 110, 115 to collect the current version information about the applications. In one embodiment, in order to keep the information in the client database 125 accurate, the user may not want to alter the configuration of the applications that are to be supported remotely by the server 105. Keeping the client system 110, 115 and the information in the client database 125 synchronized may help making the update process by the server 105 more efficient.

In block 310, using the script information sent by the server 105, the patch checker parses the server's commands to check the software parts on the client system 110, 115. The appropriate information about these software parts is collected. In one embodiment, the version of each software part is collected and sent to the server 105, as shown in block 315. The server 105 uses the information collected from the client system 110, 115 and compares it with a part database 120. For example, the server 105 may check the version number collected from the client system 110, 115 with the version of the same software part in the part database 120. In one embodiment, the server 105 may want to get the most updated version distributed to the client system 110, 115.

APPLE EXHIBIT 1102, Page 899 of 1048

When the version information collected from the client system 110, 115 is not at the same level with the version of the same software part in the part database 120, the most updated version is retrieved from the part database 120. When the version information from the client system 110, 115 is already up to date, there is nothing to download. In block 320, the patch checker asks the server 105 for the files associated with the updated versions of the software to download. The files are downloaded from the server 105 to the client system 110, 115 in block 325. In one embodiment, each download file is associated with an uniform resource locator (URL). The server 105 replies to the update request by sending the patch URL.

There may be one or more download files for each software to be updated, and there may be more than one software that needs to be updated, the server 105 may send several down load files to the client system 110, 115. The download files may be stored in a predefined directory such as, for example, the download directory. Each download file is processed individually, as shown in block 330. In one embodiment, the download files are received from the server 105 in a compressed format, such as, the zip format, and need to be uncompressed or expanded, as shown in block 335. Each download file is expanded into an executable program and multiple related data files. One of the data files is a text file or an instruction file containing a set of instructions or commands that can be parsed by the executable program to perform the update process, as shown in block 340. For example, the instruction may be one of the following commands:

- Delete a file
- ShellExecute
- ShellExecute with wait
- Registry Change
- Add message to a tool bar
- Kill a particular process
- Ask for reboot
- Force reboot

- Ask user to install now or later
- Ask user to close all programs

When all of the download files have been expanded and copied into the appropriate directories, the update process is completed. At that time, the user may be given an option of rebooting the client system 110, 115 to activate the updated version. Alternatively, the user may continue working with the currently installed version and reboot the client system 110, 115 at a later time.

Figure 4 illustrates an exemplary tool bar that can be used with the present invention. In one embodiment, the tool bar is a list of dynamic link libraries (DLL) and is always running. Additional functions can be added to the tool bar by adding DLL files. For example, the patch checker can be added to the tool bar 400 by adding a patcher.dll to the tool bar 400, and a patch checker icon can be displayed. By selecting the patch checker icon, the user can initiate the update process at any time. In one embodiment, the tool bar 400 is also used to display update related messages to the user.

Figure 5 is an exemplary diagram illustrating a relationship between a connection point, a customer data base and a part data base. In one embodiment, the server provides a connection point 505 that connects to a customer database 510. The customer database 510 maintains the state of every client system in the network. The state includes information concerning relevant hardware and software as currently installed on the client system. This information includes, for example, the versions of the installed software applications, the versions of the installed hardware drivers, etc. Additionally, the connection point 505 is also connected to a part database 515. The part database 515 may contain the different versions of the application software, the DLLs, the hardware drivers, and any other software modules that may be installed on the client system. The server uses the part database 515 to keep the client system up to date. For example, when the client system is identified to have a hardware driver that is not current, the most up-to-date hardware driver is retrieved from the part database 515.

APPLE EXHIBIT 1102, Page 901 of 1048

In one embodiment, a client part database is maintained in the client system. The client part database contains the versions of the software that are installed on the client system. As additional software is installed on the client system, the client part database is updated accordingly. In one embodiment, when the server wants to know the versions of the software installed on the client system, the patch checker retrieves the version information from the client part database.

Figure 6 is an exemplary diagram illustrating a communication protocol between a client system 600 and a server 650 through the Internet network. In one embodiment, the client system 600 has a message queue 605 to store messages that certain application 610, such as, for example, the patch checker (patcher.dll), wants to send to the server 650. The user selects the patch checker icon on the tool bar 400 displayed by the tool bar program 620 (dreambar.exe) to execute the patch checker 610. The message queue 605 is processed periodically, such as, for example, every thirty minutes, while the user is connected to the Internet 690. When the user is not connected to the Internet 690, the messages from the patch checker (applications) 610 are stored in the message queue 605. In one embodiment, the patch checker 610 connects to the server 650 through a message handler 615 (ConPoint.dll). The message handler 615 handles the messages generated by the patch checker 610 including, for example, the request for an update. The message handler 615 sends the message to the server 650. In another embodiment, the message queue 605 is implemented as a text file located in a message queue directory.

In one embodiment, the server 650 is implemented with multiple java servlets. A master servlet 655 (AnnexServlet) is used to route all the messages received from the client systems 600 to the other servlets 665, 670 on the server 650. Each of the servlets 660, 665, 670 handles different type of messages. In one embodiment, as each servlet starts up, the servlet tells the master servlet which type of messages the servlet 660, 665, 670 handles. The master servlet 655 may be used as the connection point on the server 650. Each of the servlets 660, 665, 670 may be used as a worker. For example, the serverlet 660 is the

APPLE EXHIBIT 1102, Page 902 of 1048

patch worker handling the update messages from the patch checker 610. The patch worker 660 sends the script file to the patch checker 610. The script file is used by the patch checker 610 to check the client system 600. When the patch checker 610 requests for the download, the patch worker 660 accesses the part database 665 to retrieve the necessary software versions for the client system 600. It will be apparent to one skilled in the art that there may be other workers (servlets) on the server 650, such as, for example, a buildworker to add a new client system to the client database, a viewworker to view contents of the client database 680 and the part database 675, a dataworker to store data, and a messageworker to get the messages to be displayed on the tool bar 400.

In one embodiment, each client system 600 is associated with a unique identification number known to the server 650. As a new client system 600 is inserted into the network, the client database 680 is updated with the identification number of that new client system 600. Similarly, when the client system 600 is removed from the network, the client database 680 is updated accordingly. In one embodiment, the server 650 generates a report listing all the identification number of those client systems 600 that have not communicated with the server 650 for over a predetermined length of time. The report can then be used to investigate status of these client systems.

Figure 7 illustrates an embodiment of a computer-readable medium 700 containing various sets of instructions, code sequences, configuration information, and other data used by a computer or other processing device. The embodiment illustrated in **Figure 7** is suitable for use with the software update method described above. The various information stored on medium 700 is used to perform various data processing operations. Computer-readable medium 700 is also referred to as a processor-readable medium. Computer-readable medium 700 can be any type of magnetic, optical, or electrical storage medium including a diskette, magnetic tape, CD-ROM, memory device, or other storage medium.

Computer-readable medium 700 includes interface code 705 that controls the flow of information between various devices or components in the

APPLE EXHIBIT 1102, Page 903 of 1048

computer system. Interface code 705 may control the transfer of information within a device (e.g., between the processor and a memory device), or between an input/output port and a storage device. Additionally, interface code 705 may control the transfer of information from one device to another or from one network component to another.

Computer-readable medium 700 also includes the patch checker program 710 that is used to request and receive software patches or updates from the server. Other codes stored on the computer-readable medium 700 may include the tool bar program 715 to display the patch checker icon, the message queue handler program 720 to receive the messages generated by the patch checker and send the messages to the server. The computer-readable medium 700 may also contain programs run on the server. These programs may include the patch worker 725 that communicates with the patch checker 710 from the server side, and the database access program 730 that allows the server to view the client database and the part database.

From the above description and drawings, it will be understood by those of ordinary skill in the art that the particular embodiments shown and described are for purposes of illustration only and are not intended to limit the scope of the invention. Those of ordinary skill in the art will recognize that the invention may be embodied in other specific forms without departing from its spirit or essential characteristics. References to details of particular embodiments are not intended to limit the scope of the claims.

APPLE EXHIBIT 1102, Page 904 of 1048

CLAIMS

What is claimed is:

1. A method comprising:

sending a request for an upgrade to a server system connected in a network, the upgrade being for a plurality of software applications installed in a client system connected in the network, the request sent from the client system, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

receiving at least one instruction from the server system in response to the request for the upgrade, the server system having a knowledge of the software applications installed on the client system, the at least one instruction directing the client system to collect application information about the software applications installed on the client system, the server system having no knowledge whether most-updated upgrade packages available for the software applications have been installed on the client system;

sending the application information about the software applications to the server system, wherein the server system performs a comparison between the application information about the software applications and the most-updated upgrade packages for the software applications, wherein the most-updated upgrade packages for the software applications are stored in a part database; and

receiving the most-updated upgrade packages for the software applications at the client system automatically when the comparison indicates that the most-updated upgrade packages have not been installed on the client system.

APPLE EXHIBIT 1102, Page 905 of 1048

2. The method of claim 1, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

3. The method of claim 2, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software applications installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

4. The method of claim 3, wherein the knowledge of the software applications installed on the client system comprises names of the software applications installed on the client system.

5. The method of claim 3, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

6. The method of claim 5, wherein the network is an Internet.

7. The method of claim 5, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

8. The method of claim 5, wherein the client database is in the server system.

APPLE EXHIBIT 1102, Page 906 of 1048

9. The method of claim 1, wherein the application information about the software applications comprises version information of the software applications, and wherein the application information about the software applications is stored in a database in the client system.

10. The method of claim 1, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

11. The method of claim 10, wherein the predetermined time interval is 24 hours.

12. The method of claim 1, wherein the request is sent at any time by a user using the client system.

13. The method of claim 1, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

14. The method of claim 1, wherein receiving the most-updated upgrade packages for the software applications from the server system comprises receiving a plurality of download files.

15. The method of claim 14, wherein the plurality of download files are in a compressed format.

16. The method of claim 15, wherein each download file comprises an upgrade utility, a text file and a plurality of data files, wherein the text

APPLE EXHIBIT 1102, Page 907 of 1048

file provides commands to the upgrade utility to install the plurality of data files in the client system.

17. The method of claim 16, wherein the commands comprise copy a file, delete a file, registry change, ask for reboot, force reboot, ask the user to install now or later, and ask the user to close all programs.

18. The method of claim 16, wherein each down load file is associated with a uniform resource locator (URL), and wherein each down load file is retrieved by accessing the URL.

19. A machine-readable medium providing instructions, which when executed by a set of one or more processors, cause said set of processors to perform the following:

sending a request for an upgrade to a server system connected in a network, the upgrade being for a plurality of software applications installed in a client system connected in the network, the request sent from the client system, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

receiving at least one instruction from the server system in response to the request for the upgrade, the server system having a knowledge of the software applications installed on the client system, the at least one instruction directing the client system to collect information about the software applications installed on the client system, the server system having no knowledge whether most-updated upgrade packages available for the software applications have been installed on the client system;

sending the information about the software applications to the server system, wherein the server system performs a comparison between

APPLE EXHIBIT 1102, Page 908 of 1048

the information about the software applications and the most-updated upgrade packages for the software applications, wherein the most-updated upgrade packages for the software applications are stored in a part database; and

receiving the most-updated upgrade packages for the software applications to the client system automatically when the comparison indicates that the most-updated upgrade packages have not been installed on the client system.

20. The machine-readable medium of claim 19, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

21. The machine-readable medium of claim 20, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software applications installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

22. The machine-readable medium of claim 21, wherein the knowledge of the software applications installed on the client system comprises names of the software applications installed on the client system.

APPLE EXHIBIT 1102, Page 909 of 1048

23. The machine-readable medium of claim 21, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

24. The machine-readable medium of claim 23, wherein the network is an Internet.

25. The machine-readable medium of claim 23, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

26. The machine-readable medium of claim 23, wherein the client database is in the server system.

27. The machine-readable medium of claim 19, wherein the information about the software applications comprises version information of the software applications, and wherein the information about the software applications is stored in a database in the client system.

28. The machine-readable medium of claim 19, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

29. The machine-readable medium of claim 28, wherein the predetermined time interval is 24 hours.

30. The machine-readable medium of claim 19, wherein the request is sent at any time by a user using the client system.

APPLE EXHIBIT 1102, Page 910 of 1048

31. The machine-readable medium of claim 19, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

32. The machine-readable medium of claim 19, wherein receiving the most-updated upgrade packages for the software applications from the server system comprises receiving a plurality of download files.

33. The machine-readable medium of claim 32, wherein the plurality of download files are in a compressed format.

34. The machine-readable medium of claim 33, wherein each download file comprises an upgrade utility, a text file and a plurality of data files, wherein the text file provides commands to the upgrade utility to install the plurality of data files in the client system.

35. The machine-readable medium of claim 34, wherein the commands comprise copy a file, delete a file, registry change, ask for reboot, force reboot, ask the user to install now or later, and ask the user to close all programs.

36. The machine-readable medium of claim 34, wherein each down load file is associated with a uniform resource locator (URL), and wherein each down load file is retrieved by accessing the URL.

37. In an arrangement comprising at least one computer network, the network connecting at least one server computer to at least one client computer, a data processing system for providing software upgrades to the client computer, comprising:

means for generating a request for a software upgrade, the software upgrade being for an application on the client computer;

means for processing the request for the software upgrade comprising:

means for retrieving current information about the application on the client computer;

means for comparing the current information about the application on the client computer with information about an updated package for the application, the updated package stored in a part database accessible by the server computer; and

means for sending the updated package from the part database to the client computer when the current information about the application on the client computer and the information about the updated package are not the same.

38. The data processing system of claim 37 further comprising means for verifying a unique identification associated with the client system against a client database accessible by the server computer, the client database comprising the unique identification associated with the client computer, the client computer previously registered with the server computer.

39. The data processing system of claim 37, wherein the current information about the application on the client computer and the information about the updated package comprise version information.

40. The data processing system of claim 37, wherein the request for the software upgrade is generated automatically at a predetermined time interval.

APPLE EXHIBIT 1102, Page 912 of 1048

41. In an arrangement comprising at least one computer network, the network connecting at least one server computer to at least one client computer, a data processing system for providing software upgrades to the client computer, comprising:

a first logic in the client computer to generate a request for a software upgrade, the software upgrade being for an application installed on the client computer; and

a second logic in the server computer to process the request for the software upgrade received from the first logic, the second logic comprising:

logic to extract current information about the application installed on the client computer;

logic to compare the current information about the application installed on the client computer with information about a most updated upgrade package for the application installed on the client computer, the most updated upgrade package stored in a first database; and

logic to send the most updated upgrade package for the application to the client computer when the current information about the application installed on the client computer does not match the information about the most updated upgrade package for the application.

42. The data processing system of claim 41 further comprising a third logic on the server computer to verify an identification associated with the client computer against a set of valid identifications stored in a client database, wherein the identification associated with the client computer is added to the set of valid identifications when the client computer is registered with the server computer.

43. The data processing system of claim 41, wherein the current information about the application installed on the client computer comprises version numbers of the applications.

44. The data processing system of claim 41, wherein the request for software upgrade is automatically generated by the client computer at a predetermined time interval.

45. The data processing system of claim 41, wherein the request for software upgrade is generated at any time by a user.

46. A method comprising:

receiving a request for an upgrade from a client system connected in a network, the upgrade being for a software application installed in a client system, the request received at a server system connected to the network, the request comprising a unique identification associated with the client system, the unique identification recognized by the server system as belonging to the client system;

sending at least one instruction from the server system to the client system in response to the request for the upgrade, the server system having a knowledge of the software application installed on the client system, the at least one instruction directing the client system to collect information about the software application installed on the client system, the server system having no knowledge whether most-updated upgrade package available for the software application have been installed on the client system;

receiving the information about the software application from the client system, wherein the server system performs a comparison between the information about the software application and the most-updated upgrade package for the software application, wherein the most-

APPLE EXHIBIT 1102, Page 914 of 1048

updated upgrade package for the software application is stored in a database; and

sending the most-updated upgrade package for the software application to the client system automatically when the comparison indicates that the most-updated upgrade package have not been installed on the client system.

47. The method of claim 46, wherein the unique identification for the client system is stored in a registry in the client system, wherein the unique identification is recognized by the server system as belonging to the client system when the unique identification is found in a plurality of unique identifications stored in a client database, wherein when the unique identification is not found, the client system is not authenticated.

48. The method of claim 47, wherein the client database comprises a configuration file for each client system connected in the network, the configuration file providing the server system the knowledge of the software application installed on the client system, wherein the server system uses the configuration file to generate the at least one instruction.

49. The method of claim 48, wherein the knowledge of the software application installed on the client system comprises name of the software application installed on the client system.

50. The method of claim 48, wherein the client database is in a first database server connected to the network, and wherein the part database is in a second database server connected to the network.

51. The method of claim 50, wherein the network is an Internet.

APPLE EXHIBIT 1102, Page 915 of 1048

52. The method of claim 50, wherein functions associated with the first database server and functions associated with the second database server are implemented in the server system.

53. The method of claim 50, wherein the client database is in the server system.

54. The method of claim 46, wherein the information about the software application comprises version information of the software application, and wherein the information about the software application is stored in a database in the client system.

55. The method of claim 46, wherein the request for the upgrade is sent automatically from the client system to the server system at a predetermined time interval.

56. The method of claim 46, wherein the request is sent at any time by a user using the client system.

57. The method of claim 46, wherein the at least one instruction received from the server system comprises get a version information for a file, get a modified date for the file, get a size of the file, get an amount of free disk space for a storage drive, and get a value of a registry key.

58. The method of claim 46, wherein receiving the most-updated upgrade package for the software application from the server system comprises at lest one download file.

APPLE EXHIBIT 1102, Page 916 of 1048

59. The method of claim 58, wherein the at least one download file is associated with a uniform resource locator (URL), and wherein the at least one down load file is retrieved by accessing the URL.

APPLE EXHIBIT 1102, Page 917 of 1048

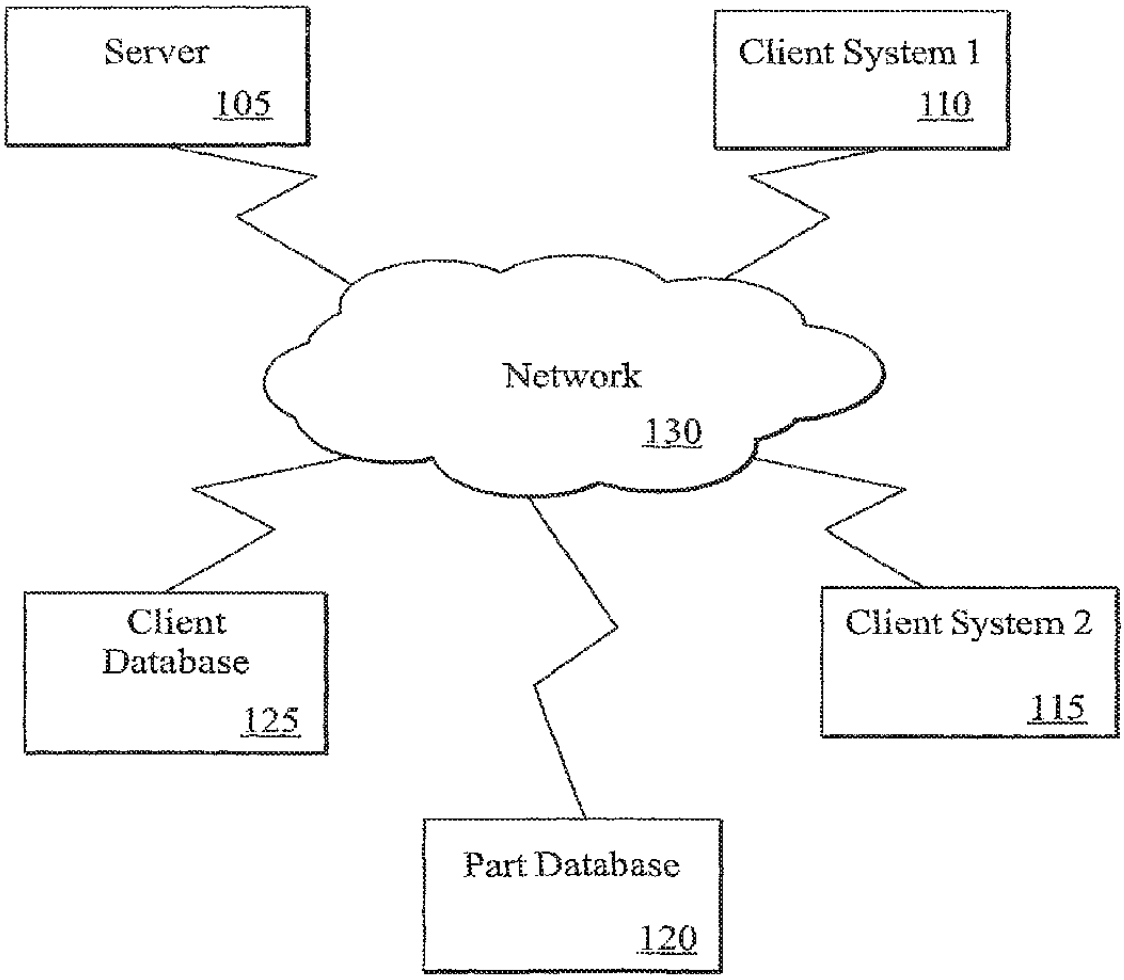

Server
105
Client System 1
110
Network
130
Client
Database
125
Client System 2
115
Part Database
120

Fig. 1

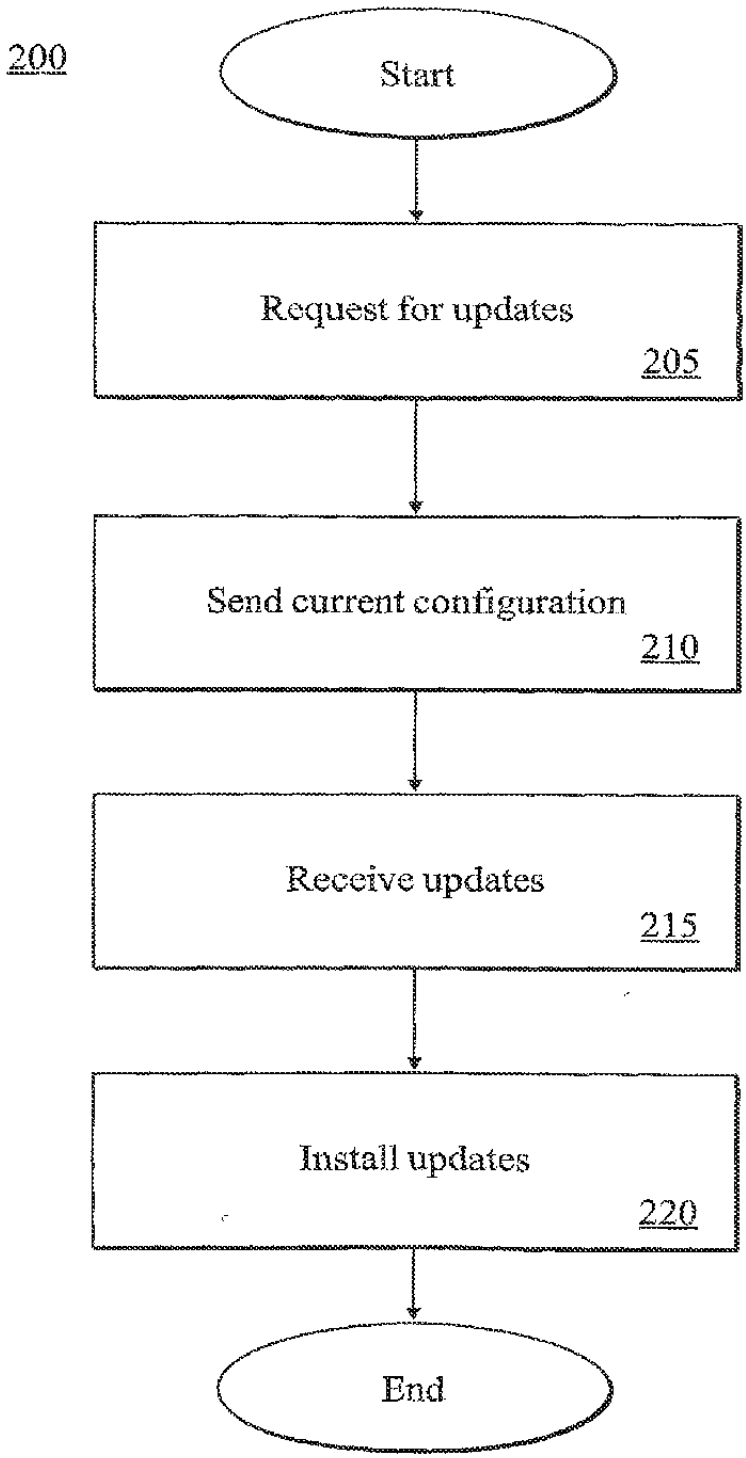

200
Start
Request for updates
205
Send current configuration
210
Receive updates
215
Install updates
220
End

Fig. 2

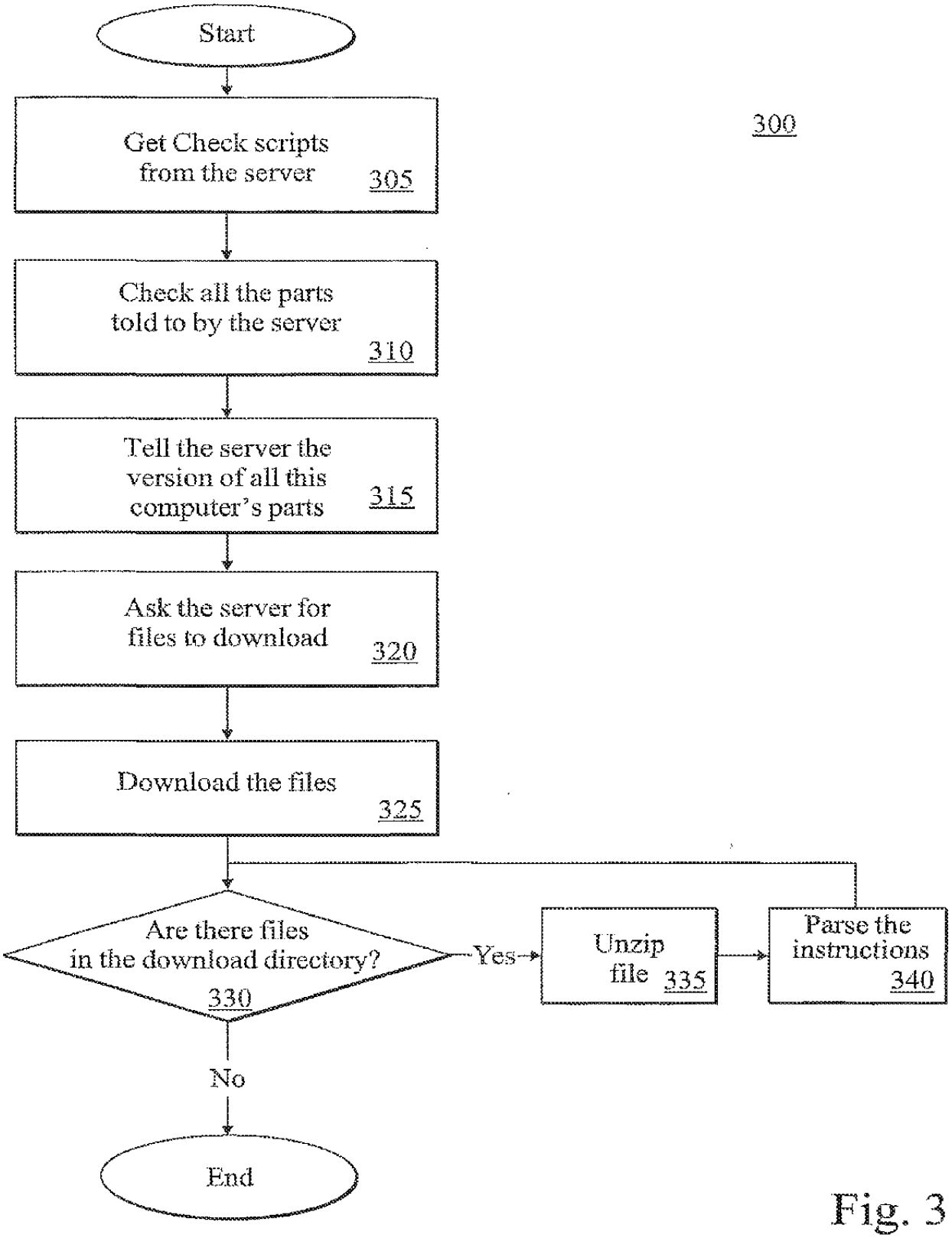

Start
Get Check scripts from the server 305
300
Check all the parts told to by the server 310
Tell the server the version of all this computer's parts 315
Ask the server for files to download 320
Download the files 325
Are there files in the download directory? 330
Yes
Unzip file 335
Parse the instructions 340
No
End

Fig. 3

400
Help
Marco Polo
explore the internet
Personalize your dreambar.
right-click on the middle graphic.
BizCenter

Fig. 4

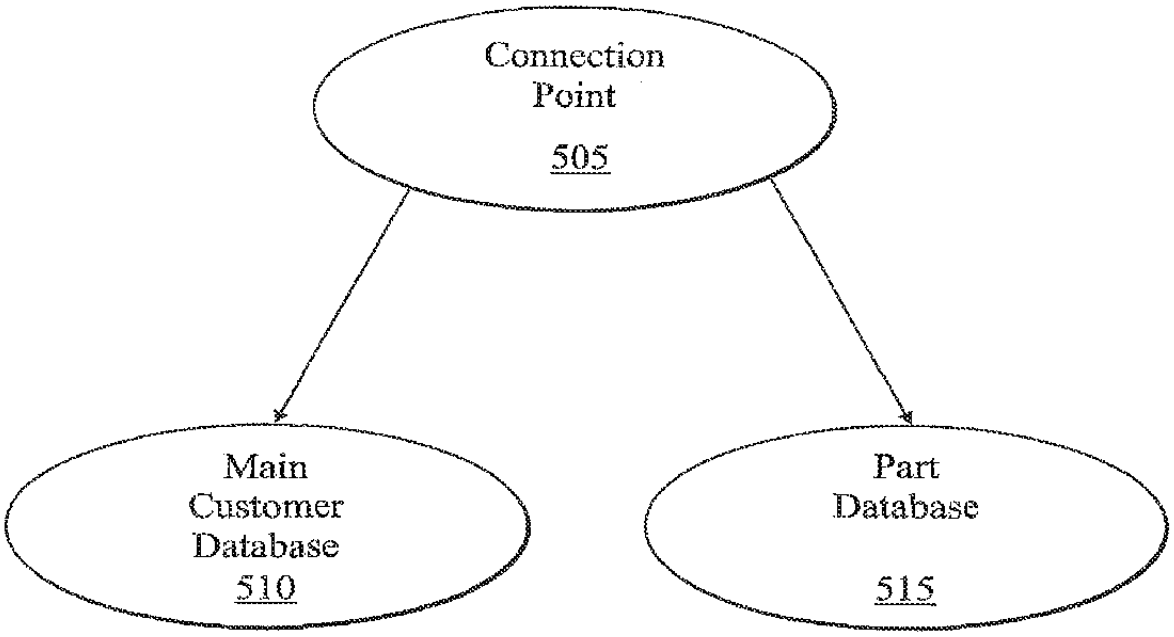

Connection Point
505
Main Customer Database
510
Part Database
515

Fig. 5

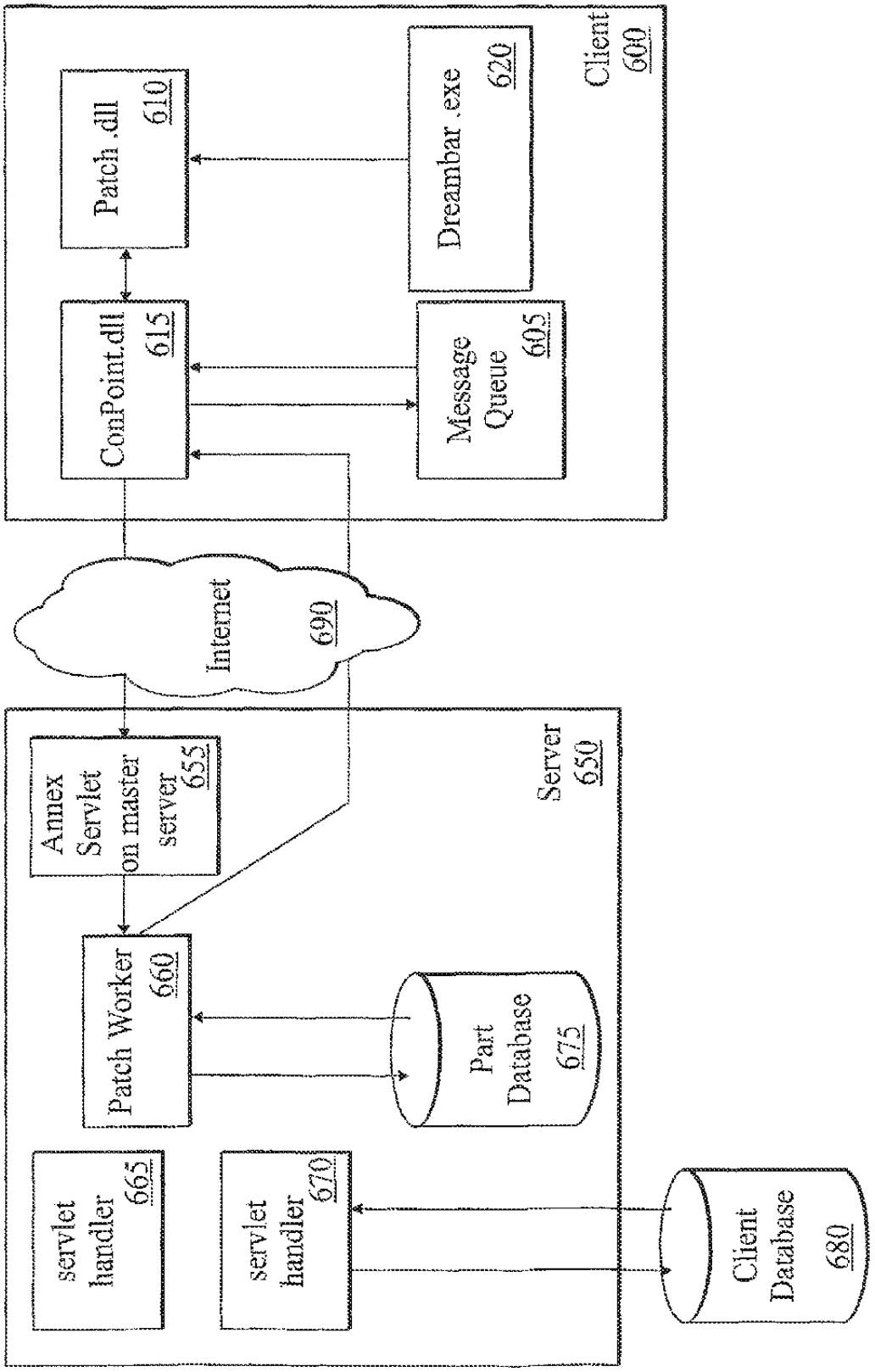

Client
600
Patch .dll
610
Dreambar .exe
620
ComPoint.dll
615
Message
Queue
605
Internet
690
Server
650
Annex
Servlet
on master
server
655
Patch Worker
660
Part
Database
675
servlet
handler
665
servlet
handler
670
Client
Database
680

Fig. 6

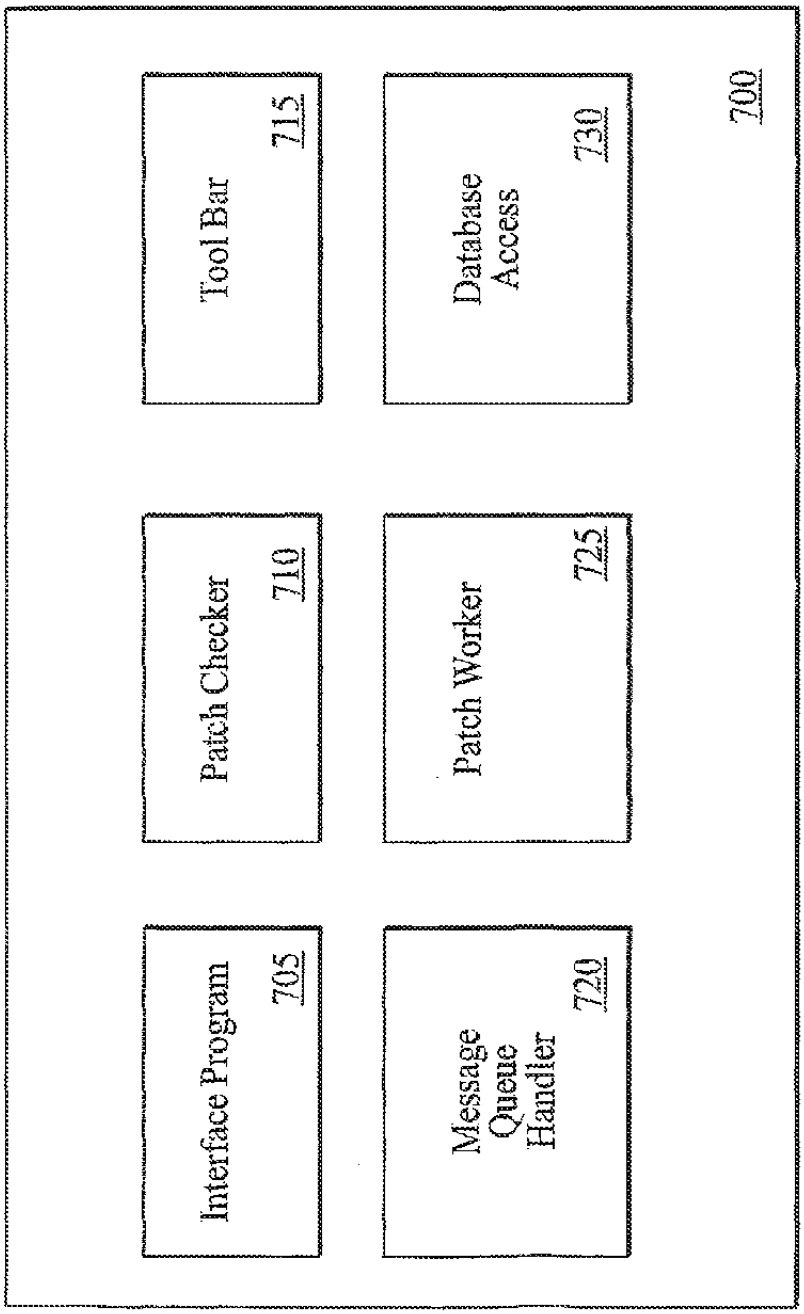

Interface Program
705
Patch Checker
710
Tool Bar
715
Message Queue Handler
720
Patch Worker
725
Database Access
730
700

Fig. 7

INTERNATIONAL SEARCH REPORT

Internation Ication No
PCT/US 01/15720

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/445 G06F1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5 845 077 A (FAWCETT PHILIP E) 1 December 1998 (1998-12-01) column 4, line 60 -column 9, line 27	1-59
A	EP 0 809 182 A (NIPPON ELECTRIC CO) 26 November 1997 (1997-11-26) column 4, line 3 -column 5, line 53	1,2,19, 20,37, 38,41, 42,46,47
A	US 5 752 042 A (PRITKO STEVEN MICHAEL ET AL) 12 May 1998 (1998-05-12) column 3, line 7 -column 6, line 60	1-59
A	US 6 006 034 A (PORT GRAEME ET AL) 21 December 1999 (1999-12-21) column 4, line 59 -column 6, last line	1-59

[ ] Further documents are listed in the continuation of box C. [X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 October 2001	06/11/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2046, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bijn, K

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US 01/15720

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5845077 A	01-12-1998	US 6073214 A	06-06-2000
EP 0809182 A	26-11-1997	JP 9305675 A	28-11-1997
		AU 2348897 A	27-11-1997
		CA 2204317 A1	20-11-1997
		EP 0809182 A1	26-11-1997
US 5752042 A	12-05-1998	US 6074434 A	13-06-2000
US 6006034 A	21-12-1999	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
12/818,906	06/18/2010	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☒ RATE ($)	SMALL ENTITY ☒ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 11/14/2011	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 19	Minus	** 20	= 0	X $30 =	0	OR	X $ =	
Independent (37 CFR 1.16(h))	* 3	Minus	*** 3	= 0	X $125 =	0	OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)		RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/BRUCE HARRISON/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

96051 7590 12/30/2011

Uniloc USA Inc.
2151 Michelson Ste. 100
Irvine, CA 92612

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

MAIL DATE	DELIVERY MODE
12/30/2011	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	12/818,906	ETCHEGOYEN, CRAIG STEPHEN	
	Examiner	Art Unit	
	QING CHEN	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 November 2011*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

5)☒ Claim(s) *1-17,21 and 22* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-17,21 and 22* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

12)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

13)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *20111114*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on November 14, 2011.

2. **Claims 1-17, 21, and 22** are pending.

3. **Claims 1, 5, and 6** have been amended.

4. **Claims 18-20** have been canceled.

5. **Claims 21 and 22** have been added.

6. The objections to Claims 2, 3, and 5 are withdrawn in view of Applicant's amendments to the claims.

Response to Amendment

Claim Objections

7. **Claims 1, 6, 21, and 22** are objected to because of the following informalities:

- Claims 1, 21, and 22 recite the limitations "the device identifier" and "the software identifier." They should read -- the *unique* device identifier -- and -- the *unique* software identifier --, respectively.

- Claim 6 recites the limitation "the geo-location codes." It should read -- the *one or more* geo-location codes --.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

8. The following is a quotation of the second paragraph of 35 U.S.C. 112:

> The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

9. **Claims 1-17 and 21** are rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 1 recites the limitation "features of the software." There is insufficient antecedent basis for this limitation in the claim. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "features of software" for the purpose of further examination.

Claims 1 and 21 recite the limitation "the particular user of the software." There is insufficient antecedent basis for this limitation in the claims. In the interest of compact prosecution, the Examiner subsequently interprets this limitation as reading "a particular user of the software" for the purpose of further examination.

Claims 2-17 depend on Claim 1 and, therefore, suffer the same deficiencies as Claim 1.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

11. **Claims 1-17 and 21** are rejected under 35 U.S.C. 103(a) as being unpatentable over US 6,467,088 (hereinafter "alSafadi") in view of US 6,880,086 (hereinafter "Kidder") and US 2009/0037337 (hereinafter "Baitalmal").

As per Claim 1, alSafadi discloses:

A system for remotely updating a program configuration (col. 7 lines 46 and 47, "An apparatus for controlling the reconfiguration of an electronic device …"), **comprising a client device** (Figure 1: 12) **and an update server** (Figure 1: 10) **wherein:**

- **(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device** (col. 5 lines 63-66, "The functional operations associated with the … electronic device 12, as described in detail in conjunction with FIGS. 1 and 2, may be implemented in whole or in part in one or more software programs …"), **the client device comprising:**
  - **a first processor** (Figure 3: 220) **coupled to memory** (Figure 3: 222) **storing the computer program which, when executed by the processor (iii) collects a unique software identifier for the software on the client device, the software identifier being unique to a particular copy of the software** (col. 3 lines 20-24, "The device 12 includes [collects] a number of software components 14A, 14B and 14C, corresponding to version 1.1 [unique software identifier] of a software component A, version 2.3 of a software component B, and version 2.0 of a software component C, respectively."); **and**

- **a first transceiver configured to send the unique device and software identifiers to an update server via the Internet** (col. 4 lines 39-47, "... the reconfiguration manager 10 obtains information regarding the hardware and software configuration [unique device and software identifiers] of device X, i.e., electronic device 12 of FIG. 1. This information is generally included as part of the request 20 sent by the device 12 to the reconfiguration manager 10. In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier of the electronic device (emphasis added)."; col. 6 lines 1-3, "The network 214 may represent a global computer communications network such as the Internet ..."); **and**

- **(b) the update server is configured to receive the unique device and software identifiers from the client device** (col. 4 lines 39-47, "... the reconfiguration manager 10 obtains [receives] information regarding the hardware and software configuration [unique device and software identifiers] of device X, i.e., electronic device 12 of FIG. 1 ... In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier of the electronic device."), **the update server comprising:**

- **a second processor** (Figure 3: 230) **coupled to memory** (Figure 3: 232) **and configured to analyze the unique device and software identifiers at the update server, and to determine based on the analyzed unique device and software identifiers an updated program configuration** (col. 5 lines 8-12, "If step 112 indicates [analyzes] that the set is not empty, a particular set of upgrade configuration is selected [determined] in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X."); **and**

- **a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein** (col. 5 lines 16-18, "The reconfiguration manager or other server associated therewith then downloads [delivers] the upgrade to device X in step 120.").

alSafadi does not explicitly disclose:

- **a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters.**

However, Kidder discloses:

- **a first processor coupled to memory storing a computer program which, when executed by the processor (i) performs physical device recognition on a client device to determine machine parameters** (col. 61 lines 9-13, "Also within modular system services is a Master Control Driver (MCD) that learns the physical characteristics of the particular computer system on which it is running, in this instance, computer system 10."), **and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters** (col. 152 lines 32-42, "... the Master MCD (Master Control Driver) 38 takes a physical inventory of the network device (e.g., computer system 10, FIG. 1, network device 540, FIGS. 35, 59) and assigns a unique physical identification number (PID) to each physical component within the system, including the network device itself ...").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

alSafadi also does not explicitly disclose:

- **account information for a user of the client device and features of software that the user of the client device is entitled to use;**
- **the software identifier being unique to a particular user of the software; and**
- **if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier.**

However, Baitalmal discloses:

- **account information for a user of a client device and features of software that the user of the client device is entitled to use** (paragraph [0387], "An account is a data structure associated with a user … where the account data structure includes information regarding which software applications are licensed to the user."; paragraph [0389], "A user account allows a user (person licensed to use software applications provided by the marketplace) to authenticate to system services. The user account also provides a user with the opportunity to be authorized to

access deployed software applications. Users may need to identify themselves for the purposes of accounting, security, logging and resource management. In some embodiments, a user identifies him or herself using an account identifier and a username, and in some embodiments the user also has a password.");

- **a software identifier being unique to a particular user of the software** (paragraph [0582], "… the apply_user table 6806-1 can include foreign keys referring to a data record in the users table 6804-1 associated with a given user (e.g., user_id) and a data record in the applications table associated with a given application (e.g., app_id).")**; and**

- **if the user associated with a device identifier is entitled to use features of an updated program configuration according to a license associated with the software identifier** (paragraph [0258], "User client device ID/type 416 can store information about the type or IDs of computer devices that the user has used to access applications on the application server."; paragraph [0261], "User enabled features 424 includes a list of features on the application server that have been enabled for the user. In some embodiments, user enabled features 424 are determined by the license (e.g., subscription, free, purchased, etc.) granted to the user.")**.**

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Baitalmal into the teaching of alSafadi to include account information for a user of the client device and features of software that the user of the client device is entitled to use; the software identifier being unique to a particular user of the software; and if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier. The

modification would be obvious because one of ordinary skill in the art would be motivated to prevent licensees of the software application from misusing the software application outside the terms of the license (Baitalmal, paragraph [0009]).

As per Claim 2, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the unique device identifier comprises a hash code.**

However, Kidder discloses:

- **wherein a unique identifier comprises a hash code** (col. 88 lines 62-64, "To avoid versioning errors, instead of assigning a version number, a signature is "machine generated" based on the content of the software component."; col. 89 lines 15-17, "The Sha-1 utility is a secure hash algorithm that uses the contents of a software component to generate a signature that is 20 bytes in length.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the unique device identifier comprises a hash code. The modification would be obvious because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 3, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.**

However, Kidder discloses:

- **wherein a computer program when executed implements at least one irreversible transformation such that machine parameters cannot be derived from a unique device identifier** (col. 89 lines 5-7, "In one embodiment, the signatures are generated using the "Sha-1" cryptography utility (often called the "sha1sum").").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier. The modification would be obvious because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 4, the rejection of Claim 3 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the at least one irreversible transformation comprises a cryptographic hash function.**

However, Kidder discloses:

- **wherein at least one irreversible transformation comprises a cryptographic hash function** (col. 89 lines 5-7, "In one embodiment, the signatures are generated using the "Sha-1" cryptography utility (often called the "sha1sum").").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the at least one irreversible transformation comprises a cryptographic hash function. The modification would be obvious because one of ordinary skill in the art would be motivated to eliminate errors often caused when humans generate version numbers (Kidder, col. 89 lines 21-23).

As per Claim 5, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the unique device identifier further comprises one or more geo-location codes.**

However, Kidder discloses:

- **wherein a unique identifier further comprises one or more geo-location codes** (col. 20 lines 1-4, "To configure a network device, the administrator begins by selecting (step 874, FIG. 3g) a particular network device to configure, for example, the network device corresponding to IP address 192.168.9.202 (FIG. 4f).").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the unique device identifier further comprises one or more geo-location codes. The

modification would be obvious because one of ordinary skill in the art would be motivated to access the client device using the Internet.

As per Claim 6, the rejection of Claim 5 is incorporated; and alSafadi does not explicitly disclose:

- **wherein at least one of the geo-location codes comprises an Internet Protocol address of the client device.**

However, Kidder discloses:

- **wherein at least one of geo-location codes comprises an Internet Protocol address of a client device** (col. 20 lines 1-4, "To configure a network device, the administrator begins by selecting (step 874, FIG. 3g) a particular network device to configure, for example, the network device corresponding to IP address 192.168.9.202 (FIG. 4f).").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein at least one of the geo-location codes comprises an Internet Protocol address of the client device. The modification would be obvious because one of ordinary skill in the art would be motivated to access the client device using the Internet.

As per Claim 7, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine**

bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model number) pertaining to the computer system.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 8, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model) pertaining to the CPU.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 9, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., model) pertaining to the optical drive.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 10, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., manufacturer) pertaining to the baseboard/motherboard.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 11, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.")**.** *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information (e.g., manufacturer) pertaining to the chassis.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 12, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the IDE controller.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 13, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer

system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the port connector.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 14, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item."). *[Examiner's Remarks: Note that one of ordinary skill in the art would readily*

comprehend that the physical inventory of the computer system includes information (e.g., size) pertaining to the cache.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 15, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the USB drive.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include

wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 16, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.")**.** *[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the device model.]*

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD. The modification would be

obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 17, the rejection of Claim 1 is incorporated; and alSafadi does not explicitly disclose:

- **wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.**

However, Kidder discloses:

- **wherein machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller** (col. 61 lines 29-31, "Master MCD 38 begins by taking a physical inventory of computer system 10 (over the I^2C bus) and assigning a unique physical identification number (PID) to each item.").

[Examiner's Remarks: Note that one of ordinary skill in the art would readily comprehend that the physical inventory of the computer system includes information pertaining to the PCI controller.]

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

As per Claim 21, alSafadi discloses:

A client device configured to execute a computer program to perform a remote update of a program configuration on the client device (col. 5 lines 63-66, "The functional operations associated with the … electronic device 12, as described in detail in conjunction with FIGS. 1 and 2, may be implemented in whole or in part in one or more software programs …"), **the client device comprising:**

- **a processor** (Figure 3: 220)**;**
- **memory** (Figure 3: 222) **coupled to the processor and storing the computer program which, when executed by the processor (iii) collects a unique software identifier for software on the client device, the software identifier being unique to a particular copy of the software** (col. 3 lines 20-24, "The device 12 includes [collects] a number of software components 14A, 14B and 14C, corresponding to version 1.1 [unique software identifier] of a software component A, version 2.3 of a software component B, and version 2.0 of a software component C, respectively."); **and**
- **a transceiver configured to (i) send the unique device and software identifiers to an update server via the Internet** (col. 4 lines 39-47, "… the reconfiguration manager 10 obtains information regarding the hardware and software configuration [unique device and software identifiers] of device X, i.e., electronic device 12 of FIG. 1. This information is generally included as part of the request 20 sent by the device 12 to the reconfiguration manager 10. In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier of the electronic device (emphasis added)."; col. 6 lines 1-3, "The network 214 may represent a global computer

communications network such as the Internet …"), **and (ii) receive from the update server an updated program configuration** (col. 5 lines 8-12, "If step 112 indicates that the set is not empty, a particular set of upgrade configuration is selected in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X."; col. 5 lines 16-18, "The reconfiguration manager or other server associated therewith then downloads the upgrade to device X [receive] in step 120.").

alSafadi does not explicitly disclose:

- **memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters.**

However, Kidder discloses:

- **memory coupled to a processor and storing a computer program which, when executed by the processor (i) performs physical device recognition on a client device to determine machine parameters** (col. 61 lines 9-13, "Also within modular system services is a Master Control Driver (MCD) that learns the physical characteristics of the particular computer system on which it is running, in this instance, computer system 10."), **and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters** (col. 152 lines 32-42, "… the Master MCD (Master Control Driver) 38 takes a physical inventory of the network device (e.g., computer system 10, FIG. 1, network device 540, FIGS. 35, 59) and assigns a unique physical

identification number (PID) to each physical component within the system, including the network device itself ...").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Kidder into the teaching of alSafadi to include memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters, and (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters. The modification would be obvious because one of ordinary skill in the art would be motivated to provide a physical inventory of the client device so that the parts of the client device can be identified.

alSafadi also does not explicitly disclose:

- **account information for a user of the client device and features of software that the user of the client device is entitled to use;**
- **the software identifier being unique to a particular user of the software; and**
- **if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier.**

However, Baitalmal discloses:

- **account information for a user of a client device and features of software that the user of the client device is entitled to use** (paragraph [0387], "An account is a data structure associated with a user ... where the account data structure includes information regarding which software applications are licensed to the user."; paragraph [0389], "A user account allows a user

(person licensed to use software applications provided by the marketplace) to authenticate to system services. The user account also provides a user with the opportunity to be authorized to access deployed software applications. Users may need to identify themselves for the purposes of accounting, security, logging and resource management. In some embodiments, a user identifies him or herself using an account identifier and a username, and in some embodiments the user also has a password.");

- **a software identifier being unique to a particular user of the software** (paragraph [0582], "… the apply_user table 6806-1 can include foreign keys referring to a data record in the users table 6804-1 associated with a given user (e.g., user_id) and a data record in the applications table associated with a given application (e.g., app_id).")**; and**

- **if the user associated with a device identifier is entitled to use features of an updated program configuration according to a license associated with the software identifier** (paragraph [0258], "User client device ID/type 416 can store information about the type or IDs of computer devices that the user has used to access applications on the application server."; paragraph [0261], "User enabled features 424 includes a list of features on the application server that have been enabled for the user. In some embodiments, user enabled features 424 are determined by the license (e.g., subscription, free, purchased, etc.) granted to the user.")**.**

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Baitalmal into the teaching of alSafadi to include account information for a user of the client device and features of software that the user of the client device is entitled to use; the software identifier being unique to a particular user of

the software; and if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier. The modification would be obvious because one of ordinary skill in the art would be motivated to prevent licensees of the software application from misusing the software application outside the terms of the license (Baitalmal, paragraph [0009]).

12. **Claim 22** is rejected under 35 U.S.C. 103(a) as being unpatentable over alSafadi in view of Baitalmal.

As per Claim 22, alSafadi discloses:

An update server configured to execute a computer program to receive a unique device identifier and a unique software identifier from a client device (col. 4 lines 39-47, "… the reconfiguration manager 10 obtains information regarding the hardware and software configuration of device X, i.e., electronic device 12 of FIG. 1 … In other embodiments, this information may be obtained in another suitable manner, e.g., from a local database based on a serial number or other identifier of the electronic device."**), the update server comprising:**

- **a processor** (Figure 3: 230)**;**
- **memory** (Figure 3: 232) **coupled to the processor and storing the computer program which, when executed by the processor, analyzes the unique device and software identifiers at the update server, and determines based on the analyzed unique device and software identifiers an updated program configuration** (col. 5 lines 8-12, "If step 112 indicates [analyzes] that the set is not empty, a particular set of upgrade configuration is selected

[determined] in step 116, and the upgrade is approved in step 118 as compatible with the current configuration of device X."); **and**

- **a transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein** (col. 5 lines 16-18, "The reconfiguration manager or other server associated therewith then downloads [delivers] the upgrade to device X in step 120.").

alSafadi does not explicitly disclose:

- **if a user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier.**

However, Baitalmal discloses:

- **if a user associated with a device identifier is entitled to use features of an updated program configuration according to a license associated with a software identifier** (paragraph [0258], "User client device ID/type 416 can store information about the type or IDs of computer devices that the user has used to access applications on the application server."; paragraph [0261], "User enabled features 424 includes a list of features on the application server that have been enabled for the user. In some embodiments, user enabled features 424 are determined by the license (e.g., subscription, free, purchased, etc.) granted to the user.").

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate the teaching of Baitalmal into the teaching of alSafadi to include if a user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier. The

modification would be obvious because one of ordinary skill in the art would be motivated to prevent licensees of the software application from misusing the software application outside the terms of the license (Baitalmal, paragraph [0009]).

Response to Arguments

13. Applicant's arguments with respect to Claim 1 have been considered but are moot in view of the new ground(s) of rejection.

In the Remarks, Applicant argues:

a) In response to an upgrade request, when Kidder surveys a client device, MCD 38 assigns a physical identification number (PID) to each internal component of the client device that is being surveyed, so that each individual component can be identified. Using the PIDs, Kidder takes an "inventory" of the client device so that MCD 38 can determine its configuration and thereby determine which cards need an upgraded driver. Kidder at 61:9-41; 62:14-19. There is no teaching to generate, from surveyed machine parameters, a composite device identifier that uniquely identifies the client device. Applicant submits that claim 1 should be allowed for this reason alone.

Examiner's response:

a) Examiner disagrees. With respect to the Applicant's assertion that Kidder does not teach generating, from surveyed machine parameters, a composite device identifier that uniquely identifies the client device, the Examiner respectfully submits that Kidder clearly discloses

"generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters" (col. 152 lines 32-42, "... the Master MCD (Master Control Driver) 38 takes a physical inventory of the network device (e.g., computer system 10, FIG. 1, network device 540, FIGS. 35, 59) and assigns a unique physical identification number (PID) to each physical component within the system, including the network device itself ..."). Note that the MCD takes a physical inventory of the network device and assigns a unique PID to each physical component of the network device and the network device itself.

Therefore, for at least the reason set forth above, the rejections made under 35 U.S.C. § 103(a) with respect to Claims 1-3, 21, and 22 are proper.

Conclusion

14. The prior art made of record and not relied upon is considered pertinent to Applicant's disclosure.

15. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period,

will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

16. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Thursday from 7:30 AM to 4:00 PM. The Examiner can also be reached on alternate Fridays.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached on 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/Anna Deng/

Primary Examiner, Art Unit 2191

Notice of References Cited	Application/Control No. 12/818,906	Applicant(s)/Patent Under Reexamination ETCHEGOYEN, CRAIG STEPHE	
	Examiner QING CHEN	Art Unit 2191	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-2002/0099952 A1	07-2002	Lambert et al.	713/200
*	B	US-2009/0037337 A1	02-2009	Baitalmal et al.	705/59
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2191
Examiner Name	Qing Chen
Attorney Docket Number	UN-NP-AD-037

U. S. PATENT DOCUMENTS

Examiner Initials	Cite No.	Document Number Number-Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		US-6,324,519	11/27/2001	Eldering, Charles A.	
		US-2005/0055269	03/10/2005	Roetter et al.	
		US-2007/0072676	03/29/2007	Baluja, Shumeet	
		US-2008/0167943	07/10/2008	O'Neil et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document Country Code – Number – Kind Code	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		EP 1 096 406	5/2/2001	Madonion Oy		
		WO 2001/090892	11/29/2001	Everdream Inc.		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date page(s), volume-issue number(s), publisher, city and/or country where published.	T
		Microsoft Corporation, "Operations Guide: Microsoft Systems Management Server 2003," 2003, Internet Citation retrieved on June 27, 2007. XP 002439673	
		Rivest, R. "RFC 1321 - The MD5 Message Digest Algorithm," April 1992, Retrieved from the Internet on July 21, 2005.	
		Wikipedia: "Software Extension," May 28, 2009, Internet Article retrieved on October 11, 2010. XP002604710	
		H. Williams, et al., "Web Database Applications with PHP & MySQL", Chapter 1, "Database Applications and the Web", ISBN 0-596-00041-3, O'Reilly & Associates, Inc., March 2002, avail. at: http://docstore.mik.ua/orelly/webprog/webdb/ch01_01.htm. XP002603488	

Examiner Signature	/Qing Chen/	Date Considered	12/28/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	**Art Unit**
	QING CHEN	2191

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/11/2011	12/27/2011							
	1	✓	✓							
	2	✓	✓							
	3	✓	✓							
	4	✓	✓							
	5	✓	✓							
	6	✓	✓							
	7	✓	✓							
	8	✓	✓							
	9	✓	✓							
	10	✓	✓							
	11	✓	✓							
	12	✓	✓							
	13	✓	✓							
	14	✓	✓							
	15	✓	✓							
	16	✓	✓							
	17	✓	✓							
	18	✓	-							
	19	✓	-							
	20	✓	-							
	21		✓							
	22		✓							

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S37	52	CRAIG near ETCHEGOYEN.in.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S38	1	S37 and (remote near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S39	75	("20060072444" | "20080086423" | "20080320607" | "5414269" | "5754763" | "6044471" | "7188241" | "20040030912" | "20060265337" | "20070168288" | "20070219917" | "6233567" | "6294793" | "6330670" | "7203966" | "7327280" | "7337147" | "20010034712" | "20020082997" | "20040187018" | "20090138975" | "5440635" | "5490216" | "5745879" | "6009401" | "6158005" | "6449645" | "6920567" | "7272728" | "7319987" | "0000000" | "20020161718" | "20040143746" | "20070198422" | "20070282615" | "5291598" | "5418854" | "5666415" | "6230199" | "6785825" | "7032110" | "7206765" | "20030065918" | "20030172035" | "20040024860" | "20040059929" | "20060282511" | "20070203846" | "20080065552" | "4658093" | "4796220" | "6243468" | "6976009" | "7069595" | "20020019814" | "20050172280" | "20080147556" | "20080228578" | "5925127" | "7463945" | "4351982" | "4704610" | "5210795" | "6536005" | "6859793" | "7085741" | "7343297" | "7653899" | "20050108173" | "20050138155" | "20060095454" | "20060161914" | "20090083730" | "5790664" | "5974150" | "7069440").PN.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S40	1	S39 and (remote near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S41	2	S39 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S42	2	("6327617" "6880086").pn.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S43	14	uniloc.as.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S44	0	S43 and (remote$2 near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S45	49	("5655152" "6098099" "6839564" "7146389" "7555303" "7610387" "7904622" "20020161769" "20040044698" "20050125459" "20070043793" "20070276962"	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57

		"20070288548" "20090300066" "20100199016" "20100333081" "5983241" "5995999" "6178519" "5528757" "5675752" "5699517" "5745904" "5796966" "5809543" "5809527" "5884301" "5974409" "5987376" "5999947" "6041362" "6044399" "6061799" "6061799" "6094721" "6138120" "6151708" "6157953" "6199762" "6202085" "6253188" "6301612" "6311209" "6317754" "6330560" "6336115" "6442549" "6560636" "6789255" "6954737").pn.				
S46	7271	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S47	7271	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S48	628	S47 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S49	48	S48 and (device near3 identifier)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S50	43	S49 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S51	1	"5155847".pn.	US-PGPUB; USPAT	OR	OFF	2011/12/22 15:57
S52	42	("5155847" | "5253344" | "5327560" | "5497490" | "5634075" | "5822531" | "5898872" | "5918194" | "5933026" | "6058455" | "6065068" | "6167408" | "6301707" | "6385668").PN. OR ("6467088").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2011/12/22 15:57
S53	7	S52 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S54	17913	(remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR;	OR	OFF	2011/12/22 15:57

			EPO; JPO; DERWENT; IBM_TDB			
S55	782	S54 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S56	118	S55 and (hash$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S57	80	S56 and ((updat$4 or upgrad$4) near3 server)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S58	75	S57 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S59	53	S58 and configuration	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S60	42	S58 and (device near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S61	53	S47 and (hardware near3 profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S62	51	S61 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S63	11	S62 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S64	20	S47 and (hardware near3 parameter)	US-PGPUB; USPAT; USOCR;	OR	OFF	2011/12/22 15:57

			EPO; JPO; DERWENT; IBM_TDB			
S65	24	S47 and (machine near3 parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S66	6	S47 and (determin$4 with (machine near3 parameter))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S67	20	(hardware near3 profile) with identifier	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S68	18	S67 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S69	2	(device near identifier) with (hardware near profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S70	0	(device near identifier) with (hardware near parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S71	2	S64 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 15:57
S72	392	S46 and @pd>="20110810"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:00
S73	29	S72 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:00
S74	12	(US-20030014745-$ or US-20050034115-$ or US-20070169087-$ or US-20030195995-$ or US-	US-PGPUB; USPAT	OR	OFF	2011/12/22 16:01

		20050262498-$).did. or (US-6327617-$ or US-6467088-$ or US-5155847-$ or US-7200237-$ or US-7577948-$ or US-6880086-$ or US-7676804-$).did.				
S75	0	S74 and ((software or program) near id)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:01
S77	0	S74 and (user near3 account)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:02
S78	260	S46 and (user near account)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:32
S79	38	S46 and (user near account near information)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:32
S80	75	S78 and permission	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:36
S81	17	S78 and (user near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:36
S82	6	S78 and (user near privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:38
S83	82	S78 and license	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:56
S84	14	S78 and (user near license)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 16:57
S85	5	S46 and ((user near account) with permission)	US-PGPUB; USPAT;	OR	OFF	2011/12/22 17:06

			USOCR; EPO; JPO; DERWENT; IBM_TDB			
S86	2	S46 and ((user near account) with privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:07
S87	2	("20020052796" "6092088").pn.	US-PGPUB; USPAT	OR	OFF	2011/12/22 17:10
S88	232	((user near account) with privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:12
S89	3	S88 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:12
S90	323	((user near account) with permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:12
S91	7	S90 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:12
S92	3	(user near account) with ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:15
S93	143	user with ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:15
S94	5	S93 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:15
S95	38	user with ((software or program) near privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:16

S96	3	S95 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:16
S97	1207	user with ((software or program) near feature)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:17
S98	100	S97 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:17
S99	17	S98 and permission	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:17
S100	30	(user near type) and ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:20
S101	2	S100 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:20
S102	26	(user near account) and ((software or program) near restriction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/22 17:25
S103	142	(user near3 permission) with license	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/27 10:19
S104	7	S103 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/27 10:19
S105	21	(user near3 permission) with (software near3 license)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2011/12/27 10:22

S106	1	"20090070388".pn.	US-PGPUB; USPAT	OR	OFF	2011/12/27 10:28
S107	1	"6880086".pn.	US-PGPUB; USPAT	OR	OFF	2011/12/27 11:31
S108	0	S107 and (software near id)	US-PGPUB; USPAT	OR	OFF	2011/12/27 11:31
S109	1	S107 and (application near id)	US-PGPUB; USPAT	OR	OFF	2011/12/27 11:32
S110	4	("6324519" "20050055269" "20070072676" "20080167943").pn.	US-PGPUB; USPAT	OR	OFF	2011/12/28 10:05

12/ 28/ 2011 10:15:05 AM
C:\ Users\ qchen\ Documents\ EAST\ Workspaces\ docket\ 12818906.wsp

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	Art Unit
	QING CHEN	2191

SEARCHED			
Class	Subclass	Date	Examiner

SEARCH NOTES		
Search Notes	Date	Examiner
Updated EAST Search (US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB)	12/22/2011	/QC/
717/168-178 (limited classification search using keywords)	12/22/2011	/QC/

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner

PTO/SB/31 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

NOTICE OF APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Docket Number (Optional): UN-NP-AD-037

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to "Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450" [37 CFR 1.8(a)]
on _____

Signature _____

Typed or printed name _____

In re Application of: Craig Stephen Etchegoyen

Application Number	Filed
12/818,906	June 18, 2010

For: Remote Update of Computers Based on Physical Device Recognition

Art Unit	Examiner
2191	Qing Chen

Applicant hereby **appeals** to the Board of Patent Appeals and Interferences from the last decision of the examiner.

The fee for this Notice of Appeal is (37 CFR 41.20(b)(1)) $ 620.00

☑ Applicant claims small entity status. See 37 CFR 1.27. Therefore, the fee shown above is reduced by half, and the resulting fee is: $ 310.00

☐ A check in the amount of the fee is enclosed.

☑ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☐ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. _____ .

☐ A petition for an extension of time under 37 CFR 1.136(a) (PTO/SB/22) is enclosed.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

☐ applicant/inventor.

☐ assignee of record of the entire interest.
See 37 CFR 3.71. Statement under 37 CFR 3.73(b) is enclosed.
(Form PTO/SB/96)

☑ attorney or agent of record. Registration number 51,513

☐ attorney or agent acting under 37 CFR 1.34.
Registration number if acting under 37 CFR 1.34. _____

/Sean D. Burdick/
Signature

Sean D. Burdick
Typed or printed name

949-825-5527
Telephone number

March 30, 2012
Date

NOTE: Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required.
Submit multiple forms if more than one signature is required, see below*.

☐ *Total of _____ forms are submitted.

This collection of information is required by 37 CFR 41.31. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11, 1.14 and 41.6. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Electronic Patent Application Fee Transmittal

Application Number:	12818906
Filing Date:	18-Jun-2010
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Filer:	Sean Dylan Burdick/Amanda Ivey
Attorney Docket Number:	UN-NP-AD-037

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Notice of appeal	2401	1	310	310
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**310**

Electronic Acknowledgement Receipt

EFS ID:	12440170
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick/Amanda Ivey
Filer Authorized By:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	30-MAR-2012
Filing Date:	18-JUN-2010
Time Stamp:	17:53:20
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$310
RAM confirmation Number	5720
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Oath or Declaration filed	AD-037_Notice_Appeal.pdf	41828 fda5e1e6f2c185f17010e9e1717fbc6f6ba0f 2078	no	1
Warnings:					
Information:					
2	Fee Worksheet (SB06)	fee-info.pdf	29925 8d18c87c5de4009972ecaf15660972af7a0a 8fca3	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	71753		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. no.:	12/818,906	Conf. no.	8831
Applicant:	Craig S. Etchegoyen et al.	Art Unit:	2191
Filed:	June 18, 2010	Examiner:	Qing Chen

Title: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

AMENDMENT AFTER FINAL ACTION

UNDER 37 CFR § 1.116(b)(2)

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir,

In response to the Final Office Action mailed December 30, 2011, applicant submits this amendment concurrently with an Appeal Brief to present rejected claims in better form for consideration on appeal, wherein:

Amendments to the Claims are shown in the listing of claims beginning on page 2.
Remarks begin on page 8.

IN THE CLAIMS:

1. (currently amended) A system for remotely updating a program configuration, comprising a client device and an update server wherein:

(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of ~~the~~ software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the unique software identifier being unique to a particular copy of the software and to ~~the~~ a particular user of the software; and

a first transceiver configured to send the unique device identifier and the unique software identifier[[s]] to an update server via the Internet; and

(b) the update server is configured to receive the unique device identifier and the unique software identifier[[s]] from the client device, the update server comprising:

a second processor coupled to memory and configured to analyze the unique device identifier and the unique software identifier[[s]] at the update server, and to determine based on the analyzed unique device identifier and the analyzed unique software identifier[[s]] an updated program configuration if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier; and

a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

2. (original) The system of claim 1 wherein the unique device identifier comprises a hash code.

3. (original) The system of claim 1 wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.

4. (original) The system of claim 3 wherein the at least one irreversible transformation comprises a cryptographic hash function.

5. (previously presented) The system of claim 1 wherein the unique device identifier further comprises one or more geo-location codes.

6. (currently amended) The system of claim 5 wherein at least one of the one or more geo-location codes comprises an Internet Protocol address of the client device.

7. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

8. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.

9. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse model, printer model, and scanner model.

10. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.

11. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.

12. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.13.

13. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.

14. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache error correction type.

15. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.

16. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.

17. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.

18-20. (canceled)

21. (currently amended) A client device configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a processor;

APPLE EXHIBIT 1102, Page 982 of 1048

memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for software on the client device, the unique software identifier being unique to a particular copy of the software and to the particular user of the software; and

a transceiver configured to (i) send the unique device identifier and the unique software identifier[[s]] to an update server via the Internet, and (ii) receive from the update server an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier.

22. (canceled)

REMARKS

Claim 22 is canceled herein. Applicant amends claims 1, 6 and 21 as suggested by the Examiner, to overcome the claim objections and to overcome the claim rejections under 35 USC § 112 set forth in the Final Office Action of December 30, 2011. Applicant requests entry of these amendments to present the claims in better form for consideration on appeal.

Applicant respectfully requests entry of this amendment.

Respectfully Submitted,

Sean D. Burdick
Reg. No. 51,513

Uniloc USA, Inc.
2151 Michelson Drive, Suite 100
Irvine, CA 92612
(949) 825-5527

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Before the Board of Patent Appeals and Interferences

In re Application of: Craig S. Etchegoyen USSN: 12/818,906 Filed: June 18, 2010 For: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION	Examiner: Qing Chen Group Art Unit: 2191 Confirmation No. 8831

APPEAL BRIEF

This appeal brief is filed in response to the Final Action mailed December 30, 2011.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST 4

II. RELATED APPEALS AND INTERFERENCES 4

III. STATUS OF CLAIMS 4

IV. STATUS OF AMENDMENTS 5

V. SUMMARY OF CLAIMED SUBJECT MATTER 5

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL 9

VII. ARGUMENT 10

VIII. CLAIMS APPENDIX 17

IX. EVIDENCE APPENDIX 23

X. RELATED PROCEEDINGS APPENDIX 23

TABLE OF AUTHORITIES

1. 35 U.S.C. §103

2. *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991)

3. *KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398 (2007)

4. *In re Kahn*, 441 F.3d 977 (Fed. Cir. 2006)

I. REAL PARTY IN INTEREST

The real party in interest is Uniloc Luxembourg S.A.

II. RELATED APPEALS AND INTERFERENCES

To appellant's best knowledge, there are no related appeals or interferences.

III. STATUS OF CLAIMS

Claims 1-17 and 21 are under final rejection and these rejections are being appealed. Claims 18-20 and 22 are canceled.

Claims 1, 6 and 21 were objected to for informalities. Applicant submitted an Amendment After Final to put these claims in better form for consideration on appeal.

Claims 1-17 and 21 stand rejected under 35 USC §112 for insufficient antecedent basis. Applicant corrected these problems in the Amendment After Final to put the claims in better form for consideration on appeal.

Claims 1-17 and 21 stand rejected under 35 USC §103(a) as obvious over U.S. Patent 6,467,088 ("*alSafadi*") in view of U.S. Patent 6,880,086 ("*Kidder*") and U.S. Patent Application Publication 2009/0037337 ("*Baitalmal*").

IV. STATUS OF AMENDMENTS

An amendment was filed on May 30, 2012 under 37 CFR § 1.116(b)(2) to present claims 1, 6 and 21 in better form for consideration on appeal. The status of the amendment is unknown to the applicant at the time of filing this appeal brief.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention ("*Etchegoyen*") is directed to remote updating of software. *Etchegoyen* provides, in a client-server system, a specialized software program stored on the client device that generates a unique device identifier for the client device, which is derived from multiple machine parameters readable on the client device. The unique device identifier when transmitted to the server along with a unique software identifier allows the server to determine, among other things, whether the client device is *licensed* to receive an upgrade for application software identified by the unique software identifier.

In the embodiment recited in claim 1, the invention provides a system for remotely updating a program configuration, comprising a client device and an update server wherein:

(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

APPLE EXHIBIT 1102, Page 989 of 1048

a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the unique software identifier being unique to a particular copy of the software and to a particular user of the software; and

a first transceiver configured to send the unique device identifier and the unique software identifier to an update server via the Internet; and

(b) the update server is configured to receive the unique device identifier and the unique software identifier from the client device, the update server comprising:

a second processor coupled to memory and configured to analyze the unique device identifier and the unique software identifier at the update server, and to determine based on the analyzed unique device identifier and the analyzed unique software identifier an updated program configuration if the user associated with the device identifier is entitled to use

APPLE EXHIBIT 1102, Page 990 of 1048

features of the updated program configuration according to a license associated with the software identifier; and

a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

The embodiment in claim 21 recites a client computing device equipped with remote updating software according to the invention. The client device is configured to execute a computer program to perform a remote update of a program configuration on the client device, wherein the client device comprises:

a processor;

memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for software on the client device, the unique software identifier being unique to a particular copy of the software and to the particular user of the software; and

APPLE EXHIBIT 1102, Page 991 of 1048

a transceiver configured to (i) send the unique device identifier and the unique software identifier to an update server via the Internet, and (ii) receive from the update server an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier.

The embodiment of claim 22 expresses the invention from the perspective of the server. An update server is configured to execute a computer program to receive a unique device identifier and a unique software identifier from a client device, wherein the update server comprises:

a processor;

memory coupled to the processor and storing the computer program which, when executed by the processor, analyzes the unique device identifier and the unique software identifier at the update server, and determines based on the analyzed unique device identifier and the analyzed unique software identifier an updated program configuration if a user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier; and

APPLE EXHIBIT 1102, Page 992 of 1048

a transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Pursuant to applicant's Notice of Appeal filed on March 30, 2012, and pursuant to 37 CFR §41.31, applicant appeals from the Final Action of December 30, 2011, and requests review of the following grounds of rejection:

1. Whether claims 1-17 and 21 are unpatentable under 35 USC §103(a) over U.S. Patent 6,467,088 ("*alSafadi*") in view of U.S. Patent 6,880,086 ("*Kidder*") and U.S. Patent Application Publication 2009/0037337 ("*Baitalmal*").

VII. ARGUMENT

A. Introduction

The *Etchegoyen* invention is directed to remote updating of software. *Etchegoyen* provides, in a client-server system, a specialized software program stored on the client device that generates a unique device identifier for the client device, which is derived from multiple machine parameters readable on the client device. The unique device identifier when transmitted to the server along with a unique software identifier allows the server to determine, among other things, whether the client device is *licensed* to receive an upgrade for application software identified by the unique software identifier.

AlSafadi generally teaches a reconfiguration manager (or upgrade server) that receives an upgrade request from a client device. The upgrade request contains (i) identification of a desired software upgrade, and (ii) a list of hardware and software components presently installed on the client device. The reconfiguration server compares the list of components to known "good" configurations that will support the requested upgrade. If the list of components satisfies a good configuration, then the upgraded version of the software is downloaded to the client. If the list of components does not

APPLE EXHIBIT 1102, Page 994 of 1048

satisfy a good configuration, then the reconfiguration manager either denies the request, or informs the client that the requested upgrade is unknown, or gives the client an option to download all components needed to implement the desired upgrade. *AlSafadi* at 4:12 to 5:40.

Importantly, *AlSafadi* unlike *Etchegoyen* doesn't teach generating from the machine parameters of the client device, a unique device identifier for the client device. In *AlSafadi*, the client computer's software and hardware component identifiers are predetermined, and are either included with the upgrade request, or they are obtained by cross-referencing a serial number of the client device to configuration data stored in a database. *Id.* at 4:40-47. There is no test in *AlSafadi* to determine whether the client is licensed to receive the upgrade.

Kidder teaches the use of software "signatures" to promote hot upgrades of software components, whereby an early release of a software product can be upgraded to a later release. *Kidder's* method involves generating a signature for each component of a first release of a software product, and when an upgrade request is received from a client device, comparing each signature to the signatures in a second release of the product. During operation of the product, each signature that matches continues to run and is not upgraded, and each signature that doesn't

APPLE EXHIBIT 1102, Page 995 of 1048

match is ignored in favor of its corresponding upgraded component. *Kidder* at 3:49-65.

Importantly, *Kidder's* method does not generate a composite device identifier for the client device. *Kidder* generates individual signatures for each component in the client device that it surveys, to facilitate a component-by-component comparison. There is no test in *Kidder* to determine whether the client is licensed to receive the upgrade.

Baitalmal teaches a software licensing and enforcement system managed by a marketplace server that distributes software from various vendors to users. *Baitalmal* is cited in the Final Office Action for teachings relative to maintaining user accounts, authenticating users based on usernames and passwords, and for associating user-enabled features with a software license. Importantly, *Baitalmal* provides no teachings related to providing software on a client device that generates a composite device identifier for use in linking the client device to a software license.

Etchegoyen, on the other hand, teaches a security-related method for upgrading software. The *Etchegoyen* invention collects machine parameters and other identifiers from the client device to determine its identity and thus whether it is licensed to receive a software upgrade. None of *AlSafadi*, *Kidder* and *Baitalmal* is concerned with collecting machine

APPLE EXHIBIT 1102, Page 996 of 1048

parameters for the purpose of uniquely identifying the client device, to determine whether a software upgrade for that particular client device is authorized.

B. Legal standard for obviousness under §103

Well-established patent law holds that an obviousness rejection cannot be sustained unless the cited reference(s) (a) provide a suggestion or motivation to combine reference teachings in the manner claimed; (b) provide a reasonable expectation of success; and (c) teach all of the claim limitations, except for those limitations already within the knowledge or common sense of a person of ordinary skill in the art. *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991); *KSR Int'l Co. v. Teleflex, Inc.*, 550 U.S. 398 (2007). In addition, the burden is on the examiner to clearly articulate the reason(s) why the claimed invention would have been obvious. "[R]ejections on obviousness cannot be sustained with mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." *KSR Int'l v. Teleflex Inc.* 550 U.S. 398, 418 (2007) (quoting *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006)).

C. None of the cited references teach the Claim 1 step for generating a unique device identifier from machine parameters.

The Final Office Action on page 6 admits that *AlSafadi* doesn't teach the first processor element as claimed, but cites

APPLE EXHIBIT 1102, Page 997 of 1048

to *Kidder* as teaching the step for generating a unique device identifier based on the functionality of *Kidder's* Master Control Driver (MCD) 38. *Kidder's* MCD 38, however, doesn't generate a unique device identifier from machine parameters.

In response to an upgrade request, when *Kidder* surveys a client device, MCD 38 assigns a physical identification number (PID) to each internal component of the client device that is being surveyed, so that each *individual* component can be identified. Using the PIDs, *Kidder* takes an "inventory" of the client device so that MCD 38 can determine its configuration and thereby determine which cards need an upgraded driver. *Kidder* at 61:9-41; 62:14-19. There is no teaching to generate, from surveyed machine parameters, a composite device identifier that uniquely identifies the client device.

Baitalmal provides no such teachings, either, and was not cited for this purpose.

Therefore none of the references teach the element of claim 1 wherein "a first processor … (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters…."

For failure of the cited references to teach or suggest all elements of claim 1, the obviousness rejection should be reversed.

APPLE EXHIBIT 1102, Page 998 of 1048

Based on dependency from claim 1, the obviousness rejections of claims 1-17 should also be reversed.

D. The rejection of claim 2 is not supported by *Kidder*

Claim 2 should be allowed over *AlSafadi* in view of *Kidder* and *Baitalmal* for reciting "wherein the unique identifier comprises a hash code". In rejecting claim 2, the Final Office Action cites to *Kidder's* use of the Sha-1 utility at 88:62-64. But the hashed signature as taught in *Kidder* comprises a SHA-1 hash for each of the software components present in the device requesting an upgrade. Nowhere does *Kidder* teach or suggest hashing a unique identifier that is generated from multiple machine parameters.

E. The rejection of Claim 3 is not supported by *Kidder*

Claim 3 should be allowed over *AlSafadi* in view of *Kidder* and *Baitalmal* for similar reasons. The Final Office Action interprets the Sha-1 utility as used in *Kidder* as anticipating the "irreversible transformation" feature recited in claim 3. But again, there is no teaching in *Kidder* to derive a unique device identifier from multiple machine parameters. Therefore, the irreversible transformation of Sha-1 as proposed in *Kidder* is not implemented on a "device identifier" as that term is defined in the present specification.

F. Claim 21 should be allowed for the same reason as Claim 1

The rejection of claim 21 should be reversed for the same reason in favor of allowing claim 1. Claim 21 also recites the "processor" element wherein the processor "generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters".

Applicant reasserts here the arguments presented with respect to claim 1. *AlSafadi* in view of *Kidder* and *Baitalmal* cannot provide the basis for an obviousness rejection of claim 21 without teaching all elements recited in the claim.

F. Conclusion

Applicant respectfully submits that the rejections in this application are improper and should be overturned.

Respectfully submitted,

Dated: May 30, 2012

Sean D. Burdick
Registration No. 51,513
Attorney for Appellant
2151 Michelson Drive
Irvine, CA 92612
Phone: 949-825-5527
Fax: 949-788-1471
sean.burdick@unilocusa.com

APPLE EXHIBIT 1102, Page 1000 of 1048

VIII. CLAIMS APPENDIX

1. (currently amended) A system for remotely updating a program configuration, comprising a client device and an update server wherein:

(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a first processor coupled to memory storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of ~~the~~ software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the unique software identifier being unique to a particular copy of the software and to ~~the~~ a particular user of the software; and

a first transceiver configured to send the unique device identifier and the unique software identifier[[s]] to an update server via the Internet; and

APPLE EXHIBIT 1102, Page 1001 of 1048

(b) the update server is configured to receive the unique device identifier and the unique software identifier[[s]] from the client device, the update server comprising:

a second processor coupled to memory and configured to analyze the unique device identifier and the unique software identifier[[s]] at the update server, and to determine based on the analyzed unique device identifier and the analyzed unique software identifier[[s]] an updated program configuration if the user associated with the device identifier is entitled to use features of the updated program configuration according to a license associated with the software identifier; and

a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

2. (original) The system of claim 1 wherein the unique device identifier comprises a hash code.

3. (original) The system of claim 1 wherein the computer program when executed implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.

4. (original) The system of claim 3 wherein the at least one irreversible transformation comprises a cryptographic hash

APPLE EXHIBIT 1102, Page 1002 of 1048

function.

5. (previously presented) The system of claim 1 wherein the unique device identifier further comprises one or more geo-location codes.

6. (currently amended) The system of claim 5 wherein at least one of the one or more geo-location codes comprises an Internet Protocol address of the client device.

7. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: a machine model number, a machine serial number, a machine ROM version, a machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

8. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: CPU ID, CPU model, CPU details, CPU actual speed, CPU family, CPU manufacturer name, CPU voltage, and CPU external clock.

9. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: optical model, optical serial number, keyboard model, mouse

APPLE EXHIBIT 1102, Page 1003 of 1048

model, printer model, and scanner model.

10. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: baseboard manufacturer, baseboard product name, baseboard version, baseboard serial number, and baseboard asset tag.

11. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: chassis manufacturer, chassis type, chassis version, and chassis serial number.

12. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.13.

13. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: port connector designator, port connector type, port connector port type, and system slot type.

14. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: cache level, cache size, cache max size, cache SRAM type, and cache

APPLE EXHIBIT 1102, Page 1004 of 1048

error correction type.

15. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: fan, PCMCIA, modem, portable battery, tape drive, USB controller, and USB hub.

16. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: device model, device model IMEI, device model IMSI, and device model LCD.

17. (original) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: wireless 802.11, webcam, game controller, silicone serial, and PCI controller.

18-20. (canceled)

21. (currently amended) A client device configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a processor;

APPLE EXHIBIT 1102, Page 1005 of 1048

memory coupled to the processor and storing the computer program which, when executed by the processor (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for software on the client device, the unique software identifier being unique to a particular copy of the software and to the particular user of the software; and

a transceiver configured to (i) send the unique device identifier and the unique software identifier[[s]] to an update server via the Internet, and (ii) receive from the update server an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier.

22. (canceled)

APPLE EXHIBIT 1102, Page 1006 of 1048

IX. EVIDENCE APPENDIX

No new evidence is presented in this appeal brief.

X. RELATED PROCEEDINGS APPENDIX

To appellant's best knowledge, there are no related proceedings.

APPLE EXHIBIT 1102, Page 1007 of 1048

Electronic Patent Application Fee Transmittal

Application Number:	12818906
Filing Date:	18-Jun-2010
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Filer:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Filing a brief in support of an appeal	2402	1	310	310
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**310**

Electronic Acknowledgement Receipt

EFS ID:	12897780
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	Remote Update of Computers Based on Physical Device Recognition
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick/Amanda Ivey
Filer Authorized By:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	30-MAY-2012
Filing Date:	18-JUN-2010
Time Stamp:	23:39:04
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$310
RAM confirmation Number	14370
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Amendment After Final	Amendment_After_Final_12818906.pdf	47447 84e36f9c5edc33a6be1d88d9f261da09be2c3c16	no	7
Warnings:					
Information:					
2	Appeal Brief Filed	appeal_brief_12818906.pdf	84309 c2a685c884c394bbbfc80600cc719d0a630d9324a	no	23
Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	29989 360dcf5e3b5e3f9d07e0da9a65d7f7e13dd8fcfa	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	161745		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111

If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371

If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office

If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/06 (07-06)
Approved for use through 1/31/2007. OMB 0651-0032
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number	Filing Date	☐ To be Mailed
12/818,906	06/18/2010	

APPLICATION AS FILED – PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY ☒ RATE ($)	SMALL ENTITY ☒ FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =		OR	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =			X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $250 ($125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	

APPLICATION AS AMENDED – PART II

AMENDMENT 05/30/2012	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE ($)	SMALL ENTITY ADDITIONAL FEE ($)	OR	OTHER THAN SMALL ENTITY RATE ($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	* 18	Minus	** 20	= 0	X $30 =	0	OR	X $ =	
Independent (37 CFR 1.16(h))	* 2	Minus	*** 3	= 0	X $125 =	0	OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE	0	OR	TOTAL ADD'L FEE	

AMENDMENT	(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)	OR	RATE ($)	ADDITIONAL FEE ($)
Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =		OR	X $ =	
Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =		OR	X $ =	
☐ Application Size Fee (37 CFR 1.16(s))									
☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
					TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.

** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".

*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".

The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

Legal Instrument Examiner:
/KIM WATSON/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

96051 7590 06/28/2012

Uniloc USA Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

EXAMINER
CHEN, QING

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 06/28/2012

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

TITLE OF INVENTION: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$870	$300	$0	$1170	09/28/2012

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status:

A. If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status above is to be removed, check box 5b on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

B. If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check box 5a on Part B - Fee(s) Transmittal and pay the PUBLICATION FEE (if required) and 1/2 the ISSUE FEE shown above.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

96051 7590 06/28/2012

Uniloc USA Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)

(Signature)

(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

TITLE OF INVENTION: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

APPLN. TYPE	SMALL ENTITY	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$870	$300	$0	$1170	09/28/2012

EXAMINER	ART UNIT	CLASS-SUBCLASS
CHEN, QING	2191	717-168000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list

(1) the names of up to 3 registered patent attorneys or agents OR, alternatively, 1

(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed. 2 _____ 3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☐ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:

☐ Issue Fee

☐ Publication Fee (No small entity discount permitted)

☐ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**

☐ A check is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director is hereby authorized to charge the required fee(s), any deficiency, or credit any overpayment, to Deposit Account Number (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)

☐ a. Applicant claims SMALL ENTITY statns. See 37 CFR 1.27.

☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

96051 7590 06/28/2012

Uniloc USA Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

EXAMINER	
CHEN, QING	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 06/28/2012

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment to date is 0 day(s). If the issue fee is paid on the date that is three months after the mailing date of this notice and the patent issues on the Tuesday before the date that is 28 weeks (six and a half months) after the mailing date of this notice, the Patent Term Adjustment will be 0 day(s).

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No.	Applicant(s)	
	12/818,906	ETCHEGOYEN, CRAIG STEPHEN	
	Examiner	Art Unit	
	QING CHEN	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *the amendment filed on May 30, 2012*.

2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

3. ☒ The allowed claim(s) is/are *1-17 and 21, renumbered as 1-18*.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

6. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

7. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

DETAILED ACTION

1. This Office action is in response to the amendment filed on May 30, 2012.

2. **Claims 1-17 and 21** are pending.

3. **Claims 1, 3, 6, 7, 12, and 21** have been amended.

4. **Claims 18-20 and 22** have been canceled.

5. **Claims 1-17 and 21** are allowed, renumbered as 1-18.

6. The objections to Claims 1, 6, 21, and 22 are withdrawn in view of Applicant's amendments to the claims or cancellation of the claims.

7. The 35 U.S.C. § 112, second paragraph, rejections of Claims 1-17 and 21 are withdrawn in view of Applicant's amendments to the claims.

Examiner's Amendment

8. An Examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to Applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this Examiner's amendment was given in a telephone interview with Sean D. Burdick (Reg. No. 51,513) on June 8, 2012.

The application has been amended as follows:

AMENDMENTS TO THE CLAIMS

In the "Amendments to the Claims" (received on 05/30/2012), please amend Claims 1, 3, 7, 12, and 21 as follows:

1. (Currently Amended) A system for remotely updating a program configuration, comprising a client device and an update server wherein:

(a) the client device is configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a first processor coupled to a memory storing the computer program which, when executed by the first processor, (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier is generated based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the unique software identifier being unique to a particular copy of the software and to a particular user of the software; and

a first transceiver configured to send the unique device identifier and the unique software identifier to [[an]] the update server via the Internet; and

(b) the update server is configured to receive the unique device identifier and the unique software identifier from the client device, the update server comprising:

a second processor coupled to a memory and configured to analyze the unique device identifier and the unique software identifier at the update server, and to determine,

based on the analyzed unique device identifier and the analyzed unique software identifier, an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier; and

a second transceiver configured to deliver, via the Internet, data representing the updated program configuration to the client device for storage therein.

3. (Currently Amended) The system of claim 1 wherein the computer program, when executed, implements at least one irreversible transformation such that the machine parameters cannot be derived from the unique device identifier.

7. (Currently Amended) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: [[a]] machine model number, [[a]] machine serial number, [[a]] machine ROM version, [[a]] machine bus speed, machine manufacturer name, machine ROM release date, machine ROM size, machine UUID, and machine service tag.

12. (Currently Amended) The system of claim 1 wherein the machine parameters comprise information regarding at least one of: IDE controller, SATA controller, RAID controller, and SCSI controller.[[13.]]

21. (Currently Amended) A client device configured to execute a computer program to perform a remote update of a program configuration on the client device, the client device comprising:

a processor;

a memory coupled to the processor and storing the computer program which, when executed by the processor, (i) performs physical device recognition on the client device to determine machine parameters including account information for a user of the client device and features of software that the user of the client device is entitled to use, (ii) generates a unique device identifier for the client device, the unique device identifier is generated based at least in part on the determined machine parameters, and (iii) collects a unique software identifier for the software on the client device, the unique software identifier being unique to a particular copy of the software and to [[the]] a particular user of the software; and

a transceiver configured to (i) send the unique device identifier and the unique software identifier to an update server via the Internet to determine, based on analyzing the unique device identifier and the unique software identifier, an updated program configuration, and (ii) receive, from the update server, [[an]] the updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier.

-- END OF AMENDMENT --

Reasons for Allowance

9. The following is an Examiner's statement of reasons for allowance:

The cited prior art taken alone or in combination fail to teach, in combination with the other claimed limitations, "a first processor coupled to a memory storing the computer program which, when executed by the first processor, (ii) generates a unique device identifier for the client device, the unique device identifier is generated based at least in part on the determined machine parameters; a second processor coupled to a memory and configured to analyze the unique device identifier and the unique software identifier at the update server, and to determine, based on the analyzed unique device identifier and the analyzed unique software identifier, an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier" as recited in independent Claim 1; and further fail to teach, in combination with the other claimed limitations, similarly-worded limitations as recited in independent Claim 21.

The closest cited prior art, the combination of US 6,467,088 (hereinafter "alSafadi"), US 6,880,086 (hereinafter "Kidder"), and US 2009/0037337 (hereinafter "Baitalmal"), teaches techniques for upgrading or otherwise reconfiguring software and/or hardware components in electronic devices. However, the combination of alSafadi, Kidder, and Baitalmal fails to teach "a first processor coupled to a memory storing the computer program which, when executed by the first processor, (ii) generates a unique device identifier for the client device, the unique device identifier is generated based at least in part on the determined machine parameters; a second processor coupled to a memory and configured to analyze the unique device identifier and the

unique software identifier at the update server, and to determine, based on the analyzed unique device identifier and the analyzed unique software identifier, an updated program configuration if the user associated with the unique device identifier is entitled to use features of the updated program configuration according to a license associated with the unique software identifier" as recited in independent Claim 1; and further fails to teach similarly-worded limitations as recited in independent Claim 21.

Any comments considered necessary by Applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

10. Any inquiry concerning this communication or earlier communications from the Examiner should be directed to Qing Chen whose telephone number is 571-270-1071. The Examiner can normally be reached on Monday through Friday from 9:30 AM to 5:30 PM.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Wei Zhen, can be reached at 571-272-3708. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Any inquiry of a general nature or relating to the status of this application or proceeding should be directed to the TC 2100 Group receptionist whose telephone number is 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Q. C./

Examiner, Art Unit 2191

/WEI ZHEN/

Supervisory Patent Examiner, Art Unit 2191

Issue Classification

Application/Control No.	Applicant(s)/Patent Under Reexamination
12818906	ETCHEGOYEN, CRAIG STEPHEN
Examiner	**Art Unit**
QING CHEN	2191

ORIGINAL		INTERNATIONAL CLASSIFICATION	
CLASS	SUBCLASS	CLAIMED	NON-CLAIMED
717	172	G 0 6 F 9 / 44 (2006.01.01)	

CROSS REFERENCE(S)					
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)				
717	168	171			

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	17	17												
2	2		18												
3	3		19												
4	4		20												
5	5	18	21												
6	6		22												
7	7														
8	8														
9	9														
10	10														
11	11														
12	12														
13	13														
14	14														
15	15														
16	16														

/QING CHEN/ Examiner.Art Unit 2191 (Assistant Examiner)	06/08/2012 (Date)	**Total Claims Allowed:** 18
/WEI ZHEN/ Supervisory Patent Examiner, Art Unit 2191 (Primary Examiner)	(Date)	O.G. Print Claim(s): 1 O.G. Print Figure: 1

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	Art Unit
	QING CHEN	2191

SEARCHED			
Class	Subclass	Date	Examiner
717	168-178	6/8/2012	/QC/

SEARCH NOTES		
Search Notes	Date	Examiner
Updated EAST Search (US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB) -- See search history printout(s)	6/8/2012	/QC/
717/168-178 (limited classification search using keywords) -- See search history printout(s)	6/8/2012	/QC/
Updated EAST Inventor Name Search and Assignee Search (US-PGPUB; USPAT) -- See search history printout(s)	6/8/2012	/QC/
Updated PALM Inventor Name Search	6/8/2012	/QC/
Consulted SPE Wei Zhen (AU 2191) regarding allowable subject matter	6/4/2012	/QC/

INTERFERENCE SEARCH			
Class	Subclass	Date	Examiner
717	168-178	6/8/2012	/QC/

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	54	CRAIG near ETCHEGOYEN.in.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L2	1	L1 and (remote near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L3	75	("20060072444" | "20080086423" | "20080320607" | "5414269" | "5754763" | "6044471" | "7188241" | "20040030912" | "20060265337" | "20070168288" | "20070219917" | "6233567" | "6294793" | "6330670" | "7203966" | "7327280" | "7337147" | "20010034712" | "20020082997" | "20040187018" | "20090138975" | "5440635" | "5490216" | "5745879" | "6009401" | "6158005" | "6449645" | "6920567" | "7272728" | "7319987" | "0000000" | "20020161718" | "20040143746" | "20070198422" | "20070282615" | "5291598" | "5418854" | "5666415" | "6230199" | "6785825" | "7032110" | "7206765" | "20030065918" | "20030172035" | "20040024860" | "20040059929" | "20060282511" | "20070203846" | "20080065552" | "4658093" | "4796220" | "6243468" | "6976009" | "7069595" | "20020019814" | "20050172280" | "20080147556" | "20080228578" | "5925127" | "7463945" | "4351982" | "4704610" | "5210795" | "6536005" | "6859793" | "7085741" | "7343297" | "7653899" | "20050108173" | "20050138155" | "20060095454" | "20060161914" | "20090083730" | "5790664" | "5974150" | "7069440").PN.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L4	1	L3 and (remote near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L5	2	L3 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L6	2	("6327617" "6880086").pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L7	17	uniloc.as.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L8	0	L7 and (remote$2 near3 (updat$4 or upgrad$4)).clm.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L9	49	("5655152" "6098099" "6839564" "7146389" "7555303" "7610387" "7904622" "20020161769" "20040044698" "20050125459" "20070043793" "20070276962" "20070288548" "20090300066" "20100199016" "20100333081"	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07

		"5983241" "5995999" "6178519" "5528757" "5675752" "5699517" "5745904" "5796966" "5809543" "5809527" "5884301" "5974409" "5987376" "5999947" "6041362" "6044399" "6061799" "6061799" "6094721" "6138120" "6151708" "6157953" "6199762" "6202085" "6253188" "6301612" "6311209" "6317754" "6330560" "6336115" "6442549" "6560636" "6789255" "6954737").pn.				
L10	7750	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L11	7750	717/168-178.ccls.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L12	653	L11 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L13	50	L12 and (device near3 identifier)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L14	44	L13 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L15	1	"5155847".pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L16	43	("5155847" | "5253344" | "5327560" | "5497490" | "5634075" | "5822531" | "5898872" | "5918194" | "5933026" | "6058455" | "6065068" | "6167408" | "6301707" | "6385668").PN. OR ("6467088").URPN.	US-PGPUB; USPAT; USOCR	OR	OFF	2012/06/08 15:07
L17	7	L16 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L18	18964	(remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT;	OR	OFF	2012/06/08 15:07

			IBM_TDB			
L19	819	L18 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L20	128	L19 and (hash$4)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L21	87	L20 and ((updat$4 or upgrad$4) near3 server)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L22	78	L21 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L23	55	L22 and configuration	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L24	43	L22 and (device near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L25	58	L11 and (hardware near3 profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L26	54	L25 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L27	12	L26 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L28	20	L11 and (hardware near3 parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT;	OR	OFF	2012/06/08 15:07

			IBM_TDB			
L29	24	L11 and (machine near3 parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L30	6	L11 and (determin$4 with (machine near3 parameter))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L31	22	(hardware near3 profile) with identifier	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L32	19	L31 and (@pd<"20090624" or @ad<"20090624" or @prad<"20090624" or @rlad<"20090624")	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L33	2	(device near identifier) with (hardware near profile)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L34	0	(device near identifier) with (hardware near parameter)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L35	2	L28 and hash$4	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L36	870	L10 and @pd>="20110810"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L37	54	L36 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L38	12	(US-20030014745-$ or US-20050034115-$ or US-20070169087-$ or US-20030195995-$ or US-20050262498-$).did. or (US-6327617-$ or US-6467088-$ or US-5155847-$ or US-7200237-$ or	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07

		US-7577948-$ or US-6880086-$ or US-7676804-$).did.				
L39	0	L38 and ((software or program) near id)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L40	0	L38 and (user near3 account)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L41	287	L10 and (user near account)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L42	40	L10 and (user near account near information)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L43	79	L41 and permission	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L44	17	L41 and (user near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L45	6	L41 and (user near privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L46	88	L41 and license	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L47	16	L41 and (user near license)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L48	5	L10 and ((user near account) with permission)	US-PGPUB; USPAT; USOCR; EPO; JPO;	OR	OFF	2012/06/08 15:07

			DERWENT; IBM_TDB			
L49	4	L10 and ((user near account) with privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L50	2	("20020052796" "6092088").pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:07
L51	247	((user near account) with privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L52	5	L51 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L53	373	((user near account) with permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L54	7	L53 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L55	3	(user near account) with ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L56	148	user with ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L57	5	L56 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L58	41	user with ((software or program) near privilege)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:07
L59	3	L58 and "717".clas.	US-PGPUB; USPAT;	OR	OFF	2012/06/08 15:08

			USOCR; EPO; JPO; DERWENT; IBM_TDB			
L60	1266	user with ((software or program) near feature)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L61	102	L60 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L62	17	L61 and permission	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L63	31	(user near type) and ((software or program) near permission)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L64	2	L63 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L65	27	(user near account) and ((software or program) near restriction)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L66	155	(user near3 permission) with license	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L67	7	L66 and "717".clas.	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L68	25	(user near3 permission) with (software near3 license)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:08
L69	1	"20090070388".pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:08

L70	1	"6880086".pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:08
L71	0	L70 and (software near id)	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:08
L72	1	L70 and (application near id)	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:08
L73	4	("6324519" "20050055269" "20070072676" "20080167943").pn.	US-PGPUB; USPAT	OR	OFF	2012/06/08 15:08
L74	482	10 and @pd>="20111222"	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:10
L75	25	74 and (device near3 identifier)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:10
L76	0	75 and (software near identifier)	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:11
L77	2	75 and (remote$2 near3 (updat$4 or upgrad$4))	US-PGPUB; USPAT; USOCR; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2012/06/08 15:11

6/ 8/ 2012 3:55:56 PM
C:\ Users\ qchen\ Documents\ EAST\ Workspaces\ docket\ 12818906.wsp

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. no.:	12/818,906	Conf. no.	8831
Applicant:	Craig S. Etchegoyen et al.	Art Unit:	2191
Filed:	June 18, 2010	Examiner:	Qing Chen

Title: REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION

OK TO ENTER /QC/ 06/08/2012

AMENDMENT AFTER FINAL ACTION

UNDER 37 CFR § 1.116(b)(2)

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir,

In response to the Final Office Action mailed December 30, 2011, applicant submits this amendment concurrently with an Appeal Brief to present rejected claims in better form for consideration on appeal, wherein:

Amendments to the Claims are shown in the listing of claims beginning on page 2.

Remarks begin on page 8.

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12818906	ETCHEGOYEN, CRAIG STEPHEN
	Examiner	**Art Unit**
	QING CHEN	2191

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

CLAIM		DATE								
Final	Original	08/11/2011	12/27/2011	06/08/2012						
1	1	✓	✓	=						
2	2	✓	✓	=						
3	3	✓	✓	=						
4	4	✓	✓	=						
5	5	✓	✓	=						
6	6	✓	✓	=						
7	7	✓	✓	=						
8	8	✓	✓	=						
9	9	✓	✓	=						
10	10	✓	✓	=						
11	11	✓	✓	=						
12	12	✓	✓	=						
13	13	✓	✓	=						
14	14	✓	✓	=						
15	15	✓	✓	=						
16	16	✓	✓	=						
17	17	✓	✓	=						
	18	✓	-	-						
	19	✓	-	-						
	20	✓	-	-						
18	21		✓	=						
	22		✓	-						

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Mail Stop ISSUE FEE
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571) 273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

Uniloc USA, Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	06/18/2010	Craig Stephen Etchegoyen	UN-NP-AD-037	8831

TITLE OF INVENTION:

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$870	$300	$1170	09/28/2012

EXAMINER	ART UNIT	CLASS-SUBCLASS

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) the names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 Sean D. Burdick
2
3

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: UNILOC LUXEMBOURG S.A.

(B) RESIDENCE: (CITY and STATE OR COUNTRY): 15, Rue Edward Steichen, Luxembourg City, L-2450, Grand-Duchy of Luxembourg

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☑ Corporation or other private group entity ☐ Government

4a. The following fee(s) are enclosed:
☑ Issue Fee
☑ Publication Fee (No small entity discount permitted)
☐ Advance Order - # of Copies _

4b. Payment of Fee(s):
☐ A check in the amount of the fee(s) is enclosed.
☑ Payment by credit card. Form PTO-2038 is attached.
☐ The Director is hereby authorized to charge the required fee(s), or credit any overpayment, to Deposit Account Number

5. **Change in Entity Status** (from status indicated above)
☑ a. Applicant claims SMALL ENTITY status. See 37 CFR 1.27.
☐ b. Applicant is no longer claiming SMALL ENTITY status. See 37 CFR 1.27(g)(2).

The Director of the USPTO is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.
NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

Authorized Signature /Sean D. Burdick/ Date 06/28/2012

Typed or printed name Sean D. Burdick Registration No. 51513

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Electronic Patent Application Fee Transmittal

Application Number:	12818906
Filing Date:	18-Jun-2010
Title of Invention:	REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Filer:	Sean Dylan Burdick
Attorney Docket Number:	UN-NP-AD-037

Filed as Small Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl issue fee	2501	1	870	870
Publ. Fee- early, voluntary, or normal	1504	1	300	300

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension-of-Time:				
Miscellaneous:				
			Total in USD ($)	**1170**

Electronic Acknowledgement Receipt

EFS ID:	13133083
Application Number:	12818906
International Application Number:	
Confirmation Number:	8831
Title of Invention:	REMOTE UPDATE OF COMPUTERS BASED ON PHYSICAL DEVICE RECOGNITION
First Named Inventor/Applicant Name:	Craig Stephen Etchegoyen
Customer Number:	96051
Filer:	Sean Dylan Burdick
Filer Authorized By:	
Attorney Docket Number:	UN-NP-AD-037
Receipt Date:	28-JUN-2012
Filing Date:	18-JUN-2010
Time Stamp:	17:42:59
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$1170
RAM confirmation Number	5785
Deposit Account	
Authorized User	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)

1	Issue Fee Payment (PTO-85B)	ptol85b.pdf	229970 220087f030c2f28dabb9939c11b5021f68a1f56d6	no	2
Warnings:					
Information:					
2	Fee Worksheet (SB06)	fee-info.pdf	32145 88078171de1620af5b1ad39b5f027efa79a400f6	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	262115		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/SB/08a (07-09)
Approved for use through 07/31/2010. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO
(modified by Applicant)

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet 5 of 5

Complete if Known	
Application Number	12/818,906
Filing Date	June 18, 2010
First Named Inventor	Craig S. Etchegoyen
Art Unit	2192
Examiner Name	
Attorney Docket Number	UN-NP-AD-037

FOREIGN PATENT DOCUMENTS

Examiner Initials	Cite No.	Foreign Patent Document (Country Code – Number – Kind Code)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T
		WO 2009158525	12/30/2009	Uniloc USA, Inc.		

NON PATENT LITERATURE DOCUMENTS

Examiner Initials	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date page(s), volume-issue number(s), publisher, city and/or country where published.	T
		WILLIAMS, R., "A Painless Guide to CRC Error Detection Algorithms", Ver. 3, Aug. 19, 1993.	
		ANGHA, F. et al., "Securing Transportation Network Infrastructure with Patented Technology of Device Locking – Developed By Uniloc USA", *avail. at:* http://www.dksassociates.com/admin/paperfile/ITS%20World%20Paper%20Submission_Uniloc%20_2_.pdf, Oct. 24, 2006.	
		ECONOLITE, "Econolite and Uniloc Partner to Bring Unmatched Infrastructure Security to Advanced Traffic Control Networks with Launch of Strongpoint", *avail. at: http://www.econolite.com/docs/press/20080304_Econolite_StrongPoint.pdf*, Mar. 4, 2008.	

Change(s) applied
to document,
/C.C.B./
6/30/2012

Examiner Signature	/Qing Chen/	Date Considered	08/10/2011

EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /QC/

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/818,906	08/07/2012	8239852	UN-NP-AD-037	8831

96051 7590 07/18/2012

Uniloc USA Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 0 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

Craig Stephen Etchegoyen, Irvine, CA;

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage and facilitate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	PATENT NUMBER	GROUP ART UNIT	FILE WRAPPER LOCATION
12/818,906	8239852	2191	9200

OC000000057209517

Correspondence Address/Fee Address Change

The following fields have been set to Customer Number 15183 on 10/18/2012

- **Correspondence Address**
- **Maintenance Fee Address**

The address of record for Customer Number 15183 is:

15183
NetAuthority
Sean Burdick
7160 Dallas Parkway
Suite 380
Plano, TX 75024

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	PATENT NUMBER	GROUP ART UNIT	FILE WRAPPER LOCATION
12/818,906	8239852	2191	9200

OC000000068258285

Correspondence Address/Fee Address Change

The following fields have been set to Customer Number 96051 on 05/05/2014

- **Correspondence Address**
- **Maintenance Fee Address**

The address of record for Customer Number 96051 is:

96051
Uniloc USA Inc.
Legacy Town Center
7160 Dallas Parkway
Suite 380
Plano, TX 75024

AO 120 (Rev. 08/10)

TO:	REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK
Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas, Marshall Division on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:17-cv-00258	4/3/2017	Eastern District of Texas, Marshall Division

PLAINTIFF	DEFENDANT
UNILOC USA, INC. and UNILOC LUXEMBOURG, S.A.	APPLE INC.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	U.S. No. 9,414,199	8/9/2016	Uniloc Luxembourg, S.A.
2	U.S. No. 8,838,976	9/16/2014	Uniloc Luxembourg, S.A.
3	U.S. No. 8,239,852	8/7/2012	Uniloc Luxembourg, S.A.
4			
5			

In the above entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading	
PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1		
2		
3		
4		
5		

In the above entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director Copy 3—Upon termination of action, mail this copy to Director
Copy 2—Upon filing document adding patent(s), mail this copy to Director Copy 4—Case file copy

Solicitor

AO 120 (Rev. 08/10)

APR 07 2017

Mail Stop 8 Director of the U.S. Patent and Trademark Office P.O. Box 1450 Alexandria, VA 22313-1450	**REPORT ON THE FILING OR DETERMINATION OF AN ACTION REGARDING A PATENT OR TRADEMARK**

In Compliance with 35 U.S.C. § 290 and/or 15 U.S.C. § 1116 you are hereby advised that a court action has been filed in the U.S. District Court Eastern District of Texas, Marshall Division on the following

☐ Trademarks or ☑ Patents. (☐ the patent action involves 35 U.S.C. § 292.):

DOCKET NO.	DATE FILED	U.S. DISTRICT COURT
2:17-cv-00258	4/3/2017	Eastern District of Texas, Marshall Division

PLAINTIFF	DEFENDANT
UNILOC USA, INC. and UNILOC LUXEMBOURG, S.A.	APPLE INC.

	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1	U.S. No. 9,414,199	8/9/2016	Uniloc Luxembourg, S.A.
2	U.S. No. 8,838,976	9/16/2014	Uniloc Luxembourg, S.A.
3	U.S. No. 8,239,852	8/7/2012	Uniloc Luxembourg, S.A.
4			
5			

In the above—entitled case, the following patent(s)/ trademark(s) have been included:

DATE INCLUDED	INCLUDED BY ☐ Amendment ☐ Answer ☐ Cross Bill ☐ Other Pleading		
	PATENT OR TRADEMARK NO.	DATE OF PATENT OR TRADEMARK	HOLDER OF PATENT OR TRADEMARK
1			
2			
3			
4			
5			

In the above--entitled case, the following decision has been rendered or judgement issued:

DECISION/JUDGEMENT

CLERK	(BY) DEPUTY CLERK	DATE

Copy 1—Upon initiation of action, mail this copy to Director **Copy 3—Upon termination of action, mail this copy to Director**
Copy 2—Upon filing document adding patent(s), mail this copy to Director **Copy 4—Case file copy**