invention. Referring to Figure 1, a creator creates a digital work, step 101. The creator will then determine appropriate usage rights and fees, attach them to the digital work, and store them in Repository 1, step 102. The determination of appropriate usage rights and fees will depend on various economic factors. The digital work remains securely in Repository 1 until a request for access is received. The request for access begins with a session initiation by another repository. Here a Repository 2 initiates a session with Repository 1, step 103. As will be described in greater detail below, this session initiation includes steps which helps to insure that the respective repositories are trustworthy. Assuming that a session can be established, Repository 2 may then request access to the Digital Work for a stated purpose, step 104. The purpose may be, for example, to print the digital work or to obtain a copy of the digital work. The purpose will correspond to a specific usage right. In any event, Repository 1 checks the usage rights associated with the digital work to determine if the access to the digital work may be granted, step 105. The check of the usage rights essentially involves a determination of whether a right associated with the access request has been attached to the digital work and if all conditions associated with the right are satisfied. If the access is denied, repository 1 terminates the session with an error message, step 106. If access is granted, repository 1 transmits the digital work to repository 2, step 107. Once the digital work has been transmitted to repository 2, repository 1 and 2 each generate billing information for the access which is transmitted to a credit server, step 108. Such double billing reporting is done to insure against attempts to circumvent the billing process.

Figure 2 illustrates the basic interactions between repository types in the present invention. As will become apparent from Figure 2, the various repository types will serve different functions. It is fundamental that repositories will share a core set of functionality which will enable secure and trusted communications. Referring to Figure 2, a repository 201 represents the general instance of a repository. The repository 201 has two modes of operation; a server mode and a requester mode. When in the server mode, the repository will be receiving and processing access requests to digital works. When in the requester mode, the repository will be initiating requests to access digital works. Repository 201 is general in the sense that its primary purpose is as an exchange medium for digital works. During the course of operation, the repository 201 may communicate with a plurality of other repositories, namely authorization repository 202, rendering repository 203 and master repository 204. Communication between repositories occurs utilizing a repository transaction protocol 205.

Communication with an authorization repository 202 may occur when a digital work being accessed has a condition requiring an authorization. Conceptually, an authorization is a digital certificate such that possession of the certificate is required to gain access to the digital work. An authorization is itself a digital work that can be moved between repositories and subjected to fees and usage rights conditions. An authorization may be required by both repositories involved in an access to a digital work.

Communication with a rendering repository 203 occurs in connection with the rendering of a digital work. As will be described in greater detail below, a rendering repository is coupled with a rendering device (e.g. a printer device) to comprise a rendering system.

Communication with a master repository 205 occurs in connection with obtaining an identification certificate. Identification certificates are the means by which a repository is identified as "trustworthy". The use of identification certificates is described below with respect to the registration transaction.

Figure 3 illustrates the repository 201 coupled to a credit server 301. The credit server 301 is a device which accumulates billing information for the repository 201. The credit server 301 communicates with repository 201 via billing transactions 302 to record billing transactions. Billing transactions are reported to a billing clearinghouse 303 by the credit server 301 on a periodic basis. The credit server 301 communicates to the billing clearinghouse 303 via clearinghouse transactions 304. The clearinghouse transactions 304 enable a secure and encrypted transmission of information to the billing clearinghouse 303.

RENDERING SYSTEMS

A rendering system is generally defined as a system comprising a repository and a rendering device which can render a digital work into its desired form. Examples of a rendering system may be a computer system, a digital audio system, or a printer. A rendering system has the same security features as a repository. The coupling of a rendering repository with the rendering device may occur in a manner suitable for the type of rendering device.

Figure 4a illustrates a printer as an example of a rendering system. Referring to Figure 4, printer system 401 has contained therein a printer repository 402 and a print device 403. It should be noted that the the dashed line defining printer system 401 defines a secure system boundary. Communications within the boundary are assumed to be secure. Depending on the security level, the boundary also represents a barrier intended to provide physical integrity. The printer repository 402 is an instantiation of the rendering repository 205 of Figure 2. The printer repository 402 will in some instances contain an ephemeral copy of a digital work which remains until it is printed out by the print engine 403. In other instances, the printer repository 402 may contain digital works such as fonts, which will remain and can be billed based on use. This design assures that all communication lines between printers and printing devices are

encrypted, unless they are within a physically secure boundary. This design feature eliminates a potential "fault" point through which the digital work could be improperly obtained. The printer device 403 represents the printer components used to create the printed output.

Also illustrated in Figure 4a is the repository 404. The repository 404 is coupled to the printer repository 402. The repository 404 represents an external repository which contains digital works.

Figure 4b is an example of a computer system as a rendering system. A computer system may constitute a "multi-function" device since it may execute digital works (e.g. software programs) and display digital works (e.g. a digitized photograph). Logically, each rendering device can be viewed as having its own repository, although only one physical repository is needed. Referring to Figure 4b, a computer system 410 has contained therein a display/execution repository 411. The display/execution repository 411 is coupled to display device, 412 and execution device 413. The dashed box surrounding the computer system 410 represents a security boundary within which communications are assumed to be secure. The display/execution repository 411 is further coupled to a credit server 414 to report any fees to be billed for access to a digital work and a repository 415 for accessing digital works stored therein.

STRUCTURE OF DIGITAL WORKS

Usage rights are attached directly to digital works. Thus, it is important to understand the structure of a digital work. The structure of a digital work, in particular composite digital works, may be naturally organized into an acyclic structure such as a hierarchy. For example, a magazine has various articles and photographs which may have been created and are owned by different persons. Each of the articles and photographs may represent a node in a hierarchical structure. Consequently, controls, i.e. usage rights, may be placed on each node by the creator. By enabling control and fee billing to be associated with each node, a creator of a work can be assured that the rights and fees are not circumvented.

In the currently preferred embodiment, the file information for a digital work is divided into two files: a "contents" file and a "description tree" file. From the perspective of a repository, the "contents" file is a stream of addressable bytes whose format depends completely on the interpreter used to play, display or print the digital work. The description tree file makes it possible to examine the rights and fees for a work without reference to the content of the digital work. It should be noted that the term description tree as used herein refers to any type of acyclic structure used to represent the relationship between the various components of a digital work.

Figure 5 illustrates the layout of a contents file. Referring to Figure 5, a digital work is comprised of story A 510, advertisement 511, story B 512 and story C 513. It is assumed that the digital work is stored starting at a relative address of 0. Each of the parts of the digital work are stored linearly so that story A 510 is stored at approximately addresses 0-30,000, advertisement 511 at addresses 30,001-40,000, story B 512 at addresses 40,001-60,000 and story C 513 at addresses 60,001-85K. The detail of story A 510 is illustrated in Figure 6. Referring to Figure 6, the story A 510 is further broken down to show text 614 stored at address 0-1500, soldier photo 615 at addresses 1501-10,000, graphics 616 stored at addresses 10,001-25,000 and sidebar 617 stored address 25,001-30,000. Note that the data in the contents file may be compressed (for saving storage) or encrypted (for security).

From Figures 5 and 6 it is readily observed that a digital work can be represented by its component parts as a hierarchy. The description tree for a digital work is comprised of a set of related descriptor blocks (d-blocks). The contents of each d-block is described with respect to Figure 7. Referring to Figure 7, a d-block 700 includes an identifier 701 which is a unique identifier for the work in the repository, a starting address 702 providing the start address of the first byte of the work, a length 703 giving the number of bytes in the work, a rights portion 704 wherein the granted usage rights and their status data are maintained, a parent pointer 705 for pointing to a parent d-block and child pointers 706 for pointing to the child d-blocks. In the currently preferred embodiment, the identifier 701 has two parts. The first part is a unique number assigned to the repository upon manufacture. The second part is a unique number assigned to the work upon creation. The rights portion 704 will contain a data structure, such as a look-up table, wherein the various information associated with a right is maintained. The information required by the respective usage rights is described in more detail below. D-blocks form a strict hierarchy. The top d-block of a work has no parent; all other d-blocks have one parent. The relationship of usage rights between parent and child d-blocks and how conflicts are resolved is described below.

A special type of d-block is a "shell" d-block. A shell d-block adds no new content beyond the content of its parts. A shell d-block is used to add rights and fee information, typically by distributors of digital works.

Figure 8 illustrates a description tree for the digital work of Figure 5. Referring to Figure 8, a top d-block 820 for the digital work points to the various stories and advertisements contained therein. Here, the top d-block 820 points to d-block 821 (representing story A 510), d-block 822 (representing the advertisement 511), d-block 823 (representing story B 512) and and d-block 824 (representing story C 513).

The portion of the description tree for Story A 510 is illustrated in Figure 9. D-block 925 represents text 614, d-block 926 represents photo 615, d-block 927 represents graphics 616 by and d-block 928 represents sidebar 617.

The rights portion 704 of a descriptor block is further illustrated in Figure 10. Figure 10 illustrates a structure which is repeated in the rights portion 704 for each right. Referring to Figure 10, each right will have a right code field 1050 and status information field 1052. The right code field 1050 will contain a unique code assigned to a right. The status information field 1052 will contain information relating to the state of a right and the digital work. Such information is indicated below in Table 1. The rights as stored in the rights portion 704 may typically be in numerical order based on the right code.

TABLE 1

DIGITAL WORK STATE INFORMATION		
Property	**Value**	**Use**
Copies-in-Use	Number	A counter of the number of copies of a work that are in use. Incremented when another copy is used; decremented when use is completed.
Loan-Period	Time-Units	Indicator of the maximum number of time-units that a document can be loaned out
Loaner-Copy	Boolean	Indicator that the current work is a loaned out copy of an authorized digital work.
Remaining-Time	Time-Units	Indicator of the remaining time of use on a metered document right.
Document-Descr	String	A string containing various identifying information about a document. The exact format of this is not specified, but it can include information such as a publisher name, author name, ISBN number, and so on.
Revenue-Owner	RO-Descr	A handle identifying a revenue owner for a digital work. This is used for reporting usage fees.
Publication-Date	Date-Descr	The date that the digital work was published.
History-list	History-Rec	A list of events recording the repostories and dates for operations that copy, transfer, backup, or restore a digital work.

The approach for representing digtal works by separating description data from content assumes that parts of a file are contiguous but takes no position on the actual representation of content. In particular, it is neutral to the question of whether content representation may take an object oriented approach. It would be natural to represent content as objects. In principle, it may be convenient to have content objects that include the billing structure and rights information that is represented in the d-blocks. Such variations in the design of the representation are possible and are viable alternatives but may introduce processing overhead, e.g. the interpretation of the objects.

Digital works are stored in a repository as part of a hierarchical file system. Folders (also termed directories and sub-directories) contain the digital works as well as other folders. Digital works and folders in a folder are ordered in alphabetical order. The digital works are typed to reflect how the files are used. Usage rights can be attached to folders so that the folder itself is treated as a digital work. Access to the folder would then be handled in the same fashion as any other digital work As will be described in more detail below, the contents of the folder are subject to their own rights. Moreover, file management rights may be attached to the folder which define how folder contents can be managed.

ATTACHING USAGE RIGHTS TO A DIGITAL WORK

It is fundamental to the present invention that the usage rights are treated as part of the digital work. As the digital work is distributed, the scope of the granted usage rights will remain the same or may be narrowed. For example, when a digital work is transferred from a document server to a repository, the usage rights may include the right to loan a copy for a predetermined period of time (called the original rights). When the repository loans out a copy of the digital work, the usage rights in the loaner copy (called the next set of rights) could be set to prohibit any further rights to loan out the copy. The basic idea is that one cannot grant more rights than they have.

The attachment of usage rights into a digital work may occur in a variety of ways. If the usage rights will be the same for an entire digital work, they could be attached when the digital work is processed for deposit in the digital work server. In the case of a digital work having different usage rights for the various components, this can be done as the digital work is being created. An authoring tool or digital work assembling tool could be utilized which provides for an automated process of attaching the usage rights.

As will be described below, when a digital work is copied, transferred or loaned, a "next set of rights" can be specified. The"next set of rights" will be attached to the digital work as it is transported.

Resolving Conflicting Rights

Because each part of a digital work may have its own usage rights, there will be instances where the rights of a "contained part" are different from its parent or container part. As a result, conflict rules must be established to dictate when and how a right may be exercised. The hierarchical structure of a digital work facilitates the enforcement of such rules. A "strict" rule would be as follows: a right for a part in a digital work is sanctioned if and only if it is sanctioned for the part, for ancestor d-blocks containing the part and for all descendent d-blocks. By sanctioned, it is meant that (1) each of the respective parts must have the right, and (2) any conditions for exercising the right are satisfied.

It also possible to implement the present invention using a more lenient rule. In the more lenient rule, access to the part may be enabled to the descendent parts which have the right, but access is denied to the descendents which do not.

An example of applying both the strict rule and lenient is illustrated with reference to Figure 11. Referring to Figure 11, a root d-block 1101 has child d-blocks 1102-1105. In this case, root d-block represents a magazine, and each of the child d-blocks 1102-1105 represent articles in the magazine. Suppose that a request is made to PRINT the digital work represented by root d-block 1101 wherein the strict rule is followed. The rights for the root d-block 1101 and child d-blocks 1102-1105 are then examined. Root d-block 1101 and child d-blocks 1102 and 1105 have been granted PRINT rights. Child d-block 1103 has not been granted PRINT rights and child d-block 1104 has PRINT rights conditioned on payment of a usage fee.

Under the strict rule the PRINT right cannot be exercised because the child d-block does not have the PRINT right. Under the lenient rule, the result would be different. The digital works represented by child d-blocks 1102 and 1105 could be printed and the digital work represented by d-block 1104 could be printed so long as the usage fee is paid. Only the digital work represented by d-block 1103 could not be printed. This same result would be accomplished under the strict rule if the requests were directed to each of the individual digital works.

The present invention supports various combinations of allowing and disallowing access. Moreover, as will be described below, the usage rights grammar permits the owner of a digital work to specify if constraints may be imposed on the work by a container part. The manner in which digital works may be sanctioned because of usage rights conflicts would be implementation specific and would depend on the nature of the digital works.

REPOSITORIES

In the description of Figure 2, it was indicated that repositories come in various forms. All repositories provide a core set of services for the transmission of digital works. The manner in which digital works are exchanged is the basis for all transaction between repositories. The various repository types differ in the ultimate functions that they perform. Repositories may be devices themselves, or they may be incorporated into other systems. An example is the rendering repository 203 of Figure 2.

A repository will have associated with it a repository identifier. Typically, the repository identifier would be a unique number assigned to the repository at the time of manufacture. Each repository will also be classified as being in a particular security class. Certain communications and transactions may be conditioned on a repository being in a particular security class. The various security classes are described in greater detail below.

As a prerequisite to operation, a repository will require possession of an identification certificate. Identification certificates are encrypted to prevent forgery and are issued by a Master repository. A master repository plays the role of an authorization agent to enable repositories to receive digital works. Identification certificates must be updated on a periodic basis. Identification certificates are described in greater detail below with respect to the registration transaction.

A repository has both a hardware and functional embodiment. The functional embodiment is typically software executing on the hardware embodiment. Alternatively, the functional embodiment may be embedded in the hardware embodiment such as an Application Specific Integrated Circuit (ASIC) chip.

The hardware embodiment of a repository will be enclosed in a secure housing which if compromised, may cause the repository to be disabled. The basic components of the hardware embodiment of a repository are described with reference to Figure 12. Referring to Figure 12, a repository is comprised of a processing means 1200, storage system 1207, clock 1205 and external interface 1206. The processing means 1200 is comprised of a processor element 1201 and processor memory 1202. The processing means 1201 provides controller, repository transaction and usage rights transaction functions for the repository. Various functions in the operation of the repository such as decryption and/or decompression of digital works and transaction messages are also performed by the processing means 1200. The processor element 1201 may be a microprocessor or other suitable computing component. The processor memory 1202 would typically be further comprised of Read Only Memories (ROM) and Random Access Memories (RAM). Such memories would contain the software instructions utilized by the processor element 1201 in performing the functions of the repository.

The storage system 1207 is further comprised of descriptor storage 1203 and content storage 1204. The description tree storage 1203 will store the description tree for the digital work and the content storage will store the associated content. The description tree storage 1203 and content storage 1204 need not be of the same type of storage medium, nor are they necessarily on the same physical device. So for example, the descriptor storage 1203 may be stored on a solid state storage (for rapid retrieval of the description tree information), while the content storage 1204 may be on a high capacity storage such as an optical disk.

The clock 1205 is used to time-stamp various time based conditions for usage rights or for metering usage fees which may be associated with the digital works. The clock 1205 will have an uninterruptable power supply, e.g. a battery, in order to maintain the integrity of the time-stamps. The external interface means 1206 provides for the signal connection to other repositories and to a credit server. The external interface means 1206 provides for the exchange of signals via such standard interfaces such as RS-232 or Personal Computer Manufacturers Card Industry Association (PCMCIA) standards, or FDDI. The external interface means 1206 may also provide network connectivity.

The functional embodiment of a repository is described with reference to Figure 13. Referring to Figure 13, the functional embodiment is comprised of an operating system 1301, core repository services 1302, usage transaction handlers 1303, repository specific functions, 1304 and a user interface 1305. The operating system 1301 is specific to the repository and would typically depend on the type of processor being used. The operating system 1301 would also provide the basic services for controlling and interfacing between the basic components of the repository.

The core repository services 1302 comprise a set of functions required by each and every repository. The core repository services 1302 include the session initiation transactions which are defined in greater detail below. This set of services also includes a generic ticket agent which is used to "punch" a digital ticket and a generic authorization server for processing authorization specifications. Digital tickets and authorizations are specific mechanisms for controlling the distribution and use of digital works and are described in more detail below. Note that coupled to the core repository services are a plurality of identification certificates 1306. The identification certificates 1306 are required to enable the use of the repository.

The usage transactions handlers 1303 comprise functionality for processing access requests to digital works and for billing fees based on access. The usage transactions supported will be different for each repository type. For example, it may not be necessary for some repositories to handle access requests for digital works.

The repository specific functionality 1304 comprises functionality that is unique to a repository. For example, the master repository has special functionality for issuing digital certificates and maintaining encryption keys. The repository specific functionality 1304 would include the user interface implementation for the repository.

Repository Security Classes

For some digital works the losses caused by any individual instance of unauthorized copying is insignificant and the chief economic concern lies in assuring the convenience of access and low-overhead billing. In such cases, simple and inexpensive handheld repositories and network-based workstations may be suitable repositories, even though the measures and guarantees of security are modest.

At the other extreme, some digital works such as a digital copy of a first run movie or a bearer bond or stock certificate would be of very high value so that it is prudent to employ caution and fairly elaborate security measures to ensure that they are not copied or forged. A repository suitable for holding such a digital work could have elaborate measures for ensuring physical integrity and for verifying authorization before use.

By arranging a universal protocol, all kinds of repositories can communicate with each other in principle. However, creators of some works will want to specify that their works will only be transferred to repositories whose level of security is high enough. For this reason, document repositories have a ranking system for classes and levels of security. The security classes in the currently preferred embodiment are described in Table 2.

TABLE 2

REPOSITORY SECURITY LEVELS	
Level	Description of Security
0	Open system. Document transmission is unencrypted. No digital certificate is required for identification. The security of the system depends mostly on user honesty, since only modest knowledge may be needed to circumvent the security measures. The repository has no provisions for preventing unauthorized programs from running and accessing or copying files. The system does not prevent the use of removable storage and does not encrypt stored files.
1	Minimal security. Like the previous class except that stored files are minimally encrypted, including ones on removable storage.

TABLE 2 (continued)

REPOSITORY SECURITY LEVELS	
Level	Description of Security
2	Basic security. Like the previous class except that special tools and knowledge are required to compromise the programming, the contents of the repository, or the state of the clock. All digital communications are encrypted. A digital certificate is provided as identification. Medium level encryption is used. Repository identification number is unforgeable.
3	General security. Like the previous class plus the requirement of special tools are needed to compromise the physical integrity of the repository and that modest encryption is used on all transmissions. Password protection is required to use the local user interface. The digital clock system cannot be reset without authorization. No works would be stored on removable storage. When executing works as programs, it runs them in their own address space and does not give them direct access to any file storage or other memory containing system code or works. They can access works only through the transmission transaction protocol.
4	Like the previous class except that high level encryption is used on all communications. Sensors are used to record attempts at physical and electronic tampering. After such tampering, the repository will not perform other transactions until it has reported such tampering to a designated server.
5	Like the previous class except that if the physical or digital attempts at tampering exceed some preset thresholds that threaten the physical integrity of the repository or the integrity of digital and cryptographic barriers, then the repository will save only document description records of history but will erase or destroy any digital identifiers that could be misused if released to an unscrupulous party. It also modifies any certificates of authenticity to indicate that the physical system has been compromised. It also erases the contents of designated documents.
6	Like the previous class except that the repository will attempt wireless communication to report tampering and will employ noisy alarms.
10	This would correspond to a very high level of security. This server would maintain constant communications to remote security systems reporting transactions, sensor readings, and attempts to circumvent security.

The characterization of security levels described in Table 2 is not intended to be fixed. More important is the idea of having different security levels for different repositories. It is anticipated that new security classes and requirements will evolve according to social situations and changes in technology.

Repository User Interface

A user interface is broadly defined as the mechanism by which a user interacts with a repository in order to invoke transactions to gain access to a digital work, or exercise usage rights. As described above, a repository may be embodied in various forms. The user interface for a repository will differ depending on the particular embodiment. The user interface may be a graphical user interface having icons representing the digital works and the various transactions that may be performed. The user interface may be a generated dialog in which a user is prompted for information.

The user interface itself need not be part of the repository. As a repository may be embedded in some other device, the user interface may merely be a part of the device in which the repository is embedded. For example, the repository could be embedded in a "card" that is inserted into an available slot in a computer system. The user interface may be a combination of a display, keyboard, cursor control device and software executing on the computer system.

At a minimum, the user interface must permit a user to input information such as access requests and alpha numeric data and provide feedback as to transaction status. The user interface will then cause the repository to initiate the suitable transactions to service the request. Other facets of a particular user interface will depend on the functionality that a repository will provide.

CREDIT SERVERS

In the present invention, fees may be associated with the exercise of a right. The requirement for payment of fees is described with each version of a usage right in the usage rights language. The recording and reporting of such fees is performed by the credit server. One of the capabilities enabled by associating fees with rights is the possibility of

supporting a wide range of charging models. The simplest model, used by conventional software, is that there is a single fee at the time of purchase, after which the purchaser obtains unlimited rights to use the work as often and for as long as he or she wants. Alternative models, include metered use and variable fees. A single work can have different fees for different uses. For example, viewing a photograph on a display could have different fees than making a hardcopy or including it in a newly created work. A key to these alternative charging models is to have a low overhead means of establishing fees and accounting for credit on these transactions.

A credit server is a computational system that reliably authorizes and records these transactions so that fees are billed and paid. The credit server reports fees to a billing clearinghouse. The billing clearinghouse manages the financial transactions as they occur. As a result, bills may be generated and accounts reconciled. Preferably, the credit server would store the fee transactions and periodically communicate via a network with the billing clearinghouse for reconciliation. In such an embodiment, communications with the billing clearinghouse would be encrypted for integrity and security reasons. In another embodiment, the credit server acts as a "debit card" where transactions occur in "real-time" against a user account.

A credit server is comprised of memory, a processing means, a clock, and interface means for coupling to a repository and a financial institution (e.g. a modem). The credit server will also need to have security and authentication functionality. These elements are essentially the same elements as those of a repository. Thus, a single device can be both a repository and a credit server, provided that it has the appropriate processing elements for carrying out the corresponding functions and protocols. Typically, however, a credit server would be a card-sized system in the possession of the owner of the credit. The credit server is coupled to a repository and would interact via financial transactions as described below. Interactions with a financial institution may occur via protocols established by the financial institutions themselves.

In the currently preferred embodiment credit servers associated with both the server and the repository report the financial transaction to the billing clearinghouse. For example, when a digital work is copied by one repository to another for a fee, credit servers coupled to each of the repositories will report the transaction to the billing clearinghouse. This is desirable in that it insures that a transaction will be accounted for in the event of some break in the communication between a credit server and the billing clearinghouse. However, some implementations may embody only a single credit server reporting the transaction to minimize transaction processing at the risk of losing some transactions.

USAGE RIGHTS LANGUAGE

The present invention uses statements in a high level "usage rights language" to define rights associated with digital works and their parts. Usage rights statements are interpreted by repositories and are used to determine what transactions can be successfully carried out for a digital work and also to determine parameters for those transactions. For example, sentences in the language determine whether a given digital work can be copied, when and how it can be used, and what fees (if any) are to be charged for that use. Once the usage rights statements are generated, they are encoded in a suitable form for accessing during the processing of transactions.

Defining usage rights in terms of a language in combination with the hierarchical representation of a digital work enables the support of a wide variety of distribution and fee schemes. An example is the ability to attach multiple versions of a right to a work. So a creator may attach a PRINT right to make 5 copies for $10.00 and a PRINT right to make unlimited copies for $100.00. A purchaser may then choose which option best fits his needs. Another example is that rights and fees are additive. So in the case of a composite work, the rights and fees of each of the components works is used in determining the rights and fees for the work as a whole.

The basic contents of a right are illustrated in Figure 14. Referring to Figure 14, a right 1450 has a transactional component 1451 and a specifications component 1452. A right 1450 has a label (e.g. COPY or PRINT) which indicates the use or distribution privileges that are embodied by the right. The transactional component 1451 corresponds to a particular way in which a digital work may be used or distributed. The transactional component 1451 is typically embodied in software instructions in a repository which implement the use or distribution privileges for the right. The specifications components 1452 are used to specify conditions which must be satisfied prior to the right being exercised or to designate various transaction related parameters. In the currently preferred embodiment, these specifications include copy count 1453, Fees and Incentives 1454, Time 1455, Access and Security 1456 and Control 1457. Each of these specifications will be described in greater detail below with respect to the language grammar elements.

The usage rights language is based on the grammar described below. A grammar is a convenient means for defining valid sequence of symbols for a language. In describing the grammar the notation "[a | b | c] is used to indicate distinct choices among alternatives. In this example, a sentence can have either an "a", "b" or "c". It must include exactly one of them. The braces { } are used to indicate optional items. Note that brackets, bars and braces are used to describe the language of usage rights sentences but do not appear in actual sentences in the language.

In contrast, parentheses are part of the usage rights language. Parentheses are used to group items together in lists. The notation (x*) is used to indicate a variable length list, that is, a list containing one or more items of type x.

The notation (x)* is used to indicate a variable number of lists containing x.

Keywords in the grammar are words followed by colons. Keywords are a common and very special case in the language. They are often used to indicate a single value, typically an identifier. In many cases, the keyword and the parameter are entirely optional. When a keyword is given, it often takes a single identifier as its value. In some cases, the keyword takes a list of identifiers.

In the usage rights language, time is specified in an hours:minutes:seconds (or hh:mm:ss) representation. Time zone indicators, e.g. PDT for Pacific Daylight Time, may also be specified. Dates are represented as year/ month/day (or YYYY/MMM/DD). Note that these time and date representations may specify moments in time or units of time Money units are specified in terms of dollars.

Finally, in the usage rights language, various "things" will need to interact with each other. For example, an instance of a usage right may specify a bank account, a digital ticket, etc.. Such things need to be identified and are specified herein using the suffix "-ID."

The Usage Rights Grammar is listed in its entirety in Figure 15 and is described below.

Grammar element 1501 **"Digital Work Rights: = (Rights*)"** define the digital work rights as a set of rights. The set of rights attached to a digital work define how that digital work may be transferred, used, performed or played. A set of rights will attach to the entire digital work and in the case of compound digital works, each of the components of the digital work. The usage rights of components of a digital may be different.

Grammar element 1502 **"Right : = (Right-Code {Copy-Count} {Control-Spec} {Time-Spec} {Access-Spec} {Fee-Spec})"** enumerates the content of a right. Each usage right must specify a right code. Each right may also optionally specify conditions which must be satisfied before the right can be exercised. These conditions are copy count, control, time, access and fee conditions. In the currently preferred embodiment, for the optional elements, the following defaults apply: copy count equals 1, no time limit on the use of the right, no access tests or a security level required to use the right and no fee is required. These conditions will each be described in greater detail below.

It is important to note that a digital work may have multiple versions of a right, each having the same right code. The multiple version would provide alternative conditions and fees for accessing the digital work.

Grammar element 1503 **"Right-Code : = Render-Code I Transport-Code I File-Management-Code I Derivative-Works- Code Configuration-Code"** distinguishes each of the specific rights into a particular right type (although each right is identified by distinct right codes). In this way, the grammar provides a catalog of possible rights that can be associated with parts of digital works. In the following, rights are divided into categories for convenience in describing them.

Grammar element 1504 **"Render-Code : = [Play: {Player: Player-ID} I Print: {Printer: Printer-ID}]"** lists a category of rights all involving the making of ephemeral, transitory, or non-digital copies of the digital work. After use the copies are erased.

- Play — A process of rendering or performing a digital work on some processor. This includes such things as playing digital movies, playing digital music, playing a video game, running a computer program, or displaying a document on a display.
- Print — To render the work in a medium that is not further protected by usage rights, such as printing on paper.

Grammar element 1505 **"Transport-Code : = [Copy I Transfer I Loan {Remaining-Rights: Next-Set-of-Rights}] {(Next-Copy-Rights: Next-Set of Rights)}"** lists a category of rights involving the making of persistent, usable copies of the digital work on other repositories. The optional Next-Copy-Rights determine the rights on the work after it is transported. If this is not specified, then the rights on the transported copy are the same as on the original. The optional Remaining-Rights specify the rights that remain with a digital work when it is loaned out. If this is not specified, then the default is that no rights can be exercised when it is loaned out.

- Copy — Make a new copy of a work
- Transfer — Moving a work from one repository to another.
- Loan — Temporarily loaning a copy to another repository for a specified period of time.

Grammar element 1506 **"File-Management-Code : = Backup {Back-Up-Copy-Rights: Next-Set -of Rights}I Restore I Delete I Folder I Directory {Name:Hide-Local I Hide - Remote}{Parts:Hide-Local I Hide-Remote}"** lists a category of rights involving operations for file management, such as the making of backup copies to protect the copy owner against catastrophic equipment failure.

Many software licenses and also copyright law give a copy owner the right to make backup copies to protect against catastrophic failure of equipment. However, the making of uncontrolled backup copies is inherently at odds with the ability to control usage, since an uncontrolled backup copy can be kept and then restored even after the authorized copy was sold.

The File management rights enable the making and restoring of backup copies in a way that respects usage rights, honoring the requirements of both the copy owner and the rights grantor and revenue owner. Backup copies of work descriptions (including usage rights and fee data) can be sent under appropriate protocol and usage rights control to other document repositories of sufficiently high security. Further rights permit organization of digital works into folders which themselves are treated as digital works and whose contents may be "hidden" from a party seeking to determine the contents of a repository.

- Backup To make a backup copy of a digital work as protection against media failure.
- Restore To restore a backup copy of a digital work.
- Delete To delete or erase a copy of a digital work.
- Folder To create and name folders, and to move files and folders between folders.
- Directory To hide a folder or its contents.

Grammar element 1507 **"Derivative-Works-Code: [Extract | Embed | Edit {Process: Process-ID}] {Next-Copy-Rights : Next-Set-of Rights}"** lists a category of rights involving the use of a digital work to create new works.

- Extract To remove a portion of a work, for the purposes of creating a new work.
- Embed To include a work in an existing work.
- Edit To alter a digital work by copying, selecting and modifying portions of an existing digital work.

Grammar element 1508 **"Configuration-Code : = Install | Uninstall"** lists a category of rights for installing and uninstalling software on a repository (typically a rendering repository.) This would typically occur for the installation of a new type of player within the rendering repository.

- Install: To install new software on a repository.
- Uninstall: To remove existing software from a repository.

Grammar element 1509 **"Next-Set-of-Rights : = {(Add : Set-Of-Rights)} {(Delete: Set-Of-Rights)} {(Replace: Set-Of-Rights)} {(Keep: Set-Of-Rights)}"** defines how rights are carried forward for a copy of a digital work. If the Next-Copy-Rights is not specified, the rights for the next copy are the same as those of the current copy. Otherwise, the set of rights for the next copy can be specified. Versions of rights after Add: are added to the current set of rights. Rights after Delete: are deleted from the current set of rights. If only right codes are listed after Delete:, then all versions of rights with those codes are deleted. Versions of rights after Replace: subsume all versions of rights of the same type in the current set of rights.

If Remaining-Rights is not specified, then there are no rights for the original after all Loan copies are loaned out. If Remaining-Rights is specified, then the Keep: token can be used to simplify the expression of what rights to keep behind. A list of right codes following keep means that all of the versions of those listed rights are kept in the remaining copy. This specification can be overridden by subsequent Delete: or Replace: specifications.

Copy Count Specification

For various transactions, it may be desirable to provide some limit as to the number of "copies" of the work which may be exercised simultaneously for the right. For example, it may be desirable to limit the number of copies of a digital work that may be loaned out at a time or viewed at a time.

Grammar element 1510 **"Copy-Count : = (Copies: positive-integer | 0 | unlimited)"** provides a condition which defines the number of "copies" of a work subject to the right . A copy count can be 0, a fixed number, or unlimited. The copy-count is associated with each right, as opposed to there being just a single copy-count for the digital work. The Copy-Count for a right is decremented each time that a right is exercised. When the Copy-Count equals zero, the right can no longer be exercised. If the Copy-Count is not specified, the default is one.

Control Specification

Rights and fees depend in general on rights granted by the creator as well as further restrictions imposed by later distributors. Control specifications deal with interactions between the creators and their distributors governing the imposition of further restrictions and fees. For example, a distributor of a digital work may not want an end consumer of a digital work to add fees or otherwise profit by commercially exploiting the purchased digital work.

Grammar element 1511 **"Control-Spec : = (Control: {Restrictable | Unrestrictable} {Unchargeable | Chargeable})"** provides a condition to specify the effect of usage rights and fees of parents on the exercise of the right. A

digital work is restrictable if higher level d-blocks can impose further restrictions (time specifications and access specifications) on the right. It is unrestrictable if no further restrictions can be imposed. The default setting is restrictable. A right is unchargeable if no more fees can be imposed on the use of the right. It is chargeable if more fees can be imposed. The default is chargeable.

Time Specification

It is often desirable to assign a start date or specify some duration as to when a right may be exercised. Grammar element 1512 **"Time-Spec : = ({Fixed-Interval | Sliding-Interval | Meter-Time} Until: Expiration-Date)"** provides for specification of time conditions on the exercise of a right. Rights may be granted for a specified time. Different kinds of time specifications are appropriate for different kinds of rights. Some rights may be exercised during a fixed and predetermined duration. Some rights may be exercised for an interval that starts the first time that the right is invoked by some transaction. Some rights may be exercised or are charged according to some kind of metered time, which may be split into separate intervals. For example, a right to view a picture for an hour might be split into six ten minute viewings or four fifteen minute viewings or twenty three minute viewings.

The terms "time" and "date" are used synonymously to refer to a moment in time. There are several kinds of time specifications. Each specification represents some limitation on the times over which the usage right applies. The Expiration-Date specifies the moment at which the usage right ends. For example, if the Expiration-Date is "Jan 1, 1995," then the right ends at the first moment of 1995. If the Expiration-Date is specified as "forever", then the rights are interpreted as continuing without end. If only an expiration date is given, then the right can be exercised as often as desired until the expiration date.

Grammar element 1513 **"Fixed-Interval := From: Start-Time"** is used to define a predetermined interval that runs from the start time to the expiration date.

Grammar element 1514 **"Sliding-Interval : = Interval: Use-Duration"** is used to define an indeterminate (or "open") start time. It sets limits on a continuous period of time over which the contents are accessible. The period starts on the first access and ends after the duration has passed or the expiration date is reached, whichever comes first. For example, if the right gives 10 hours of continuous access, the use-duration would begin when the first access was made and end 10 hours later.

Grammar element 1515 **"Meter-Time: = Time-Remaining: Remaining-Use"** is used to define a "meter time," that is, a measure of the time that the right is actually exercised. It differs from the Sliding-Interval specification in that the time that the digital work is in use need not be continuous. For example, if the rights guarantee three days of access, those days could be spread out over a month. With this specification, the rights can be exercised until the meter time is exhausted or the expiration date is reached, whichever comes first.

Remaining-Use: = Time-Unit
Start-Time: = Time-Unit
Use-Duration: = Time-Unit
All of the time specifications include time-unit specifications in their ultimate instantiation.

Security Class and Authorization Specification

The present invention provides for various security mechanisms to be introduced into a distribution or use scheme. Grammar element 1516 **"Access-Spec : = ({SC: Security-Class} {Authorization: Authorization-ID*} {Other-Authorization: Authorization-ID*} {Ticket: Ticket-ID})"** provides a means for restricting access and transmission. Access specifications can specify a required security class for a repository to exercise a right or a required authorization test that must be satisfied.

The keyword **"SC:"** is used to specify a minimum security level for the repositories involved in the access. If **"SC: "** is not specified, the lowest security level is acceptable.

The optional **"Authorization:"** keyword is used to specify required authorizations on the same repository as the work. The optional **"Other-Authorization:"** keyword is used to specify required authorizations on the other repository in the transaction.

The optional **"Ticket:"** keyword specifies the identity of a ticket required for the transaction. A transaction involving digital tickets must locate an appropriate digital ticket agent who can "punch" or otherwise validate the ticket before the transaction can proceed. Tickets are described in greater detail below.

In a transaction involving a repository and a document server, some usage rights may require that the repository have a particular authorization, that the server have some authorization, or that both repositories have (possibly different) authorizations. Authorizations themselves are digital works (hereinafter referred to as an authorization object) that can be moved between repositories in the same manner as other digital works. Their copying and transferring is

subject to the same rights and fees as other digital works. A repository is said to have an authorization if that authorization object is contained within the repository.

In some cases, an authorization may be required from a source other than the document server and repository. An authorization object referenced by an Authorization-ID can contain digital address information to be used to set up a communications link between a repository and the authorization source. These are analogous to phone numbers. For such access tests, the communication would need to be established and authorization obtained before the right could be exercised.

For one-time usage rights, a variant on this scheme is to have a digital ticket. A ticket is presented to a digital ticket agent, whose type is specified on the ticket. In the simplest case, a certified generic ticket agent, available on all repositories, is available to "punch" the ticket. In other cases, the ticket may contain addressing information for locating a "special" ticket agent. Once a ticket has been punched, it cannot be used again for the same kind of transaction (unless it is unpunched or refreshed in the manner described below.) Punching includes marking the ticket with a timestamp of the date and time it was used. Tickets are digital works and can be copied or transferred between repositories according to their usage rights.

In the currently preferred embodiment, a "punched" ticket becomes "unpunched" or "refreshed" when it is copied or extracted. The Copy and Extract operations save the date and time as a property of the digital ticket. When a ticket agent is given a ticket, it can simply check whether the digital copy was made after the last time that it was punched. Of course, the digital ticket must have the copy or extract usage rights attached thereto.

The capability to unpunch a ticket is inportant in the following cases:

- A digital work is circulated at low cost with a limitation that it can be used only once.
- A digital work is circulated with a ticket that can be used once to give discounts on purchases of other works.
- A digital work is circulated with a ticket (included in the purchase price and possibly embedded in the work) that can be used for a future upgrade.

In each of these cases, if a paid copy is made of the digital work (including the ticket) the new owner would expect to get a fresh (unpunched) ticket, whether the copy seller has used the work or not. In contrast, loaning a work or simply transferring it to another repository should not revitalize the ticket.

Usage Fees and Incentives Specification

The billing for use of a digital work is fundamental to a commercial distribution system. Grammar Element 1517 **"Fee-Spec: = {Scheduled-Discount} Regular-Fee-Spec | Scheduled-Fee-Spec | Markup-Spec"** provides a range of options for billing for the use of digital works.

A key feature of this approach is the development of low-overhead billing for transactions in potentially small amounts. Thus, it becomes feasible to collect fees of only a few cents each for thousands of transactions.

The grammar differentiates between uses where the charge is per use from those where it is metered by the time unit. Transactions can support fees that the user pays for using a digital work as well as incentives paid by the right grantor to users to induce them to use or distribute the digital work.

The optional scheduled discount refers to the rest of the fee specification--discounting it by a percentage over time. If it is not specified, then there is no scheduled discount. Regular fee specifications are constant over time. Scheduled fee specifications give a schedule of dates over which the fee specifications change. Markup specifications are used in d-blocks for adding a percentage to the fees already being charged.

Grammar Element 1518 **"Scheduled-Discount: = (Scheduled-Discount: (Time-Spec Percentage)*)"** A Scheduled-Discount is a essentially a scheduled modifier of any other fee specification for this version of the right of the digital work. (It does not refer to children or parent digital works or to other versions of rights.). It is a list of pairs of times and percentages. The most recent time in the list that has not yet passed at the time of the transaction is the one in effect. The percentage gives the discount percentage. For example, the number 10 refers to a 10% discount.

Grammar Element 1519 **"Regular-Fee-Spec : = ({Fee: | Incentive: } [Per-Use-Spec | Metered-Rate-Spec | Best-Price-Spec | Call-For-Price-Spec] {Min: Money-Unit Per: Time-Spec)(Max: Money-Unit Per: Time-Spec} To: Account-ID)"** provides for several kinds of fee specifications.

Fees are paid by the copy-owner/user to the revenue-owner if Fee: is specified. Incentives are paid by the revenue-owner to the user if Incentive: is specified. If the Min: specification is given, then there is a minimum fee to be charged per time-spec unit for its use. If the Max: specification is given, then there is a maximum fee to be charged per time-spec for its use. When Fee: is specified, Account-ID identifies the account to which the fee is to be paid. When Incentive: is specified, Account-ID identifies the account from which the fee is to be paid.

Grammar element 1520 **"Per-Use-Spec: = Per-Use: Money-unit"** defines a simple fee to be paid every time the right is exercised, regardless of how much time the transaction takes.

Grammar element 1521 **"Metered-Rate-Spec : = Metered: Money-Unit Per: Time-Spec"** defines a metered-rate fee paid according to how long the right is exercised. Thus, the time it takes to complete the transaction determines the fee.

Grammar element 1522 **"Best-Price-Spec := Best-Price: Money-unit Max: Money-unit"** is used to specify a best-price that is determined when the account is settled. This specification is to accommodate special deals, rebates, and pricing that depends on information that is not available to the repository. All fee specifications can be combined with tickets or authorizations that could indicate that the consumer is a wholesaler or that he is a preferred customer, or that the seller be authorized in some way. The amount of money in the Max: field is the maximum amount that the use will cost. This is the amount that is tentatively debited from the credit server. However, when the transaction is ultimately reconciled, any excess amount will be returned to the consumer in a separate transaction.

Grammar element 1523 **"Call-For-Price-Spec:= Call-For-Price"** is similar to a **"Best-Price-Spec"** in that it is intended to accommodate cases where prices are dynamic. **A Call-For-Price Spec** requires a communication with a dealer to determine the price. This option cannot be exercised if the repository cannot communicate with a dealer at the time that the right is exercised. It is based on a secure transaction whereby the dealer names a price to exercise the right and passes along a deal certificate which is referenced or included in the billing process.

Grammar element 1524 **"Scheduled-Fee-Spec: = (Schedule: (Time-Spec Regular-Fee-Spec)*)"** is used to provide a schedule of dates over which the fee specifications change. The fee specification with the most recent date not in the future is the one that is in effect. This is similar to but more general than the scheduled discount. It is more general, because it provides a means to vary the fee agreement for each time period.

Grammar element 1525 **"Markup-Spec: = Markup: percentage To: Account-ID"** is provided for adding a percentage to the fees already being charged. For example, a 5% markup means that a fee of 5% of cumulative fee so far will be allocated to the distributor. A markup specification can be applied to all of the other kinds of fee specifications. It is typically used in a shell provided by a distributor. It refers to fees associated with d-blocks that are parts of the current d-block. This might be a convenient specification for use in taxes, or in distributor overhead.

REPOSITORY TRANSACTIONS

When a user requests access to a digital work, the repository will initiate various transactions. The combination of transactions invoked will depend on the specifications assigned for a usage right. There are three basic types of transactions, Session Initiation Transactions, Financial Transactions and Usage Transactions. Generally, session initiation transactions are initiated first to establish a valid session. When a valid session is established, transactions corresponding to the various usage rights are invoked. Finally, request specific transactions are performed.

Transactions occur between two repositories (one acting as a server), between a repository and a document playback platform (e.g. for executing or viewing), between a repository and a credit server or between a repository and an authorization server. When transactions occur between more than one repository, it is assumed that there is a reliable communication channel between the repositories. For example, this could be a TCP/IP channel or any other commercially available channel that has built-in capabilities for detecting and correcting transmission errors. However, it is not assumed that the communication channel is secure. Provisions for security and privacy are part of the requirements for specifying and implementing repositories and thus form the need for various transactions.

Message Transmission

Transactions require that there be some communication between repositories. Communication between repositories occurs in units termed as messages. Because the communication line is assumed to be unsecure, all communications with repositories that are above the lowest security class are encrypted utilizing a public key encryption technique. Public key encryption is a well known technique in the encryption arts. The term key refers to a numeric code that is used with encryption and decryption algorithms. Keys come in pairs, where "writing keys" are used to encrypt data and "checking keys" are used to decrypt data. Both writing and checking keys may be public or private. Public keys are those that are distributed to others. Private keys are maintained in confidence.

Key management and security is instrumental in the success of a public key encryption system. In the currently preferred embodiment, one or more master repositories maintain the keys and create the identification certificates used by the repositories.

When a sending repository transmits a message to a receiving repository, the sending repository encrypts all of its data using the public writing key of the receiving repository. The sending repository includes its name, the name of the receiving repository, a session identifier such as a nonce (described below), and a message counter in each message.

In this way, the communication can only be read (to a high probability) by the receiving repository, which holds the private checking key for decryption. The auxiliary data is used to guard against various replay attacks to security. If

messages ever arrive with the wrong counter or an old nonce, the repositories can assume that someone is interfering with communication and the transaction terminated.

The respective public keys for the repositories to be used for encryption are obtained in the registration transaction described below.

Session Initiation Transactions

A usage transaction is carried out in a session between repositories. For usage transactions involving more than one repository, or for financial transactions between a repository and a credit server, a registration transaction is performed. A second transaction termed a login transaction, may also be needed to initiate the session. The goal of the registration transaction is to establish a secure channel between two repositories who know each others identities. As it is assumed that the communication channel between the repositories is reliable but not secure, there is a risk that a non-repository may mimic the protocol in order to gain illegitimate access to a repository.

The registration transaction between two repositories is described with respect to Figures 16 and 17. The steps described are from the perspective of a "repository-1" registering its identity with a "repository-2". The registration must be symmetrical so the same set of steps will be repeated for repository-2 registering its identity with repository-1. Referring to Figure 16, repository-1 first generates an encrypted registration identifier, step 1601 and then generates a registration message, step 1602. A registration message is comprised of an identifier of a master repository, the identification certificate for the repository-1 and an encrypted random registration identifier. The identification certificate is encrypted by the master repository in its private key and attests to the fact that the repository (here repository-1) is a bona fide repository. The identification certificate also contains a public key for the repository, the repository security level and a timestamp (indicating a time after which the certificate is no longer valid.) The registration identifier is a number generated by the repository for this registration. The registration identifier is unique to the session and is encrypted in repository-1's private key. The registration identifier is used to improve security of authentication by detecting certain kinds of communications based attacks. Repository-1 then transmits the registration message to repository-2, step 1603.

Upon receiving the registration message, repository-2 determines if it has the needed public key for the master repository, step 1604. If repository-2 does not have the needed public key to decrypt the identification certificate, the registration transaction terminates in an error, step 1618.

Assuming that repository-2 has the proper public key the identification certificate is decrypted, step 1605. Repository-2 saves the encrypted registration identifier, step 1606, and extracts the repository identifier, step 1607. The extracted repository identifier is checked against a "hotlist" of compromised document repositories, step 1608. In the currently preferred embodiment, each repository will contain "hotlists" of compromised repositories. If the repository is on the "hotlist", the registration transaction terminates in an error per step 1618. Repositories can be removed from the hotlist when their certificates expire, so that the list does not need to grow without bound. Also, by keeping a short list of hotlist certificates that it has previously received, a repository can avoid the work of actually going through the list. These lists would be encrypted by a master repository. A minor variation on the approach to improve efficiency would have the repositories first exchange lists of names of hotlist certificates, ultimately exchanging only those lists that they had not previously received. The "hotlists" are maintained and distributed by Master repositories.

Note that rather than terminating in error, the transaction could request that another registration message be sent based on an identification certificate created by another master repository. This may be repeated until a satisfactory identification certificate is found, or it is determined that trust cannot be established.

Assuming that the repository is not on the hotlist, the repository identification needs to be verified. In other words, repository-2 needs to validate that the repository on the other end is really repository-1. This is termed performance testing and is performed in order to avoid invalid access to the repository via a counterfeit repository replaying a recording of a prior session initiation between repository-1 and repository-2. Performance testing is initiated by repository-2 generating a performance message, step 1609. The performance message consists of a nonce, the names of the respective repositories, the time and the registration identifier received from repository-1. A nonce is a generated message based on some random and variable information (e.g. the time or the temperature.) The nonce is used to check whether repository-1 can actually exhibit correct encrypting of a message using the private keys it claims to have, on a message that it has never seen before. The performance message is encrypted using the public key specified in the registration message of repository-1. The performance message is transmitted to repository-1, step 1610, where it is decrypted by repository-1 using its private key, step 1611. Repository-1 then checks to make sure that the names of the two repositories are correct, step 1612, that the time is accurate, step 1613 and that the registration identifier corresponds to the one it sent, step 1614. If any of these tests fails, the transaction is terminated per step 1616. Assuming that the tests are passed, repository-1 transmits the nonce to repository-2 in the clear, step 1615. Repository-2 then compares the received nonce to the original nonce, step 1617. If they are not identical, the registration transaction terminates in an error per step 1618. If they are the same, the registration transaction has successfully completed.

At this point, assuming that the transaction has not terminated, the repositories exchange messages containing session keys to be used in all communications during the session and synchronize their clocks. Figure 17 illustrates the session information exchange and clock synchronization steps (again from the perspective of repository-1.) Referring to Figure 17, repository-1 creates a session key pair, step 1701. A first key is kept private and is used by repository-1 to encrypt messages. The second key is a public key used by repository-2 to decrypt messages. The second key is encrypted using the public key of repository-2, step 1702 and is sent to repository-2, step 1703. Upon receipt, repository-2 decrypts the second key, step 1704. The second key is used to decrypt messages in subsequent communications. When each repository has completed this step, they are both convinced that the other repository is bona fide and that they are communicating with the original. Each repository has given the other a key to be used in decrypting further communications during the session. Since that key is itself transmitted in the public key of the receiving repository only it will be able to decrypt the key which is used to decrypt subsequent messages.

After the session information is exchanged, the repositories must synchronize their clocks. Clock synchronization is used by the repositories to establish an agreed upon time base for the financial records of their mutual transactions. Referring back to Figure 17, repository-2 initiates clock synchronization by generating a time stamp exchange message, step 1705, and transmits it to repository-1, step 1706. Upon receipt, repository-1 generates its own time stamp message, step 1707 and transmits it back to repository-2, step 1708. Repository-2 notes the current time, step 1709 and stores the time received from repository-1, step 1710. The current time is compared to the time received from repository-1, step 1711. The difference is then checked to see if it exceeds a predetermined tolerance (e.g. one minute), step 1712. If it does, repository-2 terminates the transaction as this may indicate tampering with the repository, step 1713. If not repository-2 computes an adjusted time delta, step 1714. The adjusted time delta is the difference between the clock time of repository-2 and the average of the times from repository-1 and repository-2.

To achieve greater accuracy, repository-2 can request the time again up to a fixed number of times (e.g. five times), repeat the clock synchronization steps, and average the results.

A second session initiation transaction is a Login transaction. The Login transaction is used to check the authenticity of a user requesting a transaction. A Login transaction is particularly prudent for the authorization of financial transactions that will be charged to a credit server. The Login transaction involves an interaction between the user at a user interface and the credit server associated with a repository. The information exchanged here is a login string supplied by the repository/credit server to identify itself to the user, and a Personal Identification Number (PIN) provided by the user to identify himself to the credit server. In the event that the user is accessing a credit server on a repository different from the one on which the user interface resides, exchange of the information would be encrypted using the public and private keys of the respective repositories.

Billing Transactions

Billing Transactions are concerned with monetary transactions with a credit server. Billing Transactions are carried out when all other conditions are satisfied and a usage fee is required for granting the request. For the most part, billing transactions are well understood in the state of the art. These transactions are between a repository and a credit server, or between a credit server and a billing clearinghouse. Briefly, the required transactions include the following:

- Registration and LOGIN transactions by which the repository and user establish their bona fides to a credit server. These transactions would be entirely internal in cases where the repository and credit server are implemented as a single system.
- Registration and LOGIN transactions, by which a credit server establishes its bona fides to a billing clearinghouse.
- An Assign-fee transaction to assign a charge. The information in this transaction would include a transaction identifier, the identities of the repositories in the transaction, and a list of charges from the parts of the digital work. If there has been any unusual event in the transaction such as an interruption of communications, that information is included as well.
- A Begin-charges transaction to assign a charge. This transaction is much the same as an assign-fee transaction except that it is used for metered use. It includes the same information as the assign-fee transaction as well as the usage fee information. The credit-server is then responsible for running a clock.
- An End-charges transaction to end a charge for metered use. (In a variation on this approach, the repositories would exchange periodic charge information for each block of time.)
- A report-charges transaction between a personal credit server and a billing clearinghouse. This transaction is invoked at least once per billing period. It is used to pass along information about charges. On debit and credit cards, this transaction would also be used to update balance information and credit limits as needed.

All billing transactions are given a transaction ID and are reported to the credit severs by both the server and the client. This reduces possible loss of billing information if one of the parties to a transaction loses a banking card and

provides a check against tampering with the system.

Usage Transactions

After the session initiation transactions have been completed, the usage request may then be processed. To simplify the description of the steps carried out in processing a usage request, the term requester is used to refer to a repository in the requester mode which is initiating a request, and the term server is used to refer to a repository in the server mode and which contains the desired digital work. In many cases such as requests to print or view a work, the requester and server may be the same device and the transactions described in the following would be entirely internal. In such instances, certain transaction steps, such as the registration transaction, need not be performed.

There are some common steps that are part of the semantics of all of the usage rights transactions. These steps are referred to as the common transaction steps. There are two sets -- the "opening" steps and the "closing" steps. For simplicity, these are listed here rather than repeating them in the descriptions of all of the usage rights transactions.

Transactions can refer to a part of a digital work, a complete digital work, or a Digital work containing other digital works. Although not described in detail herein, a transaction may even refer to a folder comprised of a plurality of digital works. The term "work" is used to refer to what ever portion or set of digital works is being accessed.

Many of the steps here involve determining if certain conditions are satisfied. Recall that each usage right may have one or more conditions which must be satisfied before the right can be exercised. Digital works have parts and parts have parts. Different parts can have different rights and fees. Thus, it is necessary to verify that the requirements are met for ALL of the parts that are involved in a transaction For brevity, when reference is made to checking whether the rights exist and conditions for exercising are satisfied, it is meant that all such checking takes place for each of the relevant parts of the work.

Figure 18 illustrates the initial common opening and closing steps for a transaction. At this point it is assumed that registration has occurred and that a "trusted" session is in place. General tests are tests on usage rights associated with the folder containing the work or some containing folder higher in the file system hierarchy. These tests correspond to requirements imposed on the work as a consequence of its being on the particular repository, as opposed to being attached to the work itself. Referring to Figure 18, prior to initiating a usage transaction, the requester performs any general tests that are required before the right associated with the transaction can be exercised, step, 1801. For example, install, uninstall and delete rights may be implemented to require that a requester have an authorization certificate before the right can be exercised. Another example is the requirement that a digital ticket be present and punched before a digital work may be copied to a requester. If any of the general tests fail, the transaction is not initiated, step, 1802. Assuming that such required tests are passed, upon receiving the usage request, the server generates a transaction identifier that is used in records or reports of the transaction, step 1803. The server then checks whether the digital work has been granted the right corresponding to the requested transaction, step 1804. If the digital work has not been granted the right corresponding to the request, the transaction terminates, step 1805. If the digital work has been granted the requested right, the server then determines if the various conditions for exercising the right are satisfied. Time based conditions are examined, step 1806. These conditions are checked by examining the time specification for the the version of the right. If any of the conditions are not satisfied, the transaction terminates per step 1805.

Assuming that the time based conditions are satisfied, the server checks security and access conditions, step 1807. Such security and access conditions are satisfied if: 1) the requester is at the specified security class, or a higher security class, 2) the server satisfies any specified authorization test and 3) the requester satisfies any specified authorization tests and has any required digital tickets. If any of the conditions are not satisfied, the transaction terminates per step 1805.

Assuming that the security and access conditions are all satisfied, the server checks the copy count condition, step 1808. If the copy count equals zero, then the transaction cannot be completed and the transaction terminates per step 1805.

Assuming that the copy count does not equal zero, the server checks if the copies in use for the requested right is greater than or equal to any copy count for the requested right (or relevant parts), step 1809. If the copies in use is greater than or equal to the copy count, this indicates that usage rights for the version of the transaction have been exhausted. Accordingly, the server terminates the transaction, step 1805. If the copy count is less than the copies in use for the transaction the transaction can continue, and the copies in use would be incremented by the number of digital works requested in the transaction, step 1810.

The server then checks if the digital work has a "Loan" access right, step 1811. The "Loan" access right is a special case since remaining rights may be present even though all copies are loaned out. If the digital work has the "Loan" access right, a check is made to see if all copies have been loaned out, step 1812. The number of copies that could be loaned is the sum of the Copy-Counts for all of the versions of the loan right of the digital work. For a composite work, the relevant figure is the minimal such sum of each of the components of the composite work. If all copies have been loaned out, the remaining rights are determined, step 1813. The remaining-rights is determined from the remaining

rights specifications from the versions of the Loan right. If there is only one version of the Loan right, then the determination is simple. The remaining rights are the ones specified in that version of the Loan right, or none if Remaining-Rights: is not specified. If there are multiple versions of the Loan right and all copies of all of the versions are loaned out, then the remaining rights is taken as the minimum set (intersection) of remaining rights across all of the versions of the loan right. The server then determines if the requested right is in the set of remaining rights, step 1814. If the requested right is not in the set of remaining rights, the server terminates the transaction, step 1805.

If Loan is not a usage right for the digital work or if all copies have not been loaned out or the requested right is in the set of remaining rights, fee conditions for the right are then checked, step 1815. This will initiate various financial transactions between the repository and associated credit server. Further, any metering of usage of a digital work will commence. If any financial transaction fails, the transaction terminates per step 1805.

It should be noted that the order in which the conditions are checked need not follow the order of steps 1806-1815.

At this point, right specific steps are now performed and are represented here as step 1816. The right specific steps are described in greater detail below.

The common closing transaction steps are now performed. Each of the closing transaction steps are performed by the server after a successful completion of a transaction. Referring back to Figure 18, the copies in use value for the requested right is decremented by the number of copies involved in the transaction, step 1817. Next, if the right had a metered usage fee specification, the server subtracts the elapsed time from the Remaining-Use-Time associated with the right for every part involved in the transaction, step 1818. Finally, if there are fee specifications associated with the right, the server initiates End-Charge financial transaction to confirm billing, step 1819.

Transmission Protocol

An important area to consider is the transmission of the digital work from the server to the requester. The transmission protocol described herein refers to events occurring after a valid session has been created. The transmission protocol must handle the case of disruption in the communications between the repositories. It is assumed that interference such as injecting noise on the communication channel can be detected by the integrity checks (e.g., parity, checksum, etc.) that are built into the transport protocol and are not discussed in detail herein.

The underlying goal in the transmission protocol is to preclude certain failure modes, such as malicious or accidental interference on the communications channel. Suppose, for example, that a user pulls a card with the credit server at a specific time near the end of a transaction. There should not be a vulnerable time at which "pulling the card" causes the repositories to fail to correctly account for the number of copies of the work that have been created. Restated, there should be no time at which a party can break a connection as a means to avoid payment after using a digital work.

If a transaction is interrupted (and fails), both repositories restore the digital works and accounts to their state prior to the failure, modulo records of the failure itself.

Figure 19 is a state diagram showing steps in the process of transmitting information during a transaction. Each box represents a state of a repository in either the server mode (above the central dotted line 1901) or in the requester mode (below the dotted line 1901). Solid arrows stand for transitions between states. Dashed arrows stand for message communications between the repositories. A dashed message arrow pointing to a solid transition arrow is interpreted as meaning that the transition takes place when the message is received. Unlabeled transition arrows take place unconditionally. Other labels on state transition arrows describe conditions that trigger the transition.

Referring now to Figure 19, the server is initially in a state 1902 where a new transaction is initiated via start message 1903. This message includes transaction information including a transaction identifier and a count of the blocks of data to be transferred. The requester, initially in a wait state 1904 then enters a data wait state 1905.

The server enters a data transmit state 1906 and transmits a block of data 1907 and then enters a wait for acknowledgement state 1908. As the data is received, the requester enters a data receive state 1909 and when the data blocks are completely received it enters an acknowledgement state 1910 and transmits an Acknowledgement message 1911 to the server.

If there are more blocks to send, the server waits until receiving an Acknowledgement message from the requester. When an Acknowledgement message is received it sends the next block to the requester and again waits for acknowledgement. The requester also repeats the same cycle of states.

If the server detects a communications failure before sending the last block, it enters a cancellation state 1912 wherein the transaction is cancelled. Similarly, if the requester detects a communications failure before receiving the last block it enters a cancellation state 1913.

If there are no more blocks to send, the server commits to the transaction and waits for the final Acknowledgement in state 1914. If there is a communications failure before the server receives the final Acknowledgement message, it still commits to the transaction but includes a report about the event to its credit server in state 1915. This report serves two purposes. It will help legitimize any claims by a user of having been billed for receiving digital works that were not completely received. Also it helps to identify repositories and communications lines that have suspicious patterns of

use and interruption. The server then enters its completion state 1916.

On the requester side, when there are no more blocks to receive, the requester commits to the transaction in state 1917. If the requester detects a communications failure at this state, it reports the failure to its credit server in state 1918, but still commits to the transaction. When it has committed, it sends an acknowledgement message to the server. The server then enters its completion state 1919.

The key property is that both the server and the requester cancel a transaction if it is interrupted before all of the data blocks are delivered, and commits to it if all of the data blocks have been delivered.

There is a possibility that the server will have sent all of the data blocks (and committed) but the requester will not have received all of them and will cancel the transaction. In this case, both repositories will presumably detect a communications failure and report it to their credit server. This case will probably be rare since it depends on very precise timing of the communications failure. The only consequence will be that the user at the requester repository may want to request a refund from the credit services -- and the case for that refund will be documented by reports by both repositories.

To prevent loss of data, the server should not delete any transferred digital work until receiving the final acknowledgement from the requester. But it also should not use the file. A well known way to deal with this situation is called "two-phase commit" or 2PC.

Two-phase commit works as follows. The first phase works the same as the method described above. The server sends all of the data to the requester. Both repositories mark the transaction (and appropriate files) as uncommitted. The server sends a ready-to-commit message to the requester. The requester sends back an acknowledgement. The server then commits and sends the requester a commit message. When the requester receives the commit message, it commits the file.

If there is a communication failure or other crash, the requester must check back with the server to determine the status of the transaction. The server has the last word on this. The requester may have received all of the data, but if it did not get the final message, it has not committed. The server can go ahead and delete files (except for transaction records) once it commits, since the files are known to have been fully transmitted before starting the 2PC cycle.

There are variations known in the art which can be used to achieve the same effect. For example, the server could use an additional level of encryption when transmitting a work to a client. Only after the client sends a message acknowledging receipt does it send the key. The client then agrees to pay for the digital work. The point of this variation is that it provides a clear audit trail that the client received the work. For trusted systems, however, this variation adds a level of encryption for no real gain in accountability.

The transaction for specific usage rights are now discussed.

The Copy Transaction

A Copy transaction is a request to make one or more independent copies of the work with the same or lesser usage rights. Copy differs from the extraction right discussed later in that it refers to entire digital works or entire folders containing digital works. A copy operation cannot be used to remove a portion of a digital work.

- The requester sends the server a message to initiate the Copy Transaction. This message indicates the work to be copied, the version of the copy right to be used for the transaction, the destination address information (location in a folder) for placing the work, the file data for the work (including its size), and the number of copies requested.
- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the client according to the transmission protocol. If a Next-Set-Of-Rights has been provided in the version of the right, those rights are transmitted as the rights for the work. Otherwise, the rights of the original are transmitted. In any event, the Copy-Count field for the copy of the digital work being sent right is set to the number-of-copies requested.
- The requester records the work contents, data, and usage rights and stores the work. It records the date and time that the copy was made in the properties of the digital work.
- The repositories perform the common closing transaction steps.

The Transfer Transaction

A Transfer transaction is a request to move copies of the work with the same or lesser usage rights to another repository. In contrast with a copy transaction, this results in removing the work copies from the server.

- The requester sends the server a message to initiate the Transfer Transaction. This message indicates the work to be transferred, the version of the transfer right to be used in the transaction, the destination address information for placing the work, the file data for the work, and the number of copies involved.

- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the requester according to the transmission protocol. If a Next-Set-Of-Rights has been provided, those rights are transmitted as the rights for the work. Otherwise, the rights of the original are transmitted.

In either case, the Copy-Count field for the transmitted rights are set to the number-of-copies requested.

- The requester records the work contents, data, and usage rights and stores the work.
- The server decrements its copy count by the number of copies involved in the transaction.
- The repositories perform the common closing transaction steps.
- If the number of copies remaining in the server is now zero, it erases the digital work from its memory.

The Loan Transaction

A loan transaction is a mechanism for loaning copies of a digital work. The maximum duration of the loan is determined by an internal parameter of the digital work. Works are automatically returned after a predetermined time period.

- The requester sends the server a message to initiate the Transfer Transaction. This message indicates the work to be loaned, the version of the loan right to be used in the transaction, the destination address information for placing the work, the number of copies involved, the file data for the work, and the period of the loan.
- The server checks the validity of the requested loan period, and ends with an error if the period is not valid. Loans for a loaned copy cannot extend beyond the period of the original loan to the server.
- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the requester. If a Next-Set-Of-Rights has been provided, those rights are transmitted as the rights for the work. Otherwise, the rights of the original are transmitted, as modified to reflect the loan period.
- The requester records the digital work contents, data, usage rights, and loan period and stores the work.
- The server updates the usage rights information in the digital work to reflect the number of copies loaned out.
- The repositories perform the common closing transaction steps.
- The server updates the usage rights data for the digital work. This may preclude use of the work until it is returned from the loan. The user on the requester platform can now use the transferred copies of the digital work. A user accessing the original repository cannot use the digital work , unless there are copies remaining. What happens next depends on the order of events in time.

Case 1. If the time of the loan period is not yet exhausted and the requester sends the repository a Return message.

- The return message includes the requester identification, and the transaction ID.
- The server decrements the copies-in-use field by the number of copies that were returned. (If the number of digital works returned is greater than the number actually borrowed, this is treated as an error.) This step may now make the work available at the server for other users.
- The requester deactivates its copies and removes the contents from its memory.

Case 2. If the time of the loan period is exhausted and the requester has not yet sent a Return message.

- The server decrements the copies-in-use field by the number digital works that were borrowed.
- The requester automatically deactivates its copies of the digital work. It terminates all current uses and erases the digital work copies from memory. One question is why a requester would ever return a work earlier than the period of the loan, since it would be returned automatically anyway. One reason for early return is that there may be a metered fee which determines the cost of the loan. Returning early may reduce that fee.

The Play Transaction

A play transaction is a request to use the contents of a work. Typically, to "play" a work is to send the digital work through some kind of transducer, such as a speaker or a display device. The request implies the intention that the contents will not be communicated digitally to any other system. For example, they will not be sent to a printer, recorded on any digital medium, retained after the transaction or sent to another repository.

This term "play" is natural for examples like playing music, playing a movie, or playing a video game. The general

form of play means that a "player" is used to use the digital work. However, the term play covers all media and kinds of recordings. Thus one would "play" a digital work, meaning, to render it for reading, or play a computer program, meaning to execute it. For a digital ticket the player would be a digital ticket agent.

- The requester sends the server a message to initiate the play transaction. This message indicates the work to be played, the version of the play right to be used in the transaction, the identity of the player being used, and the file data for the work.
- The server checks the validity of the player identification and the compatibility of the player identification with the player specification in the right. It ends with an error if these are not satisfactory.
- The repositories perform the common opening transaction steps.
- The server and requester read and write the blocks of data as requested by the player according to the transmission protocol. The requester plays the work contents, using the player.
- When the player is finished, the player and the requester remove the contents from their memory.
- The repositories perform the common closing transaction steps.

The Print Transaction

A Print transaction is a request to obtain the contents of a work for the purpose of rendering them on a "printer." We use the term "printer" to include the common case of writing with ink on paper. However, the key aspect of "printing" in our use of the term is that it makes a copy of the digital work in a place outside of the protection of usage rights. As with all rights, this may require particular authorization certificates.

Once a digital work is printed, the publisher and user are bound by whatever copyright laws are in effect. However, printing moves the contents outside the control of repositories. For example, absent any other enforcement mechanisms, once a digital work is printed on paper, it can be copied on ordinary photocopying machines without intervention by a repository to collect usage fees. If the printer to a digital disk is permitted, then that digital copy is outside of the control of usage rights. Both the creator and the user know this, although the creator does not necessarily give tacit consent to such copying, which may violate copyright laws.

- The requester sends the server a message to initiate a Print transaction. This message indicates the work to be played, the identity of the printer being used, the file data for the work, and the number of copies in the request.
- The server checks the validity of the printer identification and the compatibility of the printer identification with the printer specification in the right. It ends with an error if these are not satisfactory.
- The repositories perform the common opening transaction steps.
- The server transmits blocks of data according to the transmission protocol.
- The requester prints the work contents, using the printer.
- When the printer is finished, the printer and the requester remove the contents from their memory.
- The repositories perform the common closing transaction steps.

The Backup Transaction

A Backup transaction is a request to make a backup copy of a digital work, as a protection against media failure. In the context of repositories, secure backup copies differ from other copies in three ways: (1) they are made under the control of a Backup transaction rather than a Copy transaction, (2) they do not count as regular copies, and (3) they are not usable as regular copies. Generally, backup copies are encrypted.

Although backup copies may be transferred or copied, depending on their assigned rights, the only way to make them useful for playing, printing or embedding is to restore them.

The output of a Backup operation is both an encrypted data file that contains the contents and description of a work, and a restoration file with an encryption key for restoring the encrypted contents. In many cases, the encrypted data file would have rights for "printing" it to a disk outside of the protection system, relying just on its encryption for security. Such files could be stored anywhere that was physically safe and convenient. The restoration file would be held in the repository. This file is necessary for the restoration of a backup copy. It may have rights for transfer between repositories.

- The requester sends the server a message to initiate a backup transaction. This message indicates the work to be backed up, the version of the backup right to be used in the transaction, the destination address information for placing the backup copy, the file data for the work.
- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the requester. If a Next-Set-Of-Rights has been provided,

those rights are transmitted as the rights for the work. Otherwise, a set of default rights for backup files of the original are transmitted by the server.

- The requester records the work contents, data, and usage rights. It then creates a one-time key and encrypts the contents file. It saves the key information in a restoration file.
- The repositories perform the common closing transaction steps.

In some cases, it is convenient to be able to archive the large, encrypted contents file to secure offline storage, such as a magneto-optical storage system or magnetic tape. This creation of a non-repository archive file is as secure as the encryption process. Such non-repository archive storage is considered a form of "printing" and is controlled by a print right with a specified "archive-printer." An archive-printer device is programmed to save the encrypted contents file (but not the description file) offline in such a way that it can be retrieved.

The Restore Transaction

A Restore transaction is a request to convert an encrypted backup copy of a digital work into a usable copy. A restore operation is intended to be used to compensate for catastrophic media failure. Like all usage rights, restoration rights can include fees and access tests including authorization checks.

- The requester sends the server a message to initiate a Restore transaction. This message indicates the work to be restored, the version of the restore right for the transaction, the destination address information for placing the work, and the file data for the work.
- The server verifies that the contents file is available (i.e. a digital work corresponding to the request has been backed-up.) If it is not, it ends the transaction with an error.
- The repositories perform the common opening transaction steps.
- The server retrieves the key from the restoration file. It decrypts the work contents, data, and usage rights.
- The server transmits the requested contents and data to the requester according to the transmission protocol. If a Next-Set-Of-Rights has been provided, those rights are transmitted as the rights for the work. Otherwise, a set of default rights for backup files of the original are transmitted by the server.
- The requester stores the digital work.
- The repositories perform the common closing transaction steps.

The Delete Transaction

A Delete transaction deletes a digital work or a number of copies of a digital work from a repository. Practically all digital works would have delete rights.

- The requester sends the server a message to initiate a delete transaction. This message indicates the work to be deleted, the version of the delete right for the transaction.
- The repositories perform the common opening transaction steps.
- The server deletes the file, erasing it from the file system.
- The repositories perform the common closing transaction steps.

The Directory Transaction

A Directory transaction is a request for information about folders, digital works, and their parts. This amounts to roughly the same idea as protection codes in a conventional file system like TENEX, except that it is generalized to the full power of the access specifications of the usage rights language.

The Directory transaction has the important role of passing along descriptions of the rights and fees associated with a digital work. When a user wants to exercise a right, the user interface of his repository implicitly makes a directory request to determine the versions of the right that are available. Typically these are presented to the user -- such as with different choices of billing for exercising a right. Thus, many directory transactions are invisible to the user and are exercised as part of the normal process of exercising all rights.

- The requester sends the server a message to initiate a Directory transaction. This message indicates the file or folder that is the root of the directory request and the version of the directory right used for the transaction.
- The server verifies that the information is accessible to the requester. In particular, it does not return the names of any files that have a HIDE-NAME status in their directory specifications, and it does not return the parts of any folders or files that have HIDE-PARTS in their specification. If the information is not accessible, the server ends

the transaction with an error.

- The repositories perform the common opening transaction steps.
- The server sends the requested data to the requester according to the transmission protocol.
- The requester records the data.
- The repositories perform the common closing transaction steps.

The Folder Transaction

A Folder transaction is a request to create or rename a folder, or to move a work between folders. Together with Directory rights, Folder rights control the degree to which organization of a repository can be accessed or modified from another repository.

- The requester sends the server a message to initiate a Folder transaction. This message indicates the folder that is the root of the folder request, the version of the folder right for the transaction, an operation, and data. The operation can be one of create, rename, and move file. The data are the specifications required for the operation, such as a specification of a folder or digital work and a name.
- The repositories perform the common opening transaction steps.
- The server performs the requested operation -- creating a folder, renaming a folder, or moving a work between folders.
- The repositories perform the common closing transaction steps.

The Extract Transaction

A extract transaction is a request to copy a part of a digital work and to create a new work containing it. The extraction operation differs from copying in that it can be used to separate a part of a digital work from d-blocks or shells that place additional restrictions or fees on it. The extraction operation differs from the edit operation in that it does not change the contents of a work, only its embedding in d-blocks. Extraction creates a new digital work.

- The requester sends the server a message to initiate an Extract transaction. This message indicates the part of the work to be extracted, the version of the extract right to be used in the transaction, the destination address information for placing the part as a new work, the file data for the work, and the number of copies involved.
- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the requester according to the transmission protocol. If a Next-Set-Of-Rights has been provided, those rights are transmitted as the rights for the new work. Otherwise, the rights of the original are transmitted. The Copy-Count field for this right is set to the number-of-copies requested.
- The requester records the contents, data, and usage rights and stores the work. It records the date and time that new work was made in the properties of the work.
- The repositories perform the common closing transaction steps.

The Embed Transaction

An embed transaction is a request to make a digital work become a part of another digital work or to add a shell d-block to enable the adding of fees by a distributor of the work.

- The requester sends the server a message to initiate an Embed transaction. This message indicates the work to be embedded, the version of the embed right to be used in the transaction, the destination address information for placing the part as a a work, the file data for the work, and the number of copies involved.
- The server checks the control specifications for all of the rights in the part and the destination. If they are incompatible, the server ends the transaction with an error.
- The repositories perform the common opening transaction steps.
- The server transmits the requested contents and data to the requester according to the transmission protocol. If a Next-Set-Of-Rights has been provided, those rights are transmitted as the rights for the new work. Otherwise, the rights of the original are transmitted. The Copy-Count field for this right is set to the number-of-copies requested.
- The requester records the contents, data, and usage rights and embeds the work in the destination file.
- The repositories perform the common closing transaction steps.

The Edit Transaction

An Edit transaction is a request to make a new digital work by copying, selecting and modifying portions of an existing digital work. This operation can actually change the contents of a digital work. The kinds of changes that are permitted depend on the process being used. Like the extraction operation, edit operates on portions of a digital work. In contrast with the extract operation, edit does not affect the rights or location of the work. It only changes the contents. The kinds of changes permitted are determined by the type specification of the processor specified in the rights. In the currently preferred embodiment, an edit transaction changes the work itself and does not make a new work. However, it would be a reasonable variation to cause a new copy of the work to be made.

- The requester sends the server a message to initiate an Edit transaction. This message indicates the work to be edited, the version of the edit right to be used in the transaction, the file data for the work (including its size), the process-ID for the process, and the number of copies involved.
- The server checks the compatibility of the process-ID to be used by the requester against any process-ID specification in the right. If they are incompatible, it ends the transaction with an error.
- The repositories perform the common opening transaction steps.
- The requester uses the process to change the contents of the digital work as desired. (For example, it can select and duplicate parts of it; combine it with other information; or compute functions based on the information. This can amount to editing text, music, or pictures or taking whatever other steps are useful in creating a derivative work.)
- The repositories perform the common closing transaction steps.

The edit transaction is used to cover a wide range of kinds of works. The category describes a process that takes as its input any portion of a digital work and then modifies the input in some way. For example, for text, a process for editing the text would require edit rights. A process for "summarizing" or counting words in the text would also be considered editing. For a music file, processing could involve changing the pitch or tempo, or adding reverberations, or any other audio effect. For digital video works, anything which alters the image would require edit rights. Examples would be colorizing, scaling, extracting still photos, selecting and combining frames into story boards, sharpening with signal processing, and so on.

Some creators may want to protect the authenticity of their works by limiting the kinds of processes that can be performed on them. If there are no edit rights, then no processing is allowed at all. A processor identifier can be included to specify what kind of process is allowed. If no process identifier is specified, then arbitrary processors can be used. For an example of a specific process, a photographer may want to allow use of his photograph but may not want it to be colorized. A musician may want to allow extraction of portions of his work but not changing of the tonality.

Authorization Transactions

There are many ways that authorization transactions can be defined. In the following, our preferred way is to simply define them in terms of other transactions that we already need for repositories. Thus, it is convenient sometimes to speak of "authorization transactions," but they are actually made up of other transactions that repositories already have.

A usage right can specify an authorization-ID, which identifies an authorization object (a digital work in a file of a standard format) that the repository must have and which it must process. The authorization is given to the generic authorization (or ticket) server of the repository which begins to interpret the authorization.

As described earlier, the authorization contains a server identifier, which may just be the generic authorization server or it may be another server. When a remote authorization server is required, it must contain a digital address. It may also contain a digital certificate.

If a remote authorization server is required, then the authorization process first performs the following steps:

- The generic authorization server attempts to set up the communications channel. (If the channel cannot be set up, then authorization fails with an error.)
- When the channel is set up, it performs a registration process with the remote repository. (If registration fails, then the authorization fails with an error.)
- When registration is complete, the generic authorization server invokes a "Play" transaction with the remote repository, supplying the authorization document as the digital work to be played, and the remote authorization server (a program) as the "player." (If the player cannot be found or has some other error, then the authorization fails with an error.)
- The authorization server then "plays" the authorization. This involves decrypting it using either the public key of the master repository that issued the certificate or the session key from the repository that transmitted it. The authorization server then performs various tests. These tests vary according to the authorization server. They

include such steps as checking issue and validity dates of the authorization and checking any hot-lists of known invalid authorizations. The authorization server may require carrying out any other transactions on the repository as well, such as checking directories, getting some person to supply a password, or playing some other digital work. It may also invoke some special process for checking information about locations or recent events. The "script" for such steps is contained within the authorization server.

- If all of the required steps are completed satisfactorily, the authorization server completes the transaction normally, signaling that authorization is granted.

The Install Transaction

An Install transaction is a request to install a digital work as runnable software on a repository. In a typical case, the requester repository is a rendering repository and the software would be a new kind or new version of a player. Also in a typical case, the software would be copied to file system of the requester repository before it is installed.

- The requester sends the server an Install message. This message indicates the work to be installed, the version of the Install right being invoked, and the file data for the work (including its size).
- The repositories perform the common opening transaction steps.
- The requester extracts a copy of the digital certificate for the software. If the certificate cannot be found or the master repository for the certificate is not known to the requester, the transaction ends with an error.
- The requester decrypts the digital certificate using the public key of the master repository, recording the identity of the supplier and creator, a key for decrypting the software, the compatibility information, and a tamper-checking code. (This step certifies the software.)
- The requester decrypts the software using the key from the certificate and computes a check code on it using a 1-way hash function. If the check-code does not match the tamper-checking code from the certificate, the installation transaction ends with an error. (This step assures that the contents of the software, including the various scripts, have not been tampered with.)
- The requester retrieves the instructions in the compatibility-checking script and follows them. If the software is not compatible with the repository, the installation transaction ends with an error. (This step checks platform compatibility.)
- The requester retrieves the instructions in the installation script and follows them. If there is an error in this process (such as insufficient resources), then the transaction ends with an error. Note that the installation process puts the runnable software in a place in the repository where it is no longer accessible as a work for exercising any usage rights other than the execution of the software as part of repository operations in carrying out other transactions.
- The repositories perform the common closing transaction steps.

The Uninstall Transaction

An Uninstall transaction is a request to remove software from a repository. Since uncontrolled or incorrect removal of software from a repository could compromise its behavioral integrity, this step is controlled.

- The requester sends the server an Uninstall message. This message indicates the work to be uninstalled, the version of the Uninstall right being invoked, and the file data for the work (including its size).
- The repositories perform the common opening transaction steps.
- The requester extracts a copy of the digital certificate for the software. If the certificate cannot be found or the master repository for the certificate is not known to the requester, the transaction ends with an error.
- The requester checks whether the software is installed. If the software is not installed, the transaction ends with an error.
- The requester decrypts the digital certificate using the public key of the master repository, recording the identity of the supplier and creator, a key for decrypting the software, the compatibility information, and a tamper-checking code. (This step authenticates the certification of the software, including the script for uninstalling it.)
- The requester decrypts the software using the key from the certificate and computes a check code on it using a 1-way hash function. If the check-code does not match the tamper-checking code from the certificate, the installation transaction ends with an error. (This step assures that the contents of the software, including the various scripts, have not been tampered with.)
- The requester retrieves the instructions in the uninstallation script and follows them. If there is an error in this process (such as insufficient resources), then the transaction ends with an error.
- The repositories perform the common closing transaction steps.

Claims

1. A system for secure distribution and control of digital works between repositories comprising:

   means for creating usage rights, each instance of a usage right representing a specific instance of how a digital work may be used or distributed;
   means for attaching a created set of usage rights to a digital work;
   a communications medium for coupling repositories to enable exchange of repository transaction messages;
   a plurality of general repositories for storing and securely exchanging digital works with attached usage rights, each of said general repositories comprising:
   a storage means for storing digital works and their attached usage rights;
   an identification certificate for indicating that the associated general repository is secure;
   an external interface for removably coupling to said communications medium;
   a session initiation transaction processing means for establishing a secure and trusted session with another repository, said session initiation transaction processing means using said identification certificate;
   a usage transaction processing means having a requester mode of operation for generating usage repository transaction messages to request access to digital works stored in another general repository, said usage repository transaction message specifying a usage right, said usage transaction processing means further having a server mode of operation for determining if a request for access to a digital work stored in said storage means may be granted, said request being granted only if the usage right specified in said request is attached to said digital work; and
   an input means coupled to said usage transaction processing means for enabling user created signals to cause generation of a usage repository transaction message to request access to digital works.

2. The system as recited in Claim 1 further comprising a rendering system, said rendering system comprising:

   a rendering repository for securely accessing digital works from a general repository, said rendering repository comprising;
   a storage means for storing digital works and their attached usage rights;
   an identification certificate, said identification certificate for indicating that the rendering repository is secure;
   an external interface for removably coupling to said communications medium;
   a session initiation transaction processing means for establishing a secure and trusted session with a general repository, said session initiation transaction processing means using said identification certificate;
   a usage transaction processing means for generating usage repository transaction messages to request access to digital works stored in a general repository, said usage repository transaction message specifying a usage right;
   an input means coupled to said usage transaction processing means for enabling user created signals to cause generation of usage repository transaction messages to request access to digital works;
   a rendering device for rendering digital works.

3. The system as recited in Claim 1 wherein said means for creating usage rights is further for the specification of different sets of usage rights to be attached to digital works when a corresponding usage right is exercised.

4. The system as recited in Claim 1 wherein said usage rights grammar further defines means for specifying conditions which must be satisfied before a usage right may be exercised and said usage transaction processing means in said server mode is further comprised of means for determining if specified conditions for a usage right are satisfied before access is granted.

5. The system as recited in Claim 1 wherein a first usage right enables copying of a digital work and specification of a revenue owner who is paid a fee whenever a copy of said digital work is made.

6. A method for controlling distribution and use of digital works comprising the steps of:

   a) attaching a set of usage rights to a digital work, each of said usage rights defining a specific instance of how a digital work may be used or distributed, said usage right specifying one or more conditions which must be satisfied in order for said usage right to be exercised and a next set of usage rights to be attached to a distributed digital work;
   b) storing said digital work and its attached usage rights in a first repository;

c) a second repository initiating a request to access said digital work in said first repository, said request identifying a usage right representing how said second repository desires to use said digital work;
d) said first repository receiving said request from said second repository;
e) said first repository determining if the identified usage right is attached to said digital work;
f) said first repository denying access to said digital work if said identified usage right is not attached to said digital work;
g) if said identified usage right is attached to said digital work, said first repository determining if conditions specified by said usage right are satisfied;
h) if said conditions are not satisfied, said first repository denying access to said digital work;
i) if said conditions are satisfied, said first repository attaching a next set of usage rights to said digital work, said next set of usage rights specifying how said second repository may use and distribute said digital work; and
j) said first repository transmitting said digital work and said attached next set of usage rights to said second repository.

7. The method as recited in Claim 6 wherein said step of a second repository initiating a request to access said digital work in said first repository is further comprised of the steps of:

c1) said second repository initiating establishment of a trusted session with said first repository;
c2) said first repository performing a set of registration transaction steps with said second repository, successful completion of said set of registration transaction steps indicating that said first repository is a trusted repository;
c3) said second repository performing said set of registration transaction steps with said first repository, successful completion of said set of registration transaction steps indicating that said second repository is a trusted repository;
c4) if said first repository and said second repository each successfully complete said set of registration steps, said first and second repository exchanging session encryption and decryption keys for secure transmission of subsequent communications between said first and second repository; and
c5) if said first repository or said second repository cannot successfully complete said set of registration transaction steps, terminating said session.

8. A system for controlling distribution and use of digital works comprising:

means for attaching usage rights to said digital work, said usage rights indicating how a recipient may use and and subsequently distribute said digital work;
a communications medium for coupling repositories to enable distribution of digital works;
a plurality of repositories for managing exchange of digital works based on usage rights attached to said digital works, each of said plurality of repositories comprising:
a storage means for storing digital works and their attached usage rights;
a processor operating responsive to coded instructions;
a memory means coupled to said processor for storing coded instruction to enable said processor to operate in a first server mode for processing access requests to digital works and for attaching usage rights to digital works when transmitted to another of said plurality of repositories, a second requester mode for initiating requests to access digital works, and a session initiation mode for establishing a trusted session with another of said plurality of repositories over said communications medium;
a clock;
a repository interface for coupling to said communications medium.

9. The system as recited in Claim 8 further comprising a plurality of rendering systems for rendering of digital works, each of said rendering systems comprising:

a repository for secure receipt of a digital work; and
a rendering device having means for converting digital works to signals suitable for rendering of said digital works.

10. A method for secure access of digital works stored on a server repository, said digital works having associated therewith one or more usage rights for specifying how said digital work may be used or distributed, said method comprising the steps of:

a) a requesting repository performing a first registration transaction with a server repository, said first regis-

tration transaction for establishing to said server repository that said requesting repository is trustworthy;
b) concurrently with step a), said server repository responding with a second registration transaction, said second registration transaction for establishing to said requesting repository that said server repository is trustworthy;
c) if either said first registration transaction or said second registration transaction fails, said server repository denying access to said digital work;
d) if said first registration transaction and said second registration transaction are successful, said requesting repository initiating a usage transaction with respect to a digital work stored in said server repository, said usage transaction indicating a request to access a digital work and specifying a particular usage right;
e) determining if said usage transaction may be completed by comparing said particular usage right specified in said usage transaction and usage rights associated with said digital work;
f) if said particular usage right is not one of said usage rights associated with said digital work, denying access to said digital work; and
g) if said particular usage right is one of said usage rights associated with said digital work, granting access to said digital work and performing usage transaction steps associated with said particular usage right.

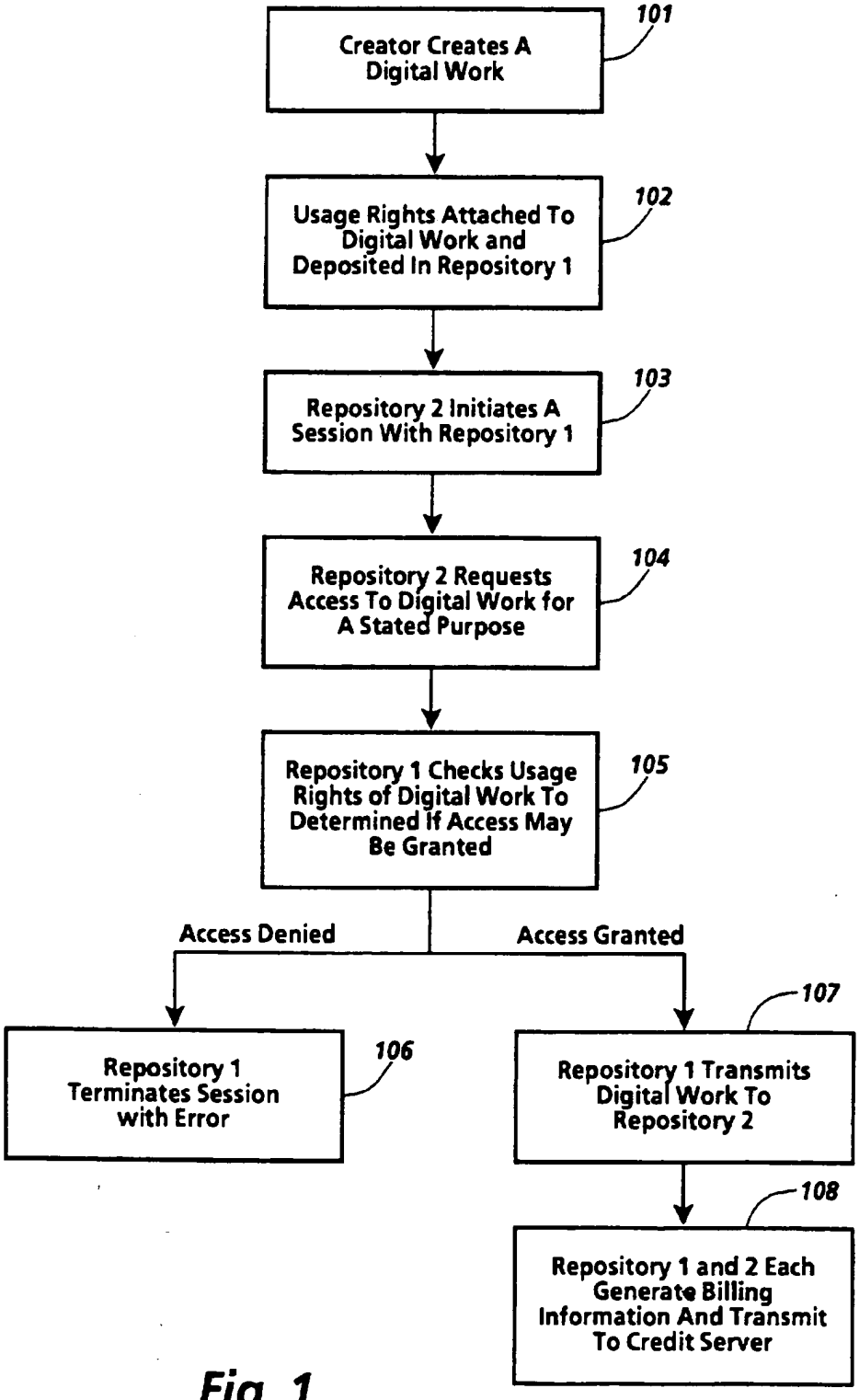


Fig. 1

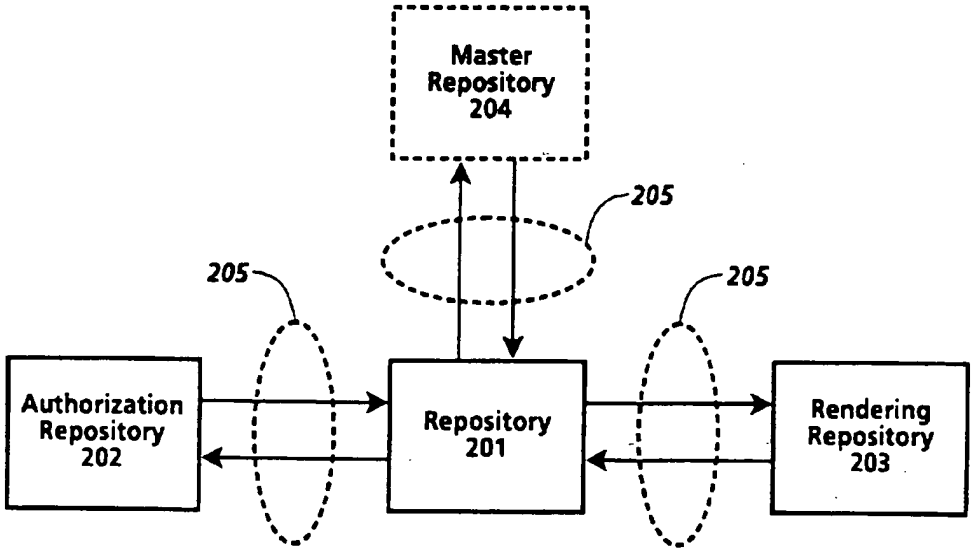


Fig. 2

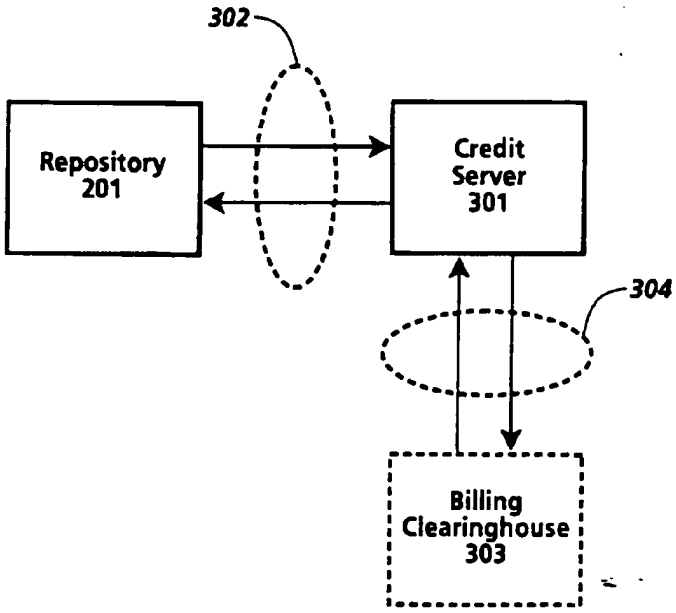


Fig. 3

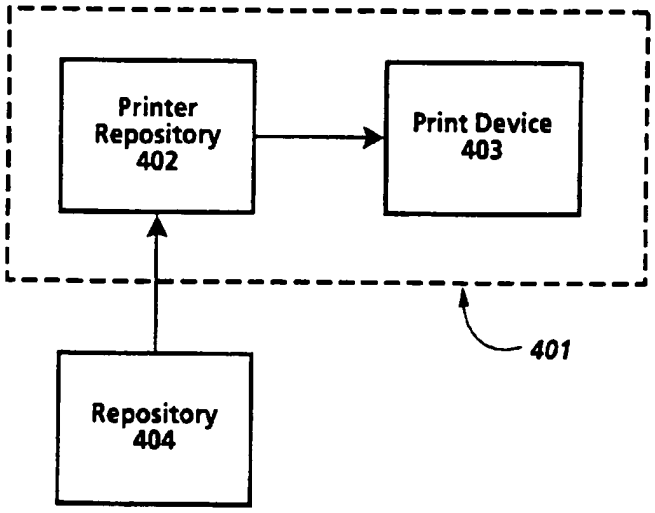


Fig. 4a

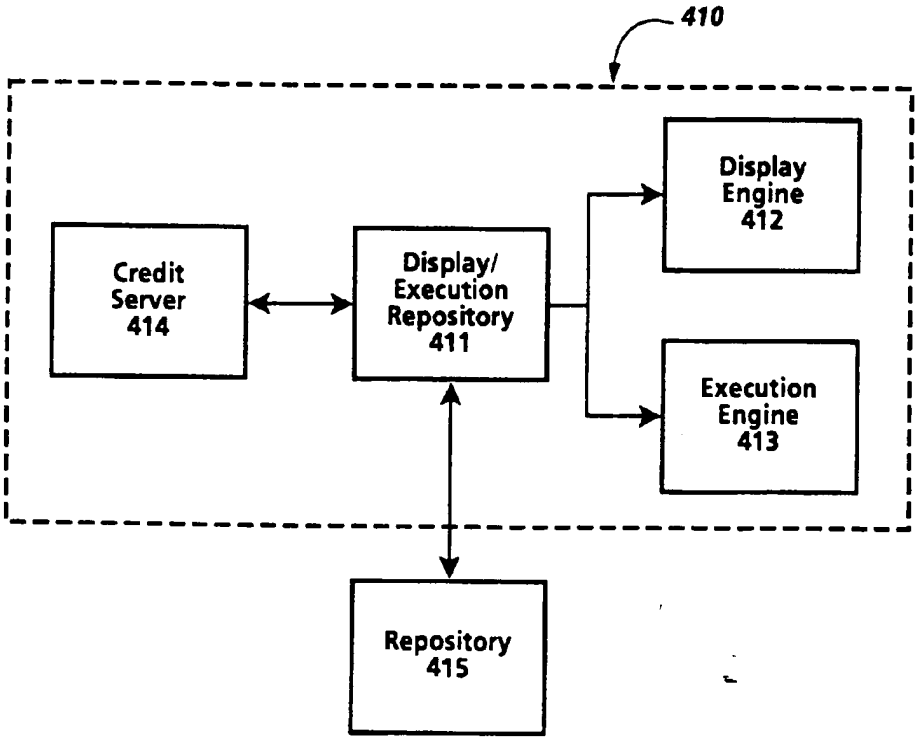


Fig. 4b

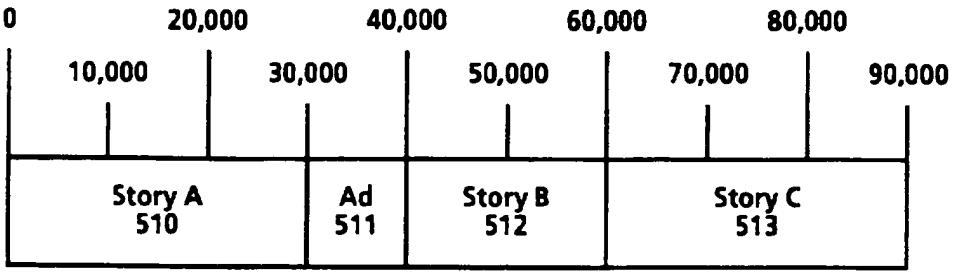


Fig. 5

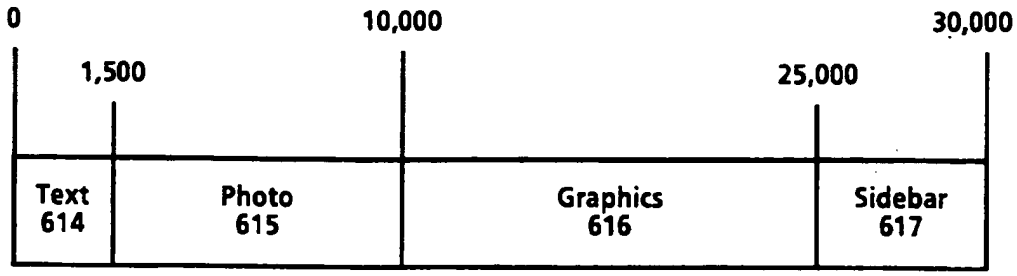


Fig. 6

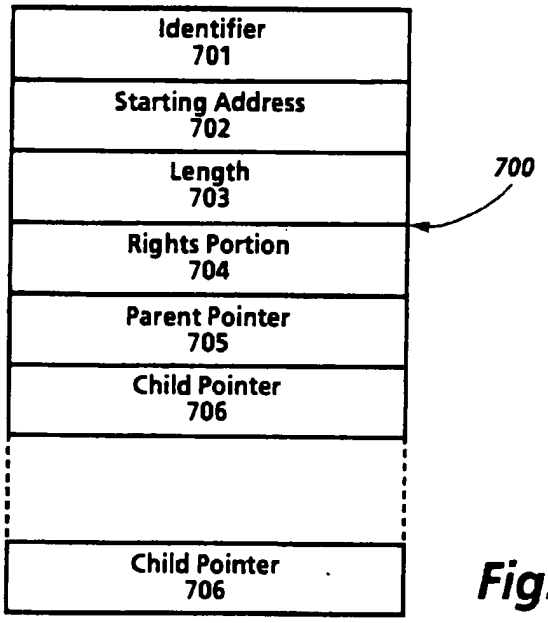


Fig. 7

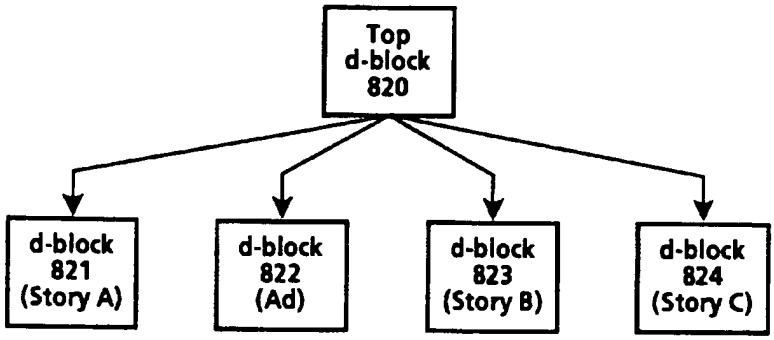


Fig. 8

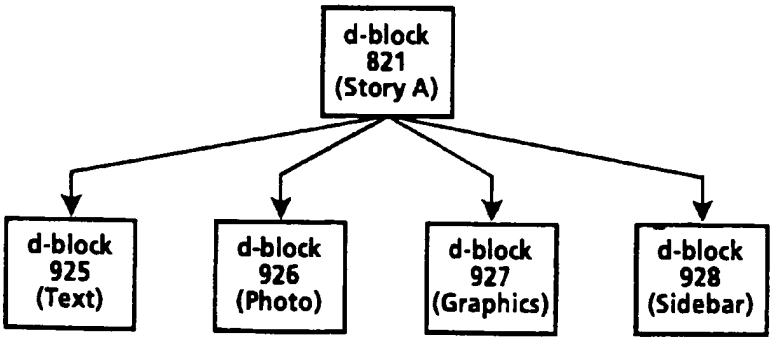


Fig. 9

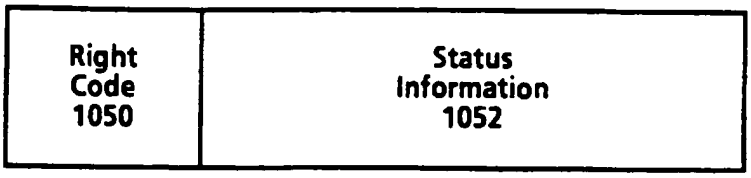


Fig.10

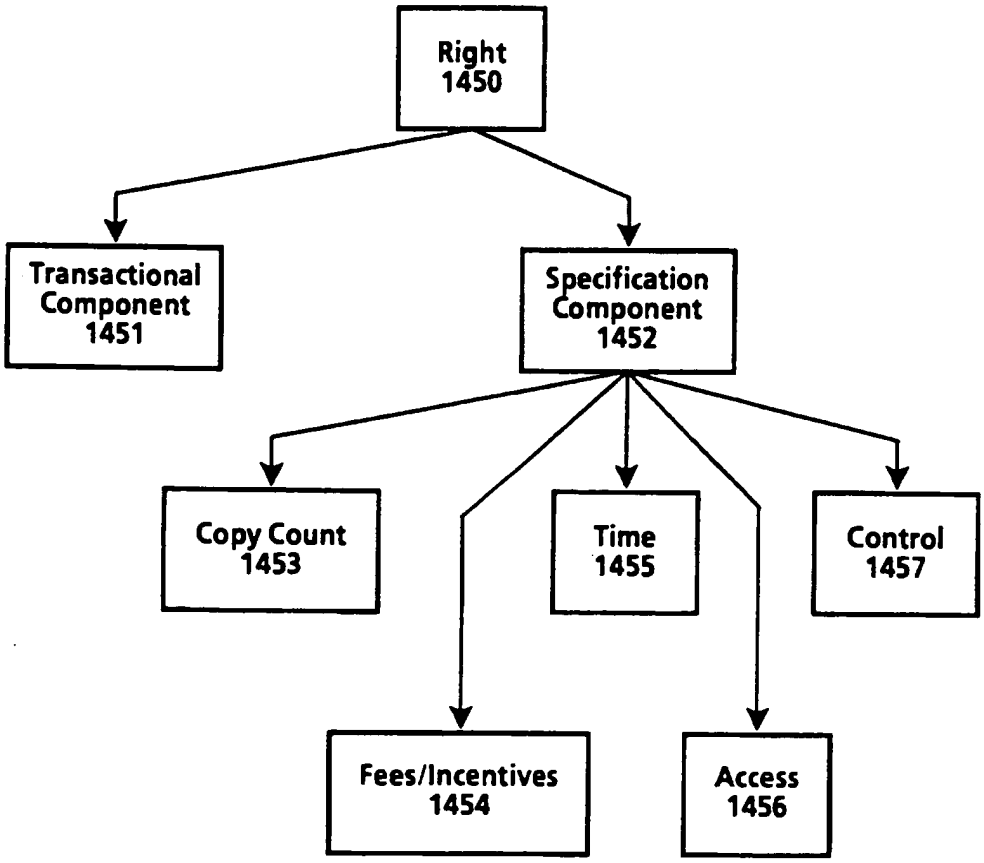


Fig.14

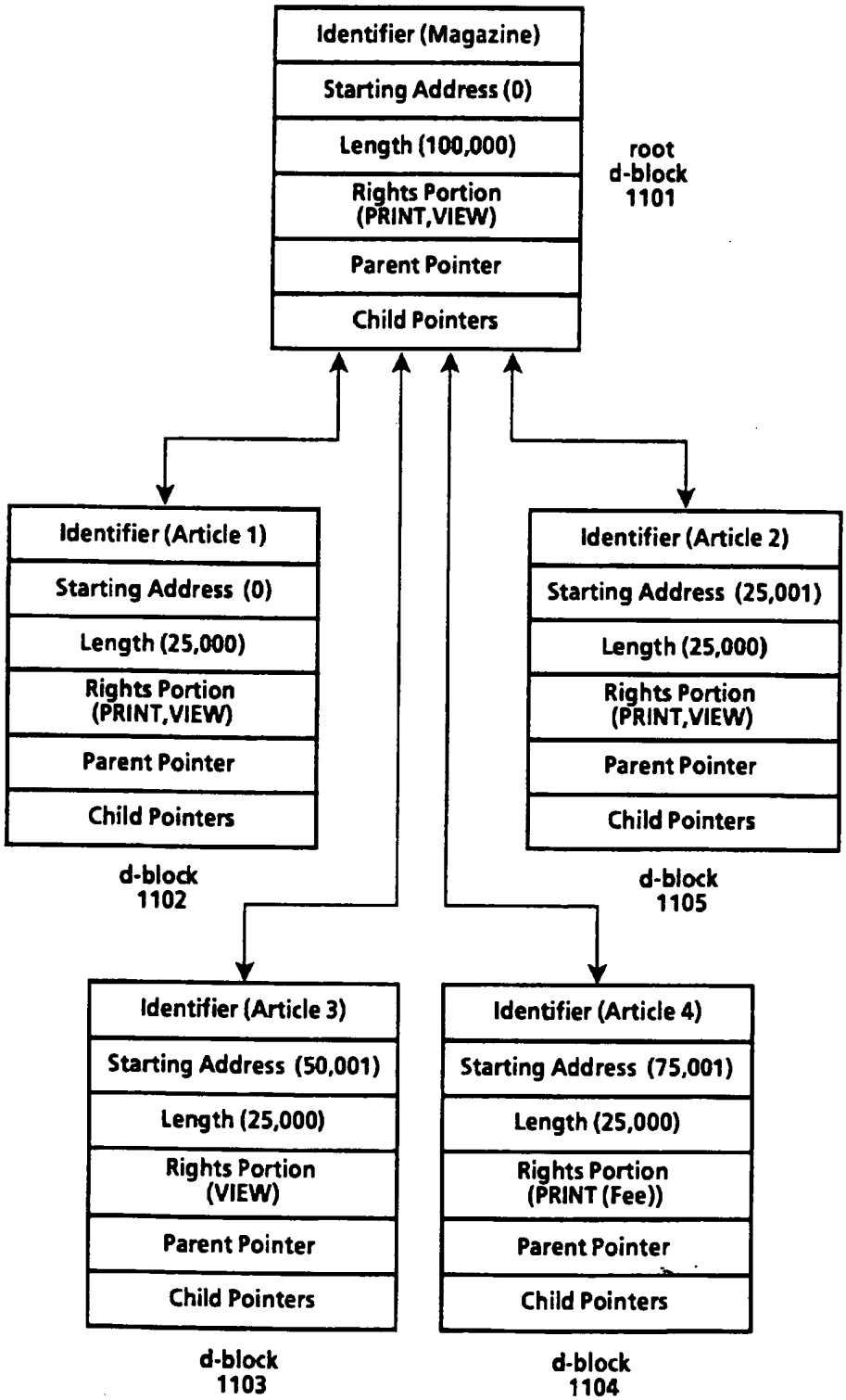


Fig.11

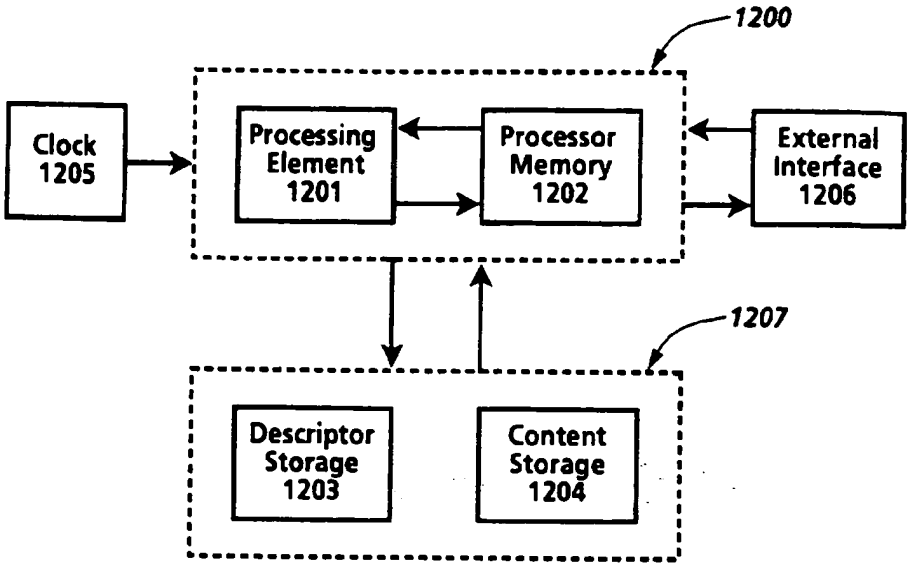


Fig.12

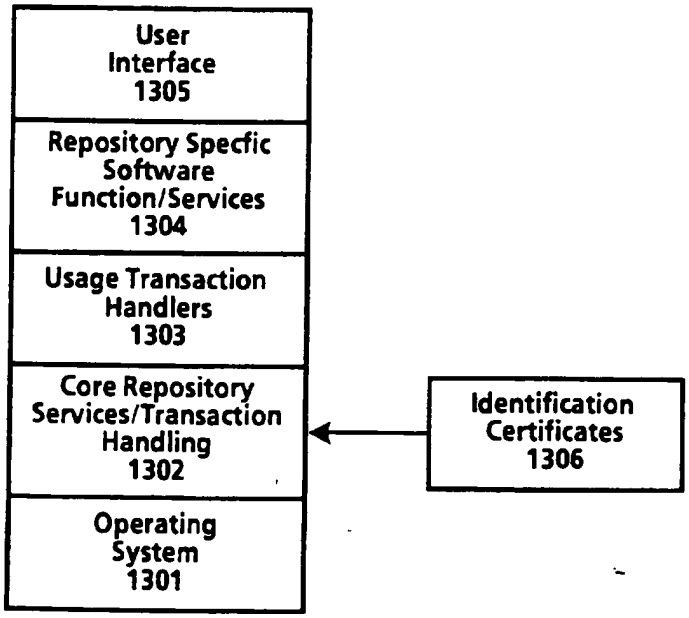


Fig.13

```
1501 ~Digital Work Rights:= (Rights*)
1502 ~Right := (Right-Code {Copy-Count} {Control-Spec} {Time-Spec}
        {Access-Spec} {Fee-Spec})
   1503 ~Right-Code := Render-Code | Transport-Code | File-Management-
          Code| Derivative-Works- Code | Configuration-Code
      1504 ~Render-Code := [ Play : {Player: Player-ID} | Print: {Printer: Printer-ID}]
      1505 ~Transport-Code := [Copy | Transfer | Loan {Remaining-Rights:
           Next-Set-of-Rights}]{(Next-Copy-Rights: Next-Set-of-Rights)}
      1506 ~File-Management-Code := Backup {Back-Up-Copy-Rights:
                                    Next-Set-of-Rights} | Restore | Delete | Folder
                                    | Directory {Name: Hide-Local | Hide-Remote}
                                    {Parts: Hide-Local | Hide-Remote}
      1507 ~Derivative-Works-Code := [Extract | Embed | Edit{Process:
                                    Process-ID}] {Next-Copy-Rights :
                                    Next-Set-of Rights}
      1508 ~Configuration-Code := Install | Uninstall
      1509 ~Next-Set-of-Rights := {(Add: Set-Of-Rights)} {(Delete:
           Set-Of-Rights)} {(Replace: Set-Of-Rights )}{(Keep: Set-Of-Rights )}
1510 ~Copy-Count := (Copies:positive-integer | 0 | Unlimited)
1511 ~Control-Spec := (Control: {Restrictable | Unrestrictable}
                      {Unchargeable | Chargeable})
1512 ~Time-Spec := ({Fixed-Interval | Sliding-Interval | Meter-Time}
                   Until: Expiration-Date)
   1513 ~Fixed-Interval := From: Start-Time
   1514 ~Sliding-Interval := Interval: Use-Duration
   1515 ~Meter-Time:= Time-Remaining: Remaining-Use
1516 ~Access-Spec := ({SC: Security-Class} {Authorization: Authorization-ID*}
        {Other-Authorization: Authorization-ID*} {Ticket: Ticket-ID})
1517 ~Fee-Spec:= {Scheduled-Discount} Regular-Fee-Spec | Scheduled-Fee-Spec |
                Markup-Spec
   1518 ~Scheduled-Discount:= Scheduled-Discount: (Scheduled-Discount:
                              (Time-Spec Percentage)*)
   1519 ~Regular-Fee-Spec := ({Fee: | Incentive:} [Per-Use-Spec | Metered-Rate-
                             Spec | Best-Price-Spec | Call-For-Price-Spec]
                             {Min: Money-Unit Per: Time-Spec}{Max:
                             Money-Unit Per: Time-Spec} To: Account-ID)
      1520 ~Per-Use-Spec:= Per-Use: Money-unit
      1521 ~Metered-Rate-Spec := Metered: Money-Unit Per: Time-Spec
      1522 ~Best-Price-Spec := Best-Price: Money-unit Max: Money-unit
      1523 ~Call-For-Price-Spec := Call-For -Price
   1524 ~Scheduled-Fee-Spec:= (Schedule: (Time-Spec Regular-Fee-Spec)* )
   1525 ~Markup-Spec:= Markup: percentage To: Account-ID
```

Fig.15

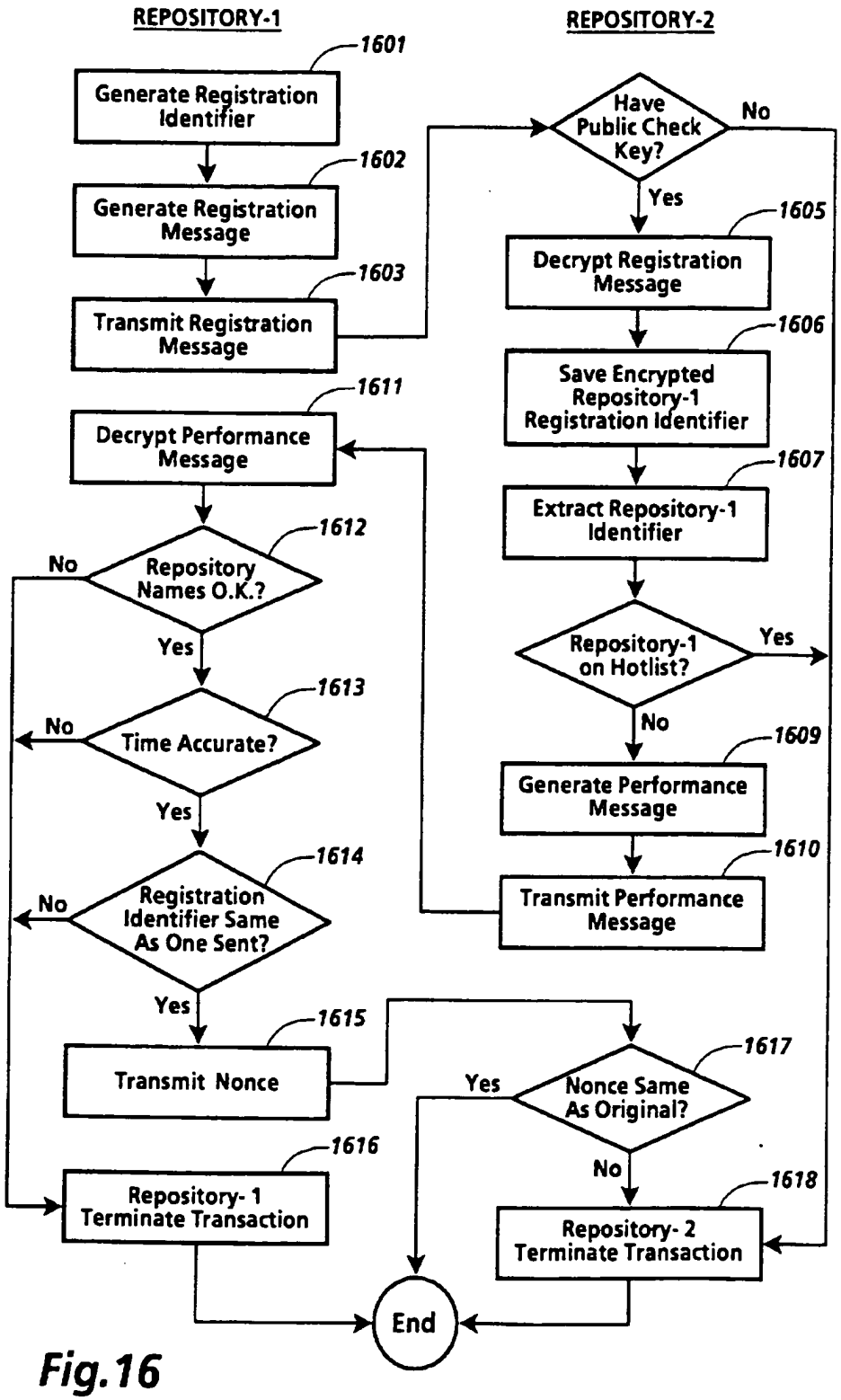


Fig.16

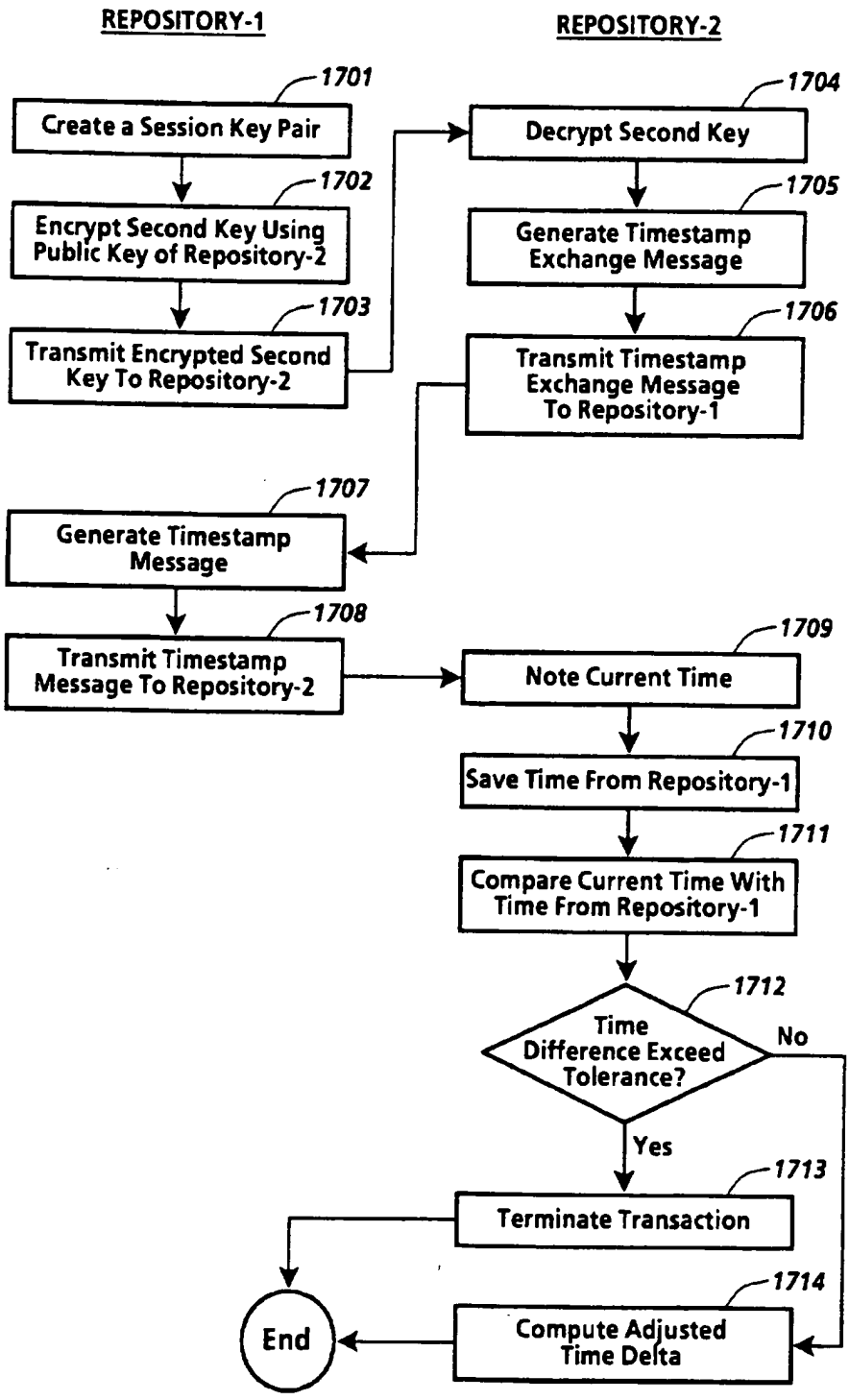


Fig.17

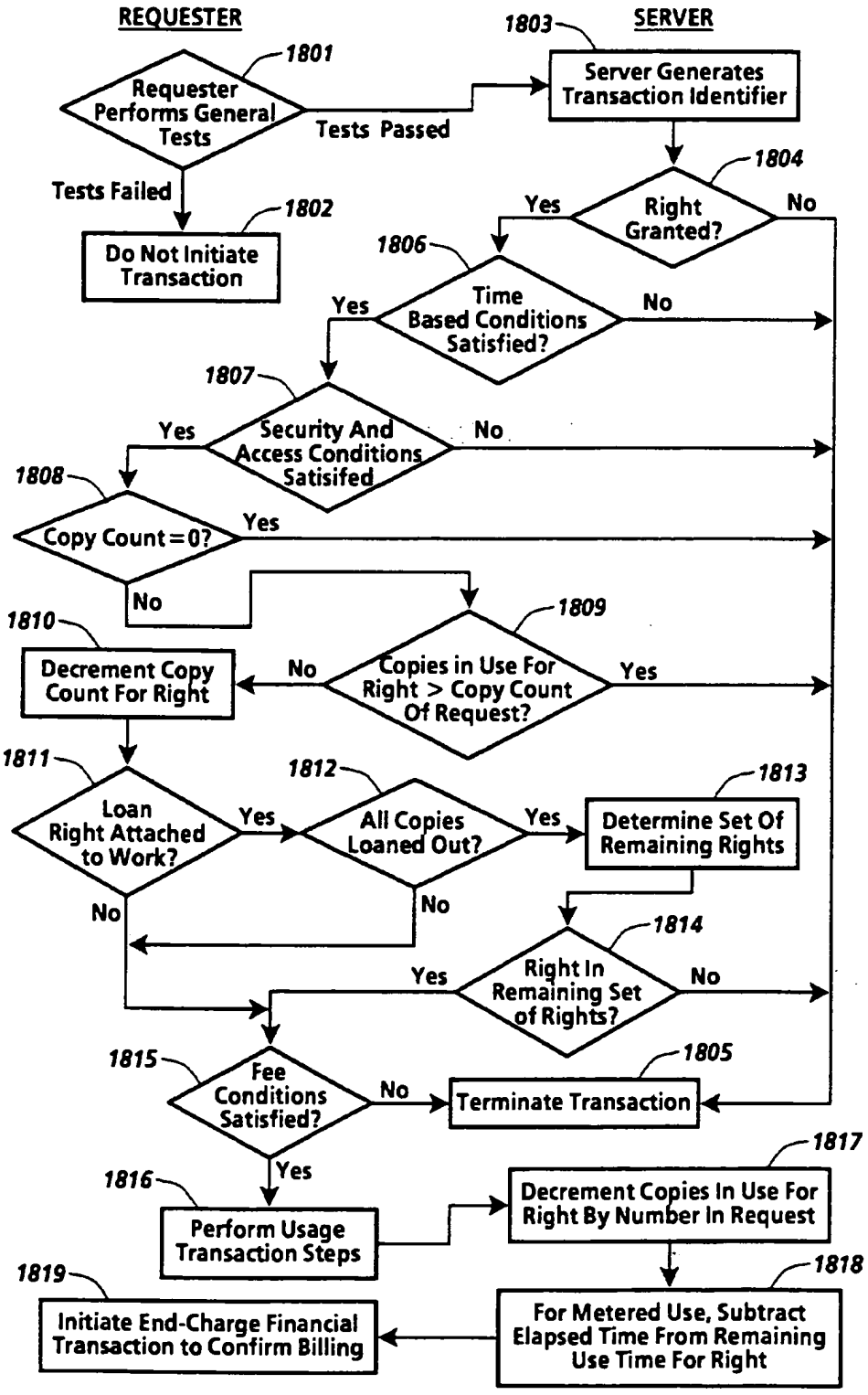


Fig.18

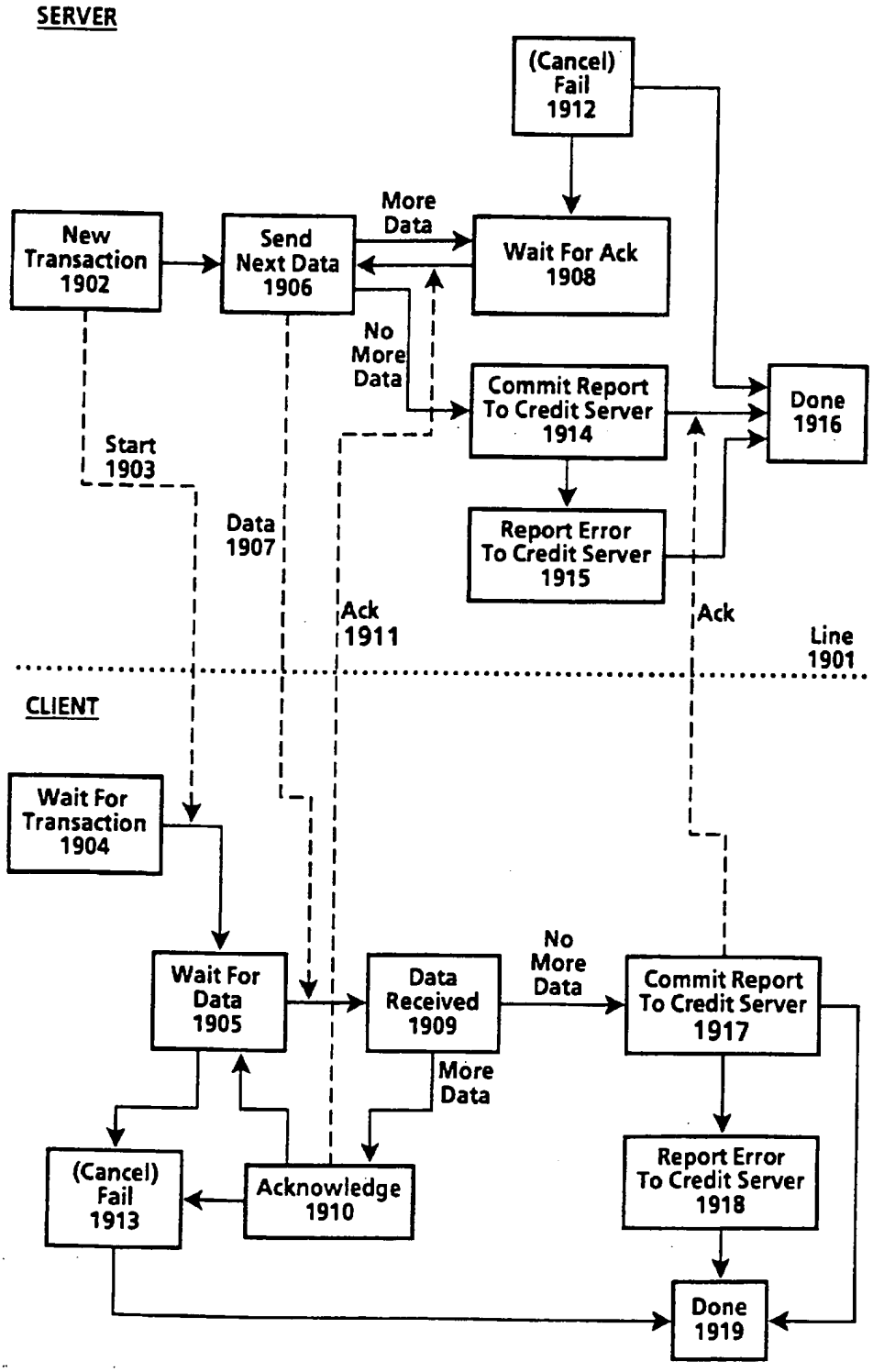


Fig.19

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 95 30 8420

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.6)
A	WO-A-92 20022 (DIGITAL EQUIPMENT CORP.) * page 45, line 10 - page 64, line 17 * ---	1,6,8,10	G06F1/00
A	GB-A-2 236 604 (SUN MICROSYSTEMS INC) * page 9, line 11 - page 20, line 15 * ---	1,6,8,10	
A	US-A-5 291 596 (MITA) * the whole document * -----	1,6,8,10	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	1 April 1996	Moens, R

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)

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

(11) EP 0 731 404 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
11.09.1996 Bulletin 1996/37

(51) Int. Cl.[6]: **G06F 1/00**, G06F 19/00

(21) Application number: **96100832.3**

(22) Date of filing: **22.01.1996**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **07.03.1995 US 401484**

(71) Applicant: **International Business Machines Corporation**
Armonk, N.Y. 10504 (US)

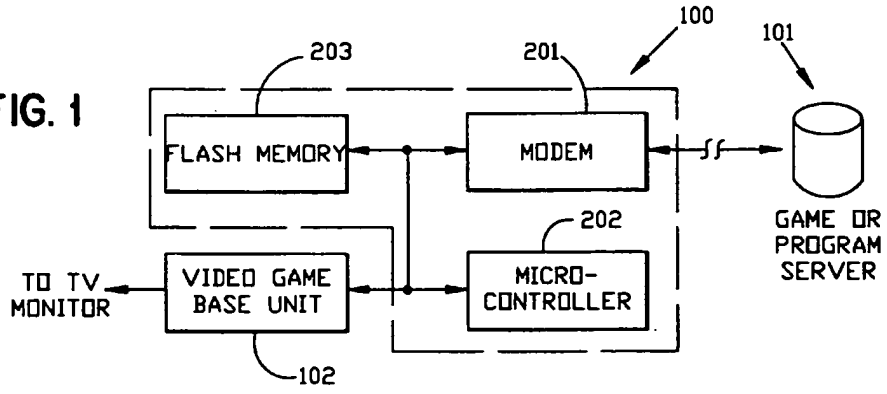
(72) Inventors:
- **Bakoglu, Halil Burhan**
  Ossining, New York 10562 (US)
- **Chen, Inching**
  Wappingers Falls, New York 12590 (US)
- **Lean, Andy Geng-Chyun**
  Merrick, New York 11566 (US)
- **Maruyama, Kiyoshi**
  Chappaqua, New York 10514 (US)
- **Yue, Chung-wai**
  Yorktown Heights, New York 10598 (US)

(74) Representative: **Rach, Werner, Dr.**
IBM Deutschland
Informationssysteme GmbH,
Patentwesen und Urheberrecht
70548 Stuttgart (DE)

(54) A universal electronic video game renting/distributing system

(57) A video game cartridge that can be plugged into a video game machine to enable a user to request and play a video game for a predetermined number of video frames. The cartridge has a receiver for receiving the video game program and the predetermined frame count in response to a request from the user. The program and frame count is then stored in a memory of the cartridge. Finally, the cartridge has a counter which changes its value when the user is actively playing the video game program. The counter ceases to change its value when the user is not playing the video game program. When the counter reaches a predetermined limit, the user is no longer authorized to play the video game program.

FIG. 1



EP 0 731 404 A1

Printed by Rank Xerox (UK) Business Services
2.13.4/3.4

Description

Technical Field

This invention relates to a video game cartridge for receiving video game programs from a remote server.

Description of the Prior Art

Today, there are many video games available for purchase or for rental at stores. Generally, there is no trial or test playing of the games in the stores, and there is no return on purchased games once the game package has been opened. Therefore, a person who is interested in any game has to buy it before playing it and thus may face the risk of not liking the game later. There is no return or refund of the game since the package has been opened. A person who rents a game from a store has to go through the usual VCR tape rental trouble of driving to the store, picking up the game and then later returning the game to the store.

To make video game rental easier for the consumer, Sega has created the Sega Channel. In this service, via cable and using a cable adapter unit which is plugged into the Sega Genesis game machine, people can play games that are downloaded to the cable adapter. It requires the on-line Sega Channel connection as well as the special adapter while the game is being played.

Down loading a software program to a personal computer over the modem connection exists today. Such software can come with a limited life where the life can be specified by expiration date, or time, or the number of times of the software usage. These schemes in limiting the software usage is not applicable to down loading video games to cartridges which are plugged into existing video game base units because these game base units do not have timer device built in. Thus a new scheme for controlling the usage of the game is needed.

The US Patent 4,905,280 to J.D. Wiedemer, et al describes a method for real time down loading of broadcast programs for pay-per-view or for subscription. Descrambling of broadcast programs is done by codes on a replaceable memory module, which is delivered to a subscriber by the service provider. This patent is applicable to the "purchase" of software content or real-time service, but it is not applicable to limiting the life of rented software.

US patent 5,251,909 to Reed et al describes software renting or distributing schemes in which access is granted to a subscriber prior to the actual programs being transmitted. This patent describes an off-line process and is not applicable to delivering software for rental purposes.

Summary of the Invention

It is an object of this invention to provide a portable video game cartridge which can be plugged into a video game machine base unit, such as Nintendos, Sega Genesis&tm. video game machine or Atari's Jaguar&tm. video game machine. The cartridge will allow a video game program to be used by receiving the video program over a telephone network or cable system.

The current invention describes a way of distributing and controlling the usage of a video game program (or any software program) by using a "watchdog mechanism" and by limiting the "life" of a game by limiting the total number of graphic frames that a video machine can generate. It offers a simple and effective way of software renting and distribution where game machines have no timer.

It is also an object of this invention to prevent piracy of video programs and programs in general by storing the frame count in a random location of the memory that is unknown to a potential pirate, especially if the count itself is encrypted. Since the count is part of the video game program or program execution path, the video game or program cannot be used without knowledge of the count.

This invention is generally an apparatus and method for enabling a user to request and use a program where the user receives the program and a frame count indicating the number of frames of the program that the user is authorized to execute or use. This program and the frame count is then stored in a memory. When the user is actively providing input to the program, the frame count changes. The frame count will cease to change when the user is not providing input to the program. When the count reaches a predetermined limit, the user is prevented from continuing use of the program.

This invention is a video game cartridge which can be plugged into a video game machine for enabling a user to receive and play a video game for a predetermined number of frames. The cartridge has a receiver for receiving the video program and for receiving a frame count indicating the number of video frames of the video game program that the user is authorized to play. The video program and frame count is then stored in a memory of the cartridge. The cartridge also has a counter which changes the frame count when the user is actively playing the video game program. When the user is not playing the video game program, the counter ceases to change its count. Finally when the counter reaches a predetermined limit, the user is prevented from further playing the video game program.

Brief Description of the Drawings

FIG. 1 schematically illustrates the major components of the video game cartridge along with a video game machine and a remote server.

FIG. 2 is a functional diagram showing the functions of each of the major components of the video game cartridge.

FIG. 3 schematically illustrates the flow chart for the watch "dog mechanism".

Description of the Preferred Embodiment

FIG. 1 illustrates a sample diagram of a electronic game or program renting system setup. The dotted line encloses the portable and programmable game cartridge unit 100 that can be plugged into a video game machine base unit 102, such as Sega Genesis&tm. video game machine, and remotely be connected to a video game server 101 via a modem connection. The connection to the remote video server can be through cable TV, or other telecommunication facilities.

When a video game base unit 102 is powered on, a user could either play a game (or games) stored in the programmable game cartridge 100 or place an order of a new game (either for rental or for purchase) to the game or program server 101. The cartridge 100 contains screen assistance (and voice assistance) to help place an order for a video game program to the server 101.

FIG. 2 illustrates the components of the video game cartridge unit 100. It consists of modem 201, microcontroller 202, flash memory 203 and an interface 204 to the video game base unit 102. The modem 201 performs the interface to the telephone or cable network. It can optionally perform decompression of received game or software if necessary. The received game is stored in flash memory 203. The game comes with its "life" which is indicated by the total number of graphic frames the video game machine 102 is authorized to generate when the game is actively played. For example, the game machine could render game graphics frame by frame at the rate of thirty framers per second.

After the number of graphic frames is exhausted, further playing of the game is prevented by the following mechanism. The flash memory 203 also stores a "watchdog mechanism" which keeps track of the remaining life of the game. An hourglass routine is embedded in the watchdog mechanism which is executed by microcontroller 202. This watchdog mechanism updates and tracks down a specified register in the flash memory 203 with its location randomly determined by the game server 101 in FIG. 1 during the down loading of the game.

The use of expiration date or time for voiding the game is an obvious approach if the video game base unit 102 comes with a timer. Since this patent application assumes a game base unit 102 which has no timer (which is the case of many existing game machines), the "life" of the rented game is determined by the total number of graphic frames that the base game unit can generate. This "life", or frame count, is what a renter gets when a game is down loaded. It is stored into a location in the flash memory 203. The location into which the frame count is stored in the flash memory is determined randomly by the video server at the time of the game down loading. The video game can resume at any time when it is being turned on, provided there is available frame count stored in the designated random location. The microcontroller 202 can pick up the frame count and allow the renting period, and thus the game or software, to be continued. As the rented game is being played, the frame count is decremented. When the user turns off the power, the hourglass routine in memory 203 will first store the remaining frame count to a random location in the non- volatile memory 203 and then shut down the game. The rental expires when there is no frame count remaining. The microcontroller 202 will not allow any portion of the game to be played by the game base unit 102 when the frame count reaches zero.

FIG. 3 illustrates the watchdog mechanism embedded with the video game program execution path that contains the hourglass routine which serves as part of the watchdog mechanism which can expire the game. When the user starts the game, the frame count is first fetched (305) and checked (306). If the frame count reaches zero, the game is over even though the game unit still has its power on (306N). If the frame count is still greater than zero (306Y), the scanner continues to monitor the game player's input in playing the video game (307). No active input (307) means the player is not playing the video game, and the scanner continues to monitor the player inputs from the key pad connected to the video game. When there is no active input, the video game will not render any game graphic frames. Therefore, the game program execution path will fall through decisions 308 and 309 and immediately return to continue scanning (307). When the game is not actively played and the player leaves the game machine's power on, the game will be sitting idle without rendering any new graphic frames. The frame count will not be consumed until the player becomes active again in playing the game as detected by the scanner (307 and 308).

If the player's input has been detected as active (307), a check is made to see if graphic rendering is required (309). Graphics rendering is required when the game program determines that the input signals from the key pad connected to the video game are valid signals. If rendering is required (309Y), the frame counter will be decremented (301). The hourglass routine (301 and 302) decrements the frame count and checks for any frame count left.

If the count is valid (302Y), then the program flows back to (310) which is the game program main collections, and then at the same time, 302 Y:sup.':esup. branches to check for power-off condition (303).

If the user decides to power-off the game, the watchdog mechanism will go through decision (303) and the shutdown routine (304) to store any remaining frame count in the flash memory. The shutdown routine stores the remaining frame count in the flash memory and exits the game. In summary flowchart components (301-306) and their associated flash memory form the "watchdog mechanism" that contains the hourglass rou-

tine (301 and 302) to keep track of the games "life" (remaining frame count). The watchdog mechanism also insures that the game can be resumed if there is still a valid frame count in the flash memory. Microcontroller (202) can also give advance warning when the rental is about to expire. Rental extension, if desired, can be downloaded again by the server (101) through a telephone or cable connection. Thus, server (101) in FIG. 1 has complete control over the game playing time, which should reflect the user's request for renting the game.

Although this embodiment was described in terms of a video game program in a cartridge, this invention can be extended to software programs in general. As long as the programs monitor user inputs, a scanner and watchdog mechanism can be implemented in similar fashion using a non-volatile memory.

The watchdog mechanism can even be made more secure by encrypting the frame count, which is stored at a random location in the memory. Even if the would-be pirate stumbles across the count in the memory, he/she wouldn't know what he/she found.

Claims

1. An apparatus for enabling a user to request and use a program, said apparatus comprising:

   a. a receiver for receiving the program and a frame count indicating a number of frames of the program that is authorized to be executed by the user;

   b. a memory for storing the program and the frame count received by the receiver; and

   c. a counter for changing the frame count when the user is actively providing input to the program, wherein the counter ceases to change its count when the user is not providing input to the program, and wherein the user is prevented from continuing use of the program when the counter reaches a predetermined limit.

2. An apparatus as recited in claim 1, further comprising:

   means for randomly determining an address in the memory in which the frame count is to be stored, and wherein the address is unknown to the user.

3. A method of enabling a user to request and use a program, said method comprising:

   a. receiving the game program and a frame count indicating a number of frames of the program that is authorized to be used by the user in response to a request;

   b. a memory for storing the program and the frame count; and

   c. changing the frame count when the user is actively using the program, wherein the frame count ceases to change when the user is not using the program and wherein the user is prevented from continuing use of the program when the counter reaches a predetermined limit.

4. A method as recited in claim 3, wherein the frame count is stored in a randomly determined location in the memory.

5. A video game cartridge which can be plugged into, for operation with, a video game machine to enable a user to request and play a video game program which is received from a remotely located server, said video game cartridge comprising:

   a. a receiver for receiving from the server the video game program and a frame count indicating a number of frames of the video game program that is authorized to be played by the user in response to a request;

   b. a memory for storing the video game program and the frame count received by the receiver; and

   c. a counter for changing the frame count when the user is actively playing the video game program, wherein the counter ceases to change its count when the user is not playing the video game program, and wherein the user is prevented from further playing the video game program when the counter reaches a predetermined limit, indicating that the user has played said video game for the number of frames.

6. A video game cartridge as recited in claim 5, further comprising:

   means for randomly determining an address in the memory in which the frame count is to be stored.

7. A video game cartridge as recited in claim 5, further comprising:

   a modem for transmitting to the server the request from the user to play a video game program.

8. A video game cartridge, as recited in claim 5, wherein said memory is a non-volatile memory.

9. A video game cartridge, as recited in claim 8, wherein the frame count indicated in the counter is stored in the memory when power for the video game machine is turned off.

10. A video game cartridge, as recited in claim 9, further comprising:

a means for fetching the frame count stored in the memory when power for said game machine is turned on.

11. A video game cartridge which can be plugged into, for operation with, a video game machine to enable a user to request and play a video game program which is received from a remotely located server, said video game cartridge comprising:

a. a modem for transmitting from the user over a telephone or cable network a request to receive the video game from the server, and for receiving the video game program and frame count from the server over the telephone or cable network, the frame count indicating a predetermined number of frames of the video game program that is authorized to be played by the user in response to the request;

b. a non-volatile memory for storing the video game program and the frame count;

c. a counter for changing the frame count when the player is actively playing the video game;

d. a means for storing the changed frame count of the counter in the memory when the power to the video game machine is turned off; and

e. a means for fetching the changed frame count stored in the memory in step (d) when the player resumes playing the video game, wherein the user is prevented from further playing of the video game program when the frame count of the counter reaches a predetermined limit, indicating that the user has played said video game for the predetermined number of frames.

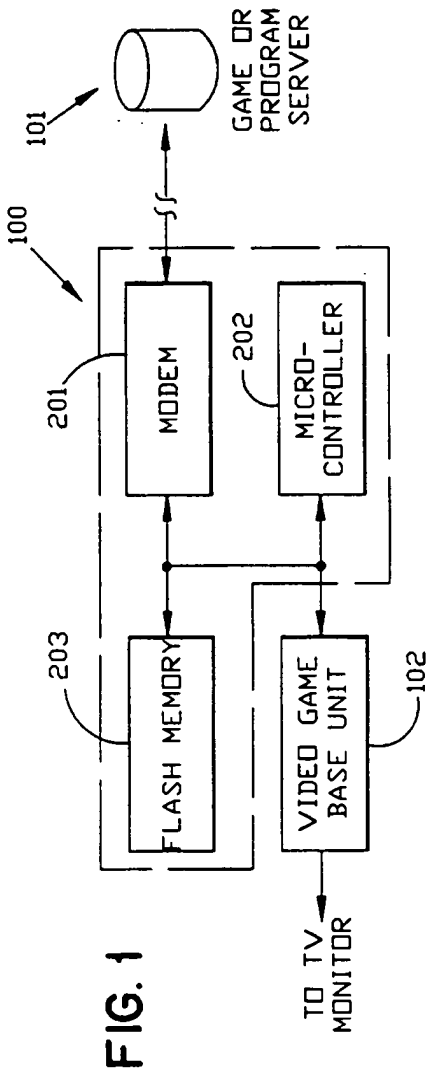


FIG. 1

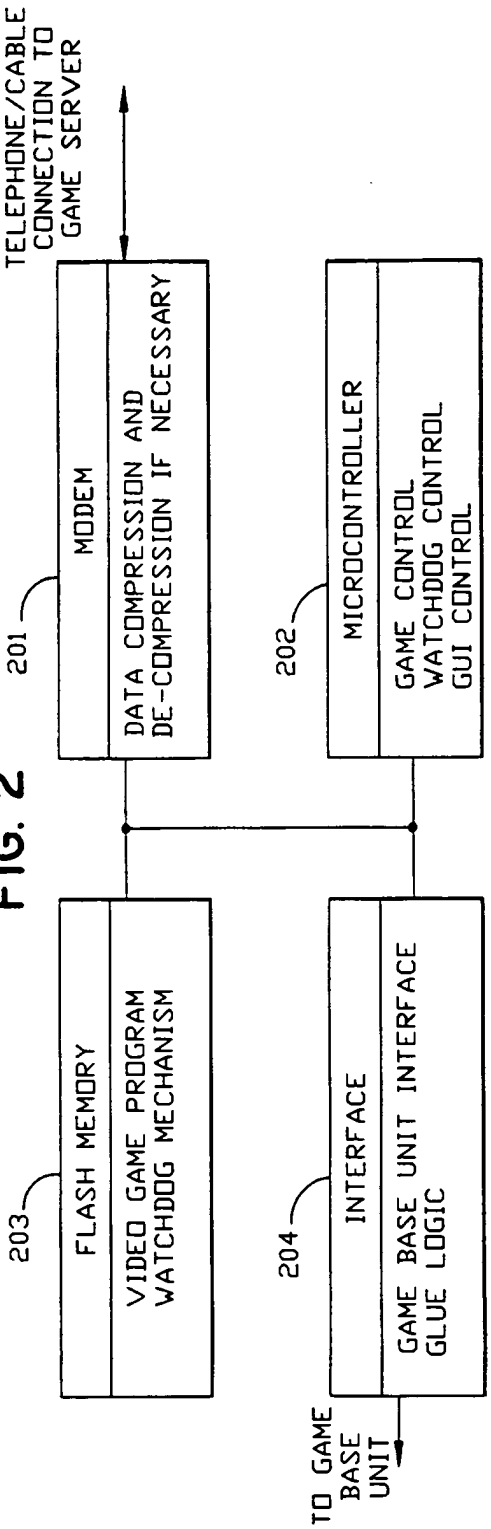
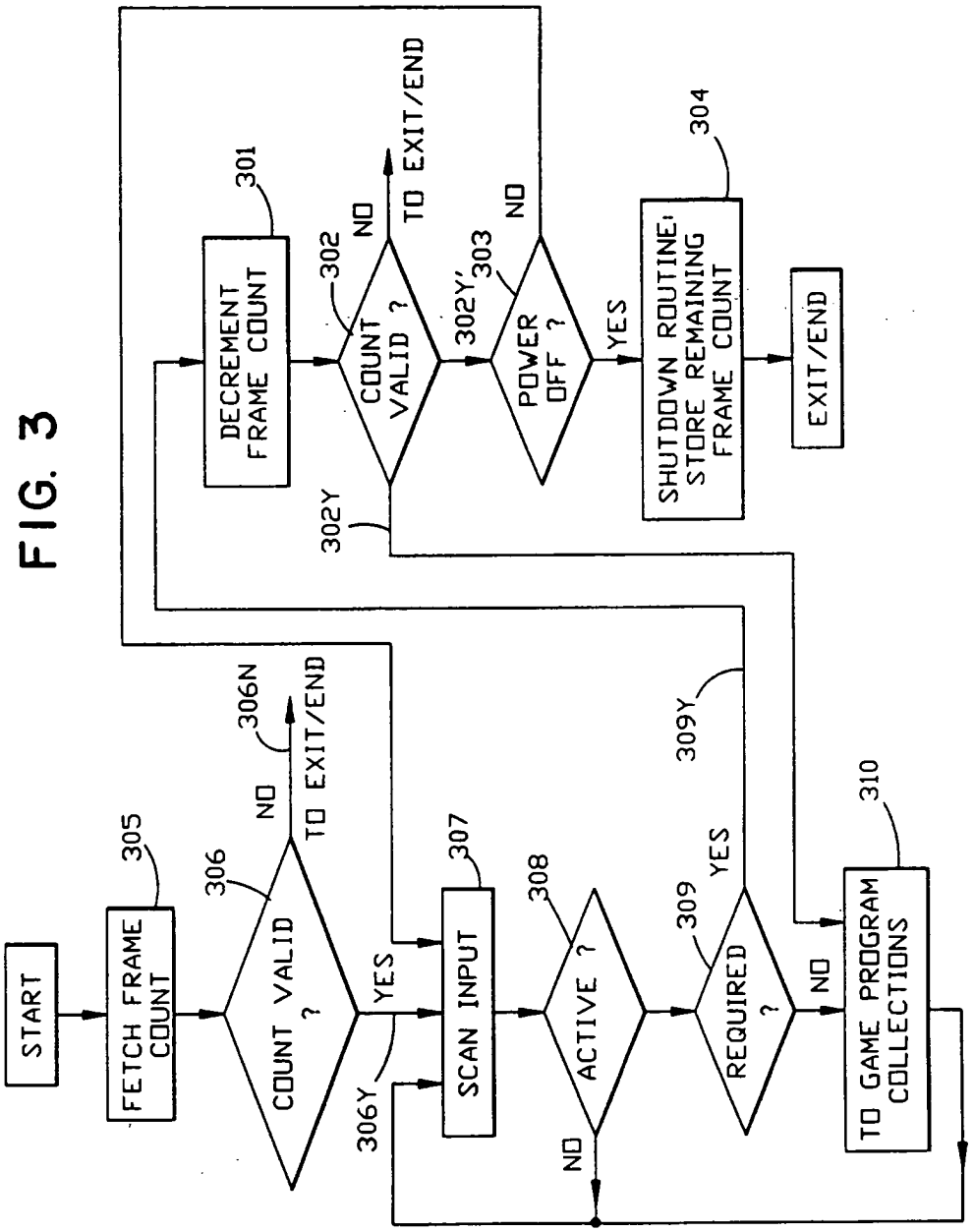


FIG. 2

FIG. 3



European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 10 0832

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.6)
P,A	EP-A-0 671 711 (SEGA ENTERPRISES KK) 13 September 1995 * the whole document * ---	1-11	G06F1/00 G06F19/00
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol. 37, no. 3, 1 March 1994, pages 413-417, XP000441522 "MULTIMEDIA MIXED OBJECT ENVELOPES SUPORTING A GRADUATED FEE SCHEME VIA ENCRYPTION" * page 413, line 1 - page 414, line 14 * ---	1-11	
A	WO-A-93 01550 (INFOLOGIC SOFTWARE INC) 21 January 1993 * page 1, line 1 - page 8, line 32 * * claims 1-3 * -----	1-11	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	14 June 1996	Powell, D

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)

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

(11) EP 0 763 936 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
19.03.1997 Bulletin 1997/12

(51) Int Cl.6: H04N 5/913

011

(21) Application number: 96306507.3

(22) Date of filing: 06.09.1996

(84) Designated Contracting States:
DE FR GB

(30) Priority: 18.09.1995 KR 9530444

(71) Applicant: LG ELECTRONICS INC.
Seoul (KR)

(72) Inventors:
- Kim, Yung Gil, c/o LG Elec. Video-Media R&D
Seoul (KR)
- Park, Tae Joon
Seoul (KR)

(74) Representative:
Cross, Rupert Edward Blount et al
BOULT WADE TENNANT
27 Furnival Street
London EC4A 1PQ (GB)

(54) Illegal view/copy protection method and apparatus for digital broadcasting system

(57) An illegal view/copy protection method for a digital broadcasting system is disclosed including an audio/video signal transmission step (200,201) for multiplexing and transmitting audio/video bit stream scrambled in control words (206) and information where the control words and CPTC information for illegal view/copy protection are encrypted (208); and an audio/video reception step (202) for decrypting (210) the transmitted bit stream to analyze the CPTC information and control words (211), deciding whether recording is allowed or not to be recorded on cassette tape, and using the control words, performing descrambling (212) and decoding (213) to output audio/video signals to a monitor, thereby protecting copyright.

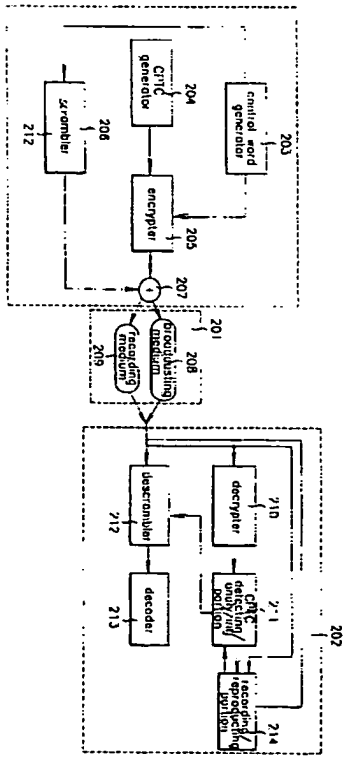


FIG.16

EP 0 763 936 A2

Printed by Jouve, 75001 PARIS (FR)

Description

Background of the Invention

The present invention relates to an illegal view/copy protection method and apparatus for a digital broadcasting system, in which digital broadcasting performed through broadcasting media such as cable, satellite and terrestrial broadcasting, or through prerecorded media such as video cassette tapes, is prevented from being illegally viewed or copied to thereby protect its copyright.

For conventional systems for copyright protection on digital media, there are Macrovision's intellectual property protection system (IPPS), which is disclosed in US Patent No. 5,315,448, and the integrated receiver/decoder (IRD), a conditional receiving system for digital broadcasting media, for receiving DirecTV's satellite broadcasting currently transmitted in the US.

The Macrovision's IPPS disclosed in US Patent No. 5,315,448 is a copy protection system for a hybrid digital VCR having digital recording functions for both a digital input signal and an analog input signal.

As shown in Figs. 1 and 2, in operating its copy protection function, Macrovision's IPPS detects, when a digital signal is input, copy protection control bits from an input signal, and when an analog signal is input, detects the analog copy protection waveform from the input signal.

More specifically, as shown in Fig. 2, a signal in which the analog copy protection waveform generated from an analog copy protection generator is added to the analog video output of the output signals of the digital VCR is output and displayed to be normal on an analog TV but distorted on an analog VCR, as shown in Fig. 1. In digital recording of the input signal, the copy protection control bits are changed to prevent digital copy or to permit one-time digital copy.

Referring to Fig. 3, the IPPS comprises an analog copy protection detector (ACP) 2 for detecting the analog copy protection waveform from an input analog NTSC video signal 1, an A/D converter 3 for A/D-converting analog NTSC video signal 1 input according to the signal output from the ACP detector, an AC bit detector 5 for detecting the AC bit from input digital video signal 4, an SCPS bit detector 6 for detecting the SCPS from input digital video signal 4, an AC bit adder 7 for adding the AC bit to input digital video signal 4 according to the SCPS bit output from SCPS bit detector 6, a switch 8 for outputting a signal output from AC bit adder 7 according to the AC bit output from AC bit detector 5, a switch 9 for selecting and outputting the signal output from A/D converter 3 and switch 8, a digital tape deck mechanism/circuit 10 for digitally recording the signal output from switch 9 and outputting a digital video signal, an AC bit detector 11 for detecting the AC bit from the signal output from digital tape deck mechanism/circuit 10, an ACP signal generator 12 for generating the ACP signal from the signal output from AC bit detector 11, and a D/A converter 13 for adding the ACP signal output from ACP signal generator 12 to the signal output from digital tape deck mechanism/circuit 10 and D/A converting the added result which is output as an analog NTSC video signal.

The operation of the IPPS will be explained below.

The copy protection control bits are made up of the AC and SCPS bits. The AC bit is added to recorded digital video data so that if the AC bit is set, digital copy is prohibited and if the SCPS bit is set, one-time digital copy is allowed.

In playback, when the AC bit is detected by AC bit detector 11, the analog copy protection waveform generated from ACP signal generator 12 is added to the analog video signal, which is output to D/A converter 13. Here, as the position of the copy protection control bits of the digital video data, an area of an MPEC-2 digital copy protection header where one-bit copyright flag and one-bit original-or-copy flag of a PES header are placed is used, or a transport-private-data field area of the transport header of the MPEC-2 is used.

The analog copy protection waveform is a signal which is severely distorted when inserted into the analog NTSC waveform and directly coupled to the analog TV. A method of generating such a signal is presented in US Patents Nos. 4,613,603 and 4,914,694. Using this method, the IPPS generates the analog copy protection waveform.

Referring to Fig. 4, the IRD, as a conditional receiving system for digital broadcasting media, for receiving the DirecTV's satellite broadcasting currently transmitted in US comprises an outdoor unit (ODU) 21 made up of a satellite antenna for receiving 12GHz-satellite broadcasting signals and a low noise block converter (LNB) for converting down the received satellite broadcasting signal into a 1GHz-signal, an IRD 20 for receiving satellite broadcasting from ODU 21 and offering audio and video services to a subscriber's TV or monitor, and an access card 22 required for conditional access (CA) for conditional reception.

Here, IRD 20 performs forward error correction (FEC), decoding, transport demultiplexing, MPEG decoding, NTSC encoding, and audio processing which is a D/A conversion.

Access card 22, whose size is similar to that of a general credit card, has a built-in IC. With this, the card receives CA-related information through a broadcast bit stream and telephone line, that is, a telco MODEM, in order to decide whether a user, subscriber, -selected channel can be viewed or not and to collect its subscription fee.

As shown in Fig. 4, IRD 20 comprises an IR receiver 25 for receiving and processing the subscriber's remote controller input, a telco MODEM 26 which is a general MODEM coupled to the telephone line, a microcomputer 27 made up of an NDC verifier code including software for the CA function and IRD software for IRD driving, a tuner/demodulator/FEC 28 for selecting one channel of

the signal received through ODU 21 and converting the selected channel into a digital bit stream for the purpose of error correction, a transport IC 29 for selecting one program of bit streams output from tuner/demodulator/FEC 28 and multiplexed with various programs, and converting the selected program into a bit stream decodable in the MPEG video decoder and MPEG audio decoder, a card reader interface 23 for data communication between transport IC 29 and access card 22, a system memory 24 coupled to transport IC 29 and for intermediate buffering of data, an MPEG video decoder 30 for expanding a video bit stream compressed in the MPEG format, a frame memory 31 for storing video data expanded in MPEG video decoder 30 in units of frame, an encode/sync/anti-tape/D/A 33 for converting the digital video data expanded in MPEG video decoder 30 into the analog NTSC format and inserting horizontal and vertical sync signals H-Sync and V-Sync and a Macrovision-mode analog copy protection signal in the conversion process, an RF modulator 34 for modulating an NTSC signal of the baseband output from encode/sync/anti-tape/D/A 33 into the RF band, an MPEG audio decoder 32 for expanding the audio bit stream compressed in the MPEG format, and a D/A 35 for converting the expanded digital audio data output from MPEG audio decoder 32 into analog.

Here, in the procedure of conversion into decodable bit stream in the MPEG video and audio decoders from transport IC 29, it is decided whether a program selected through communication with access card 22 can be viewed or not. If the bit stream is scrambled, its descrambling is performed with the access card's permission.

During the process of encode/sync/anti-tape/D/A 33 prior to NTSC video output, the analog copy protection waveform is added to prohibit copying to the analog VCR.

IRD 20 employs a CA system for conditional reception so that a subscriber views programs provided through a broadcasting medium such as satellite broadcasting.

In IRD 20, the NDC verifier code, which is software, and access card 22, which is a smart card for CA, are used to support CA function. A descrambler 36 is contained in transport IC 29.

The detailed block diagram of CA unit 37 and transport IC 29 for operating the CA function in a manner generally used in digital broadcasting is shown in Fig. 5.

More specifically, CA unit 37, included in smart card 22, is made up of smart card 38 for CA and microcomputer 39 operated with CA software.

The CA function is performed when the following two kinds of data are transmitted from a broadcasting station to the IRD. In other words, there are two types of data such as entitlement control message (ECM) or control word packet (CWP), and entitlement management message (EMM) or conditional access packet (CAP).

The EMM is accessed, through the telephone line or satellite broadcasting, to the smart card of the respective IRD at the data rate of 200kbps. The broadcasting station can access all of subscribers' smart cards in a manner that the EMM is transmitted along with ID or address. The EMM has information required to make a control word (CW) for descrambling from the ECM information. The ECM, information in which the control word is encrypted, is transmitted at a speed over 10 per second.

For satellite broadcasting, there are Europe's DVB, Korea's DBS, US' echoster, and the like, aside from DirecTV. Their CA function commonly uses the ECM and EMM information, though different means is provided for the respective broadcastings.

The conventional Macrovision's IPPS is a system having a good performance with respect to the copy protection of analog NTSC video signal. This is an appropriate copyright protection means when a program supplied through a digital medium is converted into analog audio/video signal and recorded or copied through an analog VCR.

However, the IPPS cannot guarantee a satisfactory protection if digital data is recorded or copied using a digital recording medium such as digital VCR. This is because the IPPS uses a method of operating the header's flag bits, without employing, to digital data, encoding methods such as scrambling and encryption. By doing so, hacking is easy to perform only by modulating the flag bits, resulting in very low security.

Summary of the Invention

It would therefore be desirable to provide an illegal view/copy protection method and apparatus for a digital broadcasting system in which intellectual properties supplied via digital media and protected by copyright are prohibited from being illegally recorded or copied using a digital recording medium such as digital VCR by a user.

It would also be desirable to provide an illegal view/copy protection method and apparatus for a digital broadcasting system in which data recorded on a cassette tape is always scrambled to make its hacking difficult and protect its copyright.

It would also be desirable to provide an illegal view/copy protection method and apparatus for a digital broadcasting system in which copyright is protected appropriately for respective media which are divided into broadcasting media and pre-recorded media.

It would also be desirable to provide an illegal view/copy protection method and apparatus for a digital broadcasting system in which intellectual properties supplied from a program provider are reproduced to be viewed on screen, copying of the intellectual properties copied and the number of copy are controlled arbitrarily, and fee for recording and copying is collected for the purpose of copyright protection.

According to a first aspect of the present invention, there is provided an illegal view/copy protection method for a digital broadcasting system comprising: an audio/video signal transmission step for multiplexing and transmitting audio/video bit stream scrambled in control words and information where the control words and CPTC information for illegal view/copy protection are encrypted; and an audio/video reception step for decrypting the transmitted bit stream to analyze the CPTC information and control words, deciding whether recording is allowed or not to be recorded on cassette tape, and using the control words, performing descrambling and decoding to output audio/video signals to a monitor.

According to a second aspect of the present invention, there is provided an illegal view/copy protection apparatus for a digital broadcasting system comprising: a program producing portion for multiplexing information encrypted both with the control word for scrambling and the CPTC information for prohibiting illegal view/copy, and the audio/video bit stream scrambled in control words, to thereby make a program; a distribution medium portion for distributing programs made in the program producing portion through a transmission medium; and a program receiving portion for detecting and analyzing the CPTC information from the bit stream transmitted from the distribution medium portion and the bit stream reproduced from cassette tape, and descrambling and decoding the bit stream transmitted from the distribution medium portion.

Brief Description of the Attached Drawings

Figs. 1 and 2 illustrate the operation state of a conventional IPPS;
Fig. 3 is a block diagram of a conventional IPPS;
Fig. 4 is a block diagram of an IRD system;
Fig. 5 shows a configuration of general hardware performing CA function;
Figs. 6A and 6B show formats of CPTC information of an embodiment of the present invention;
Fig. 7 shows a state of generation copy indicating the number of tape recopiable;
Figs. 8A-8D show the recording positions of the CPTC information of an embodiment of the present invention;
Fig. 9 is a flowchart of showing the transmission step of an illegal view/copy protection method embodying the present invention;
Fig. 10 is a flowchart of showing the reception step of an illegal view/copy protection method embodying the present invention;
Fig. 11 is a flowchart of the CPTC information analyzing step of Fig. 10;
Fig. 12 is a flowchart of showing the reproduction/rerecording step of an illegal view/copy protection method embodying the present invention;
Fig. 13 shows the format of an EMM lookup table;
Fig. 14 shows the format of a tape state signal;
Fig. 15 is a flowchart of showing the EMM processing step;
Fig. 16 is a block diagram of the whole configuration of an illegal view/copy protection apparatus embodying the present invention;
Fig. 17 is a block diagram of one embodiment of the program receiving portion of Fig. 16;
Fig. 18 is a block diagram of another embodiment of the program receiving portion of Fig. 16;
Fig. 19 is a block diagram of still another embodiment of the program receiving portion of Fig. 16;
Fig. 20 is a block diagram of yet another embodiment of the program receiving portion of Fig. 16;
Fig. 21 is a block diagram of the IRD shown in Figs. 17, 19 and 20;
Fig. 22 is a block diagram of the IRD and DVCR of Fig. 18;
Fig. 23 illustrates the flow of signals of Fig. 21;
Fig. 24 is a block diagram of one embodiment of the smart card of Fig. 17;
Fig. 25 is a block diagram of another embodiment of the smart card of Fig. 17; and
Fig. 26 is a block diagram of the DVCR of Fig. 17.

Detailed Description of the Invention

An illegal view/copy protection method for a digital broadcasting system embodying the present invention is performed by audio/video signal transmission and audio/video reception steps.

In the audio/video signal transmission step, audio/video bit stream scrambled in control words and information where the control words and CPTC information for illegal view/copy protection are encrypted are multiplexed and transmitted.

In the audio/video reception step, the bit stream transmitted in the audio/video signal transmission step is decrypted to analyze the CPTC information and control words. By doing so, it is decided whether recording is allowed or not. This result is recorded on cassette tape. Using the control words, descrambling and decoding are performed, and then audio/video signals are output to a monitor. Here, the CPTC information separately manages the ECM, EMM and control words, and contains CA information, to thereby control illegal view/copy protection. The CPTC information will be described with reference to Figs. 6A and 6B.

The CPTC information is formatted in a generational copy control field for limiting the number of copy available in order to control the depth of generational copy, and a reproducibility control field for limiting the reproduction of a copied program in order to control the number of copyiable tapes. As shown in Fig. 6A, formatting is performed containing a descrambling information field where part of the control words for descrambling are recorded, or containing a CA field where CA information for conditional access is recorded, as shown in Fig. 6B.

The CPTC information may be encrypted separately to be multiplexed with scrambled digital data, or contained in the ECM information for CA for encryption and multiplexing. Here, the generational copy control field is made up of a permissible generational field for limiting the number of copy permissible and a present generational field for indicating the present generation of a program copied. If the present generation stored in the present generational field is greater than or equal to the permissible generation stored in the permissible generational field, recording or copying is impossible.

A reproduction control field is made up of a reproducible number field for limiting the number of reproducing a copied program, and a maximum reproducible time field for limiting time to reproduce the copied program.

Here, the reproducible number stored in the reproducible number field implements a conditional-number reproducibility function according to the current reproduction number of cassette tape. The maximum reproducible time stored in the maximum reproducible time field implements the conditional-time reproducibility function of copied cassette tape according to the current time information of digital hardware.

The CPTC information may allow the copied cassette tape to be always reproducible, make it never reproducible, allow it to be reproducible as many as a limited number, or make the copied cassette tape reproducible for a limited time after recording or copying.

Using the permissible generational field and present generational field of the generational copy control field, the reproducible number field of the reproduction control field, and data of the maximum reproducible time field, the depth of generation copy, recopying of copied cassette tape, and reproduction time and number are controlled. This process controls the number of copiable cassette tape copied, and reproduction time and number.

In other words, as shown in Fig. 7, information stored in the permissible generational field and present generational field is used to allow first and second generation copy to be perform. Information stored in the reproducible number field and maximum reproducible time field is used to allow reproduction as many as a limited number or for a limited time.

In order to prohibit illegal recording or copy of a program protected by copyright law, collect fee for recording or copy, or arbitrarily control the number of reproducible copied tape to be made from a program supplied by a provider, the depth of generation copy and reproduction of copy tape are controlled to decide how long the first generation recording and copy and second generation copy are made possible.

For this purpose, the copy tape made to be always reproducible, it is made never to be reproducible, it is made to be reproducible as many as a limited number, or it is made to be reproducible for a limited time after recording or copy.

The data recorded on cassette tape contains scrambled audio/video bit stream and CPTC information. The CPTC information is recorded on a recording medium, that is, a rental tape, to prohibit illegal view/copy.

In other words, as shown in Fig. 8A, the CPTC information is overwritten on the scrambled audio/video bit stream for the error effect and recorded on cassette tape. Otherwise, as shown in Fig. 8B, the CPTC information is recorded on a portion of the audio track of cassette tape, on the control track of cassette tape as shown in Fig. 8C, or on the video track of cassette tape as shown in Fig. 8D.

In other words, as shown in Fig. 8A, the CPTC information is overwritten in a predetermined position in the form of error after parities for error correction, that is, inner and outer parities, are added to the scrambled digital data. This method reduces error correction capability but requires no additional tape area for recording the CPTC information. Further, during interleaving and decoding of ECC, the CPTC information is recognized as an error and removed, obtaining the scrambled digital data. Here, the CPTC information is detected separately.

In case that the CPTC information is recorded in part of audio track or control track, as shown in Figs. 8B and 8C, the audio head or control head must be additionally used as the means for detecting the CPTC so that audio track and control track are additionally accessed to detect the CPTC information.

The audio/video signal transmission step using the CPTC information will be explained with reference to Fig. 9.

One embodiment of the audio/video signal transmission step is to transmit an audio/video signal not containing the CA information for conditional access. This, having only the copy protection function, is used in case that a program which can be provided to all viewers is transmitted.

As shown in Fig. 9, the first embodiment of the audio/video signal transmission step comprises the steps of: encoding (100) the audio/video bit stream; generating (105) a control word for scrambling; scrambling (104) for the encoded audio/video bit stream using the generated control word; generating (102) CPTC information for illegal view/copy protection; encrypting (103) for encrypting the control word and CPTC information; and multiplexing and transmitting (106) the scrambled audio/video bit stream and encrypted CPTC information.

In other words, in step 100, the audio/video bit stream is encoded. In step 105, the control word for scrambling is generated. In step 104, the encoded audio/video bit stream is scrambled using the generated control word. In step 102, the CPTC information for illegal view/copy protection is generated. In step 103, the CPTC information and CA information are encrypted using the generated control word. The scrambled audio/video bit stream, encrypted CPTC information and CA

information are multiplexed and transmitted through a transmission medium in step 106. The audio/video signal transmitted through the first embodiment of the audio/video signal transmission step is received through one embodiment of an audio/video reception step.

Referring to Fig. 10, the first embodiment of the audio/video reception step comprises the steps of filtering (110) the transmitted bit stream and decrypting (111) the CPTC information; analyzing (113 and 114) the CPTC information to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information; deciding (115) whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmitted bit stream on cassette tape; and descrambling and decoding (116 and 117) the transmitted bit stream in the control word and outputting an audio/video signal.

In other words, the bit stream transmitted in the first embodiment of the audio/video signal transmission step is filtered and the CPTC information is decrypted in steps 110 and 111. The CPTC information is analyzed to generate the control word and the signal for controlling the protection of copyright, and the CPTC information is updated in steps 113 and 114. Whether to allow recording is determined by the generated signal for controlling the protection of copyright so that the scrambled and transmitted bit stream is recorded on cassette tape in step 115. Then, the transmitted bit stream is descrambled and decoded in control words and output as an audio/video signal in steps 116 and 117. Here, all of the control word is contained in the CPTC information.

Referring to Fig. 11, the CPTC information analyzing step comprises the steps of detecting (130, 131, 132 and 133) the permissible generation of the permissible generational field for limiting the available number of copy of a program of the CPTC information and the present generation of the present generational field indicating the present generation of the program copied, to thereby perform copy-impossible and update the CPTC information; and detecting (134, 135, 136 and 137) the reproducible number of the reproducible number field for limiting the number of reproduction of copied programs of the CPTC information, the maximum reproducible time of the maximum reproducible time field for limiting time to reproduce the copied program, and the number and time of reproduction of tape, to thereby process reproduction-impossible.

The copying number limiting step comprises the steps of: comparing (130) the permissible generation of the permissible generational field and the present generation of the present generational field and deciding whether the permissible generation is below the present generation; if the permissible generation is below the present generation, generating (131) an output disable signal to make copying impossible and destroying the control word; and if the permissible generation is not below the present generation, increasing (132) the present invention by '1' and recording the result on cassette tape. If the permissible generation is not below the present generation, the CPTC information is updated in step 133, instead of increasing the present generation by '1.'

In order to control generation copy, the permissible generation of the permissible generational field and the present generation of the present generational field are compared in step 130. If the permissible generation is below the present generation, the output disable signal is generated to make copying impossible and the control word is destroyed in step 131. If the permissible generation is not below the present generation, the present generation is increased by '1' and thus recorded on cassette tape in step 132. This enables generation copy. Here, it can be possible that generation copy is limited by updating the CPTC information, instead of increasing the present generation by '1.'

The reproduction limiting step comprises the steps of: comparing the reproducible number of the reproducible number field and the reproduction number of tape and deciding (134) whether the reproducible number is below the reproduction number of tape; if the reproducible number is not below the reproduction number of tape, comparing the maximum reproducible time and reproduction time of tape, and deciding (135) whether the maximum reproducible time is below the reproduction time of tape; if the maximum reproducible time is not below reproduction time of tape, turning off (136) an enable erase signal to thereby enable the copied program to be reproduced; if the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, turning on (137) the enable erase signal to make the reproduction of the copied program impossible so that part of or the whole program recorded on cassette tape is erased.

In order to control reproduction, the reproducible number of the reproducible number field and the reproduction number of tape are compared in step 134. If the reproducible number is not below the reproduction number of tape, the maximum reproducible time of the maximum reproducible time field and the reproduction time of tape are compared and it is decided whether the maximum reproducible time is below the reproduction time of tape in step 135. In other words, though reproducible, whether it is limited by the reproducible time must be checked. If the maximum reproducible time is not below the reproduction time of tape, the enable erase signal is turned off in step 136 to thereby make the copied program reproducible. If the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, the enable erase signal is turned on to prohibit the reproduction of the copied program. By doing so, part of or the whole program recorded on cassette tape is erased to make copy and reproduction impossible in step 137.

Here, the current time is transmitted to the user by

a provider along with a program. In this case, the copyright protection system implements limited time reproduction using transmitted time information. In this method, the program provider manages the whole users' time so that time modulation by a user cannot occur. Therefore, this is very secure.

The bit stream transmitted in the first embodiment of the audio/video signal transmission step contains ECM and EMM. Part of the control word may be contained in the CPTC information. Its remainder may be contained in the ECM or EMM. The whole control word is contained in the ECM or EMM.

The audio/video signal containing the control word and transmitted according to the audio/video signal transmission step is received according to another embodiment of the audio/video reception step.

Referring to Fig. 10, the second embodiment of the audio/video reception step comprises the steps of filtering (110) the transmitted bit stream and decrypting (111) the CPTC information and control word; filtering (118) the control word; analyzing (113 and 114) the CPTC information to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information; deciding (115) whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmitted bit stream on cassette tape; and descrambling and decoding (116 and 117) the transmitted bit stream in control words and outputting an audio/video signal.

In other words, the bit stream transmitted in the audio/video signal transmission step is filtered and the CPTC information and control word are decrypted in steps 110 and 111. The control word is filtered in step 118. The decrypted CPTC information is analyzed to generate the control word and the signal for controlling the protection of copyright, and the CPTC information is updated in steps 113 and 114. Whether to allow recording is determined by the generated signal for controlling the protection of copyright so that the scrambled and transmitted bit stream is recorded on cassette tape in step 115. Then, the transmitted bit stream is descrambled and decoded in control words and output as an audio/video signal in steps 116 and 117.

Referring to Fig. 11, in the same manner as the first embodiment of the audio/video reception step, the CPTC information analyzing step comprises the steps of: generating the control words; detecting (130, 131, 132 and 133) the permissible generation of the permissible generational field for limiting the available number of copy of a program of the CPTC information and the present generation of the present generational field indicating the present generation of the program copied, to thereby perform copy-impossible and update the CPTC information; and detecting (134, 135, 136 and 137) the reproducible number of the reproducible number field for limiting the number of reproduction of copied programs of the CPTC information, the maximum reproducible time of the maximum reproducible time field for limiting time to reproduce the copied program, and the number and time of reproduction of tape, to thereby process reproduction-impossible.

The copying number limiting step comprises the steps of: comparing (130) the permissible generation of the permissible generational field and the present generation of the present generational field and deciding whether the permissible generation is below the present generation; if the permissible generation is below the present generation, generating (131) an output disable signal to make copying impossible and destroying the control word; and if the permissible generation is not below the present generation, increasing (132) the present invention by '1' and recording the result on cassette tape. If the permissible generation is not below the present generation, the CPTC information is updated in step 133, instead of increasing the present generation by '1.'

The reproduction limiting step comprises the steps of: comparing the reproducible number of the reproducible number field and the reproduction number of tape and deciding (134) whether the reproducible number is below the reproduction number of tape; if the reproducible number is not below the reproduction number of tape, comparing the maximum reproducible time and reproduction time of tape, and deciding (135) whether the maximum reproducible time is below the reproduction time of tape; if the maximum reproducible time is not below reproduction time of tape, turning off (136) an enable erase signal to thereby enable the copied program to be reproduced; if the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, turning on (137) the enable erase signal to make the reproduction of the copied program impossible so that part of or the whole program recorded on cassette tape is erased.

Another embodiment of the audio/video signal transmission step is to transmit an audio/video signal containing the CA information for conditional access. This, having the illegal reception and copy protection functions, is used in case that a program which can be provided to limited viewers is transmitted.

As shown in Fig. 9, the second embodiment of the audio/video signal transmission step comprises the steps of: encoding (100) the audio/video bit stream; generating (105) a control word for scrambling; scrambling (104) for the encoded audio/video bit stream using the generated control word; generating (102) CPTC information for illegal view/copy protection; generating (101) CA information for conditional reception; encrypting (103) for encrypting the CPTC information and CA information; and multiplexing and transmitting (106) the scrambled audio/video bit stream and encrypted CPTC information and CA information.

In other words, in step 100, the audio/video bit stream is encoded. In step 105, the control word for scrambling is generated. In step 104, the encoded au-

dio/video bit stream is scrambled using the generated control word. In step 102, the CPTC information for illegal view/copy protection is generated. In step 101, CA information for conditional reception is generated. In step 103, the CPTC information and CA information are encrypted using the generated control word. The scrambled audio/video bit stream, encrypted CPTC information and CA information are multiplexed and transmitted through a transmission medium in step 106. The audio/video signal transmitted through the second embodiment of the audio/video signal transmission step is received through the second embodiment of the audio/video reception step.

Referring to Fig. 10, the second embodiment of the audio/video reception step comprises the steps of: filtering (110) the transmitted bit stream and decrypting (111) the CPTC information; analyzing (112, 113 and 114) the CPTC information and CA information to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information; deciding (115) whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmitted bit stream on cassette tape; and descrambling and decoding (116 and 117) the transmitted bit stream and outputting an audio/video signal.

Referring to Fig. 11, in the same manner as the first embodiment of the audio/video reception step, the CPTC information analyzing step comprises the steps of: generating a control word; detecting (130, 131, 132 and 133) the permissible generation of the permissible generational field for limiting the available number of copy of a program of the CPTC information and the present generation of the present generational field indicating the present generation of the program copied, to thereby perform copy-impossible and update the CPTC information; and detecting (134, 135, 136 and 137) the reproducible number of the reproducible number field for limiting the number of reproduction of copied programs of the CPTC information, the maximum reproducible time of the maximum reproducible time field for limiting time to reproduce the copied program, and the number and time of reproduction of tape, to thereby process reproduction-impossible.

In the same manner as the first embodiment of the audio/video reception step, the copying number limiting step comprises the steps of: comparing (130) the permissible generation of the permissible generational field and the present generation of the present generational field and deciding whether the permissible generation is below the present generation; if the permissible generation is below the present generation, generating (131) an output disable signal to make copying impossible and destroying the control word; and if the permissible generation is not below the present generation, increasing (132) the present invention by '1' and recording the result on cassette tape. If the permissible generation is not below the present generation, the CPTC information is updated in step 133.

The reproduction limiting step comprises the steps of: comparing the reproducible number of the reproducible number field and the reproduction number of tape and deciding (134) whether the reproducible number is below the reproduction number of tape; if the reproducible number is not below the reproduction number of tape, comparing the maximum reproducible time and reproduction time of tape, and deciding (135) whether the maximum reproducible time is below the reproduction time of tape; if the maximum reproducible time is not below reproduction time of tape, turning off (136) an enable erase signal to thereby enable the copied program to be reproduced; if the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, turning on (137) the enable erase signal to make the reproduction of the copied program impossible so that part of or the whole program recorded on cassette tape is erased.

The bit stream transmitted in the second embodiment of the audio/video signal transmission step contains ECM and EMM. Part of the control word may be contained in the CPTC information. Its remainder may be contained in the ECM or EMM. The whole control word is contained in the ECM or EMM.

The audio/video signal containing the control word and transmitted according to the audio/video signal transmission step is received according to another embodiment of the audio/video reception step. The audio/video signal transmitted in the audio/video signal transmission step containing the control word is received according to still another embodiment of the audio/video reception step.

Referring to Fig. 10, the third embodiment of the audio/video reception step comprises the steps of: filtering (110) the transmitted bit stream and decrypting (111) the CPTC information and CA information; analyzing (112, 113, 114 and 118) the CPTC information and CA information and filtering the control word to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information; deciding (115) whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmitted bit stream on cassette tape; and descrambling and decoding (116 and 117) the transmitted bit stream and outputting an audio/video signal.

Referring to Fig. 11, in the same manner as the first embodiment of the audio/video reception step, the CPTC information analyzing step comprises the steps of: generating the control words; detecting (130, 131, 132 and 133) the permissible generation of the permissible generational field for limiting the available number of copy of a program of the CPTC information and the present generation of the present generational field indicating the present generation of the program copied, to thereby perform copy-impossible and update the CPTC information; and detecting (134, 135, 136 and

137) the reproducible number of the reproducible number field for limiting the number of reproduction of copied programs of the CPTC information, the maximum reproducible time of the maximum reproducible time field for limiting time to reproduce the copied program, and the number and time of reproduction of tape, to thereby process reproduction-impossible.

The copying number limiting step comprises the steps of: comparing (130) the permissible generation of the permissible generational field and the present generation of the present generational field and deciding whether the permissible generation is below the present generation; if the permissible generation is below the present generation, generating (131) an output disable signal to make copying impossible and destroying the control word; and if the permissible generation is not below the present generation, increasing (132) the present invention by '1' and recording the result on cassette tape, and if the permissible generation is not below the present generation, updating the CPTC information in step 133.

The reproduction limiting step comprises the steps of: comparing the reproducible number of the reproducible number field and the reproduction number of tape and deciding (134) whether the reproducible number is below the reproduction number of tape; if the reproducible number is not below the reproduction number of tape, comparing the maximum reproducible time and reproduction time of tape, and deciding (135) whether the maximum reproducible time is below the reproduction time of tape; if the maximum reproducible time is not below reproduction time of tape, turning off (136) an enable erase signal to thereby enable the copied program to be reproduced; if the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, turning on (137) the enable erase signal to make the reproduction of the copied program impossible so that part of or the whole program recorded on cassette tape is erased.

The illegal view/copy protection method for digital broadcasting system embodying the present invention, after the audio/video signal transmission step and audio/video reception step, further comprises a reproduction and rerecording step of: decrypting the bit stream recorded and reproduced on cassette tape, analyzing the CPTC information, deciding whether to allow rerecording, recording the result on cassette tape, filtering the control word, and performing descrambling and decoding to output an audio/video signal.

Referring to Fig. 12, the audio/video reproduction and rerecording step comprises the steps of: filtering (120) the bit stream recorded and reproduced on video tape, and decrypting (121) the CPTC information; analyzing (122 and 123) the CPTC information to generate control words and a signal for controlling the protection of copyright and update the CPTC information; deciding (124) whether to allow recording according to the signal of controlling the protection of copyright, and recording the scrambled and transmitted bit stream on cassette tape; descrambling and decoding (125 and 126) the transmitted bit stream in control words to output an audio/video signal; and deciding whether to allow post-reproduction according to the signal for controlling the protection of copyright to thereby erase part of or the whole data recorded on cassette tape.

Here, EMM may contain information required for decoding information in order to perform the illegal view/copy protection method of a broadcasting system. In this case, a step of storing and processing the EMM is added in the audio/video reproduction and rerecording step.

In the EMM storing and processing step, in case that the EMM is updated by a broadcasting station for the purpose of copyright protection, the EMM having information required to decode the CPTC information is stored in order to continuously reproduce programs of copied cassette tape.

Here, an ID number indicative of updating the EMM is recorded on cassette tape. The EMM is stored to which the updating state and the ID number of cassette tape are mapped.

The EMM storing and processing step comprises the steps of: storing all EMM to be updated and corresponding ID information; selecting the latest EMM in recording cassette tape; recording a corresponding ID number; and selecting an EMM corresponding to the ID number recorded on cassette tape in reproducing the cassette tape.

As shown in Fig. 13, all EMMs (EMM1, EMM2, EMM3,...) to be updated on the EMM lookup table and corresponding ID information (ID1, ID2, ID3,...) are mapped and stored.

Referring to Figs. 14 and 15, in recording a program on cassette tape, that is, when recording is indicated in the recording/reproduction mode, an ID number corresponding to the latest, the final, EMM, is recorded. Thereafter, in reproducing the cassette tape, that is, when reproduction is indicated in the recording/reproduction mode, an EMM corresponding to the ID number recorded on cassette tape is selected from the EMM lookup table so that the recorded program is reproduced according to the reproducible number of the reproducible number field and the reproduction number recorded on the video tape.

Referring to Fig. 16, an illegal view/copy protection apparatus of digital broadcasting system embodying the present invention comprises a program producing portion 200, distribution medium portion 201, and program receiving portion 202.

Program producing portion 200 offers programs, in which information encrypted both with the control word for scrambling and the CPTC information for prohibiting illegal view/copy, and the audio/video bit stream scrambled in control words are multiplexed to make a program.

Distribution medium portion 201 distributes pro-

grams made in program producing portion 200 through a transmission medium.

Program receiving portion 202 detects and analyzes the CPTC information from the bit stream transmitted from distribution medium portion 201 and the bit stream reproduced from cassette tape, and descrambles and decodes the bit stream transmitted from distribution medium portion 201. The descrambled and decoded bit stream is displayed or recorded on cassette tape.

Program producing portion 200 comprises a control word generator 203 for generating a control word for scrambling, a CPTC generator 204 for generating the CPTC information for prohibiting illegal view/copy, a scrambling portion 206 for scrambling the audio/video bit stream using the control word output from control word generator 203, an encrypting portion 205 for encrypting the control word output from control word generator 203 and the CPTC information output from CPTC generator 204, and an adder 207 for multiplexing the signals output from scrambling portion 206 and encrypting portion 205 and transmitting them to distribution medium portion 201.

Distribution medium portion 201 comprises a broadcasting medium 208 for distributing the program made by program producing portion 200 through cable, satellite or terrestrial broadcasting, and a recording medium 209 for distributing the program made by program producing portion 200 through cassette tape.

Program receiving portion 202 comprises a decrypting portion 210 for decrypting the bit stream transmitted from broadcasting medium 208, a CPTC detecting/analyzing portion 211 for detecting and analyzing the CPTC information from the bit stream output from decrypting portion 210 and recording medium 209, and outputting signals for controlling the control word and illegal view/copy, a descrambling portion 212 for descrambling the bit stream transmitted from broadcasting medium 208 and recording medium 209 and the bit stream reproduced from cassette tape, a decoding portion 213 for decoding and displaying the signal output from descrambling portion 212, and a recording/reproducing portion 214 for recording the bit stream transmitted from broadcasting medium 208 and recording medium 209 according to the signal output from CPTC detecting/analyzing portion 211, and reproducing cassette tape, to thereby output the result to descrambling portion 212 and CPTC detecting/analyzing portion 211.

The operation of an illegal view/copy protection apparatus for a digital broadcasting system embodying the present invention will be described below.

Control word generator 203 generates a control word for scrambling, and CPTC generator 204 generates the CPTC information for prohibiting illegal view/copy. Scrambling portion 206 scrambles the audio/video bit stream using the generated control word. Encrypting portion 205 encrypts the CPTC information output from CPTC generator 204 using the generated control word. The audio/video bit stream scrambled in scrambling portion 206 is multiplexed with the encrypted CPTC information in adder 207. The multiplexed result is transmitted to a reception port through distribution medium portion 201.

The signal output from adder 207 is transmitted to program receiving portion 202 through broadcasting medium 208 such as cable, satellite, and terrestrial broadcastings, or through recording medium 209 made of cassette tape such as rental tape.

The bit stream transmitted through broadcasting medium 208 is decrypted in decrypting portion 210. The CPTC information is detected and analyzed in CPTC detecting/analyzing portion 211 so that signals for controlling the control word and illegal view/copy are output. Here, the bit stream transmitted to cassette tape through recording medium 209 is reproduced in recording/reproducing portion 214 and input to descrambling portion 212 and CPTC detecting/analyzing portion 211. The bit stream transmitted from broadcasting medium 208 and the bit stream reproduced from recording medium 209 through recording/reproducing portion 214 are descrambled in descrambling portion 212 according to the control word output from CPTC detecting/analyzing portion 211. The signal output from descrambling portion 212 is decoded in decoding portion 213 and displayed. The bit stream transmitted from broadcasting medium 208 and recording medium 209 is recorded on cassette tape in a recording/reproducing portion 214 according to the signal output from CPTC detecting/analyzing portion 211.

Data received from program receiving portion 202 and recorded on cassette tape is made up of the scrambled audio/video bit stream and CPTC information. The configuration of the program receiving portion having decrypting portion 210, CPTC detecting/analyzing portion 211, descrambling portion 212, decoding portion 213 and recording/reproducing portion 214 will be explained with reference to Figs. 17, 18, 19, and 20.

One embodiment of the program receiving portion of Fig. 17 receives and processes data transmitted via a broadcasting medium. Specifically, this embodiment performs conditional access and copy protection.

Referring to Fig. 17, the first embodiment of the program receiving portion comprises an IRD 222 for receiving, decoding and descrambling the bit stream transmitted from broadcasting medium 208, outputting analog audio/video data to be displayed and outputting scrambled digital audio/video data to be recorded on cassette tape, a smart card 221 for decrypting the bit stream output from IRD 222, detecting/analyzing the CPTC information, and outputting the control word and signals for controlling illegal view/copy to IRD 222 in order to perform conditional access and copy protection, a DVCR 223 for recording the digital audio/video data and CPTC information scrambled and output from IRD 222 on cassette tape, and reproducing the scrambled digital audio/video data and CPTC information recorded on cassette tape to be output to IRD 222, and a lookup table 224 for,

in case that the EMM is updated by a broadcasting station for the purpose of copyright protection, storing EMM having information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction to smart card 221 in order to continuously reproduce the program of copied cassette tape. Here, lookup table 221 is mapped and processed as shown in Figs. 13, 14 and 15.

The operation of the first embodiment of the program receiving portion will be described below.

In case that a bit stream, that is, a program, is received through a broadcasting medium, the received audio/video data is scrambled digital audio/video data.

The received bit stream is decoded in IRD 222 and decrypted in smart card 221. Its CPTC information is detected and analyzed so that a signal for controlling the control word and illegal view/copy is output to IRD 222.

IRD 222 descrambles the decoded bit stream using the bit stream output from smart card 221 and signals for controlling illegal view/copy. The descrambled bit stream is output to display analog audio/video data. IRD 222 outputs the scrambled digital audio/video data and CPTC information to DVCT 223 in order to record them on cassette tape.

The scrambled digital audio/video data and CPTC information output from IRD 222 is recorded on cassette tape in DVCR 223. They are in turn reproduced in DVCR 223 and processed in the same manner that the bit stream transmitted via the broadcasting medium is descrambled and processed in IRD 222 and smart card 221. The processed result is output to be displayed on a monitor, or output to the DVCR and recopied.

Here, reproduction and recopy are made possible by the data stored in the permissible generational field, present generational field, reproducible number field, and maximum reproducible time field contained in the CPTC information.

Updated EMM is mapped and stored in lookup table 224 so that, when the EMM is updated through a broadcasting signal in a broadcasting station in order to protect copyright, the program of cassette tape copied can be continuously reproduced.

Lookup table 224 reads out the EMM containing information required to decode the CPTC information in reproducing the cassette tape. Corresponding CPTC information is output to smart card 221 to enable reproduction.

Another embodiment of the program receiving portion shown in Fig. 18 is to receive and process data transmitted through a recording medium, for instance, rental tape.

The second embodiment of the program receiving portion, as shown in Fig. 18, comprises a DVCR 232 for detecting/analyzing the CPTC information from the bit stream transmitted from the recording medium, outputting a control word and signals for controlling illegal view/copy, and reproducing scrambled digital audio/video data, and an IRD 231 for receiving the control word and signals for controlling illegal view/copy output from DVCR 232, descrambling the scrambled digital audio/video data, and outputting analog audio/video data to be displayed or recorded.

The second embodiment of the program receiving portion is to perform CPTC detection and processing carried out in the smart card of the first embodiment of the program receiving portion shown in Fig. 17. The operation of the second embodiment of the program receiving portion will be described below.

In case that the bit stream is received through the recording medium, the audio/video data reproduced through the DVCR is scrambled digital audio/video data.

The bit stream recorded in DVCR 232 is reproduced. Its CPTC information is detected and analyzed so that the control word and signal for controlling illegal view/copy is output to IRD 231. The bit stream reproduced from DVCR 232 is decoded in IRD 231. The decoded bit stream is descrambled according to the control word and signal for controlling illegal view/copy output from DVCR 232 so that analog audio/video data is output to be displayed.

IRD 231 outputs the scrambled digital audio/video data and CPTC information to DVCR 232 to record them on cassette tape. The scrambled digital audio/video data and CPTC information output from IRD 231 is recorded on cassette tape and recopied in DVCR 223.

Here, reproduction and recopy are made possible by the data stored in the permissible generational field, present generational field, reproducible number field, and maximum reproducible time field contained in the CPTC information.

Referring to Fig. 19, still another embodiment of the program receiving portion is to receive and process data transmitted through a recording medium, performing copy protection (CP).

As shown in Fig. 19, the third embodiment of the program receiving portion comprises a DVCR 243 for reproducing the scrambled digital audio/video data and CPTC information recorded on cassette tape through a recording medium, and outputting them to IRD 242, an IRD 242 for decoding/descrambling the bit stream transmitted from DVCR 243, and outputting analog audio/video data to be displayed, and a smart card 241 for decrypting the bit stream output from IRD 242, detecting/analyzing the CPTC, and outputting the control word and signals for controlling copying to IRD 222 to thereby perform CP. The operation of the third embodiment of the program receiving portion will be explained below.

In case that the bit stream is received via a recording medium, that is, through rental tape, the reproduced audio/video data is scrambled digital audio/video data.

The scrambled digital audio/video data and CPTC information reproduced from DVCR 243 are decoded in IRD 242 and decrypted in smart card 241. The CPTC information is detected and analyzed so that the control word and signal for controlling copying are output to IRD 242.

IRD 242 descrambles the decoded bit stream using the CPTC information output from smart card 241 and signals for controlling copying so that analog audio/video data is output to be displayed.

IRD 242 outputs the scrambled digital audio/video data and CPTC information to DVCR 243 in order to record them on cassette tape. The scrambled digital audio/video data and CPTC information output from IRD 242 are recorded on cassette tape in DVCR 243.

Here, reproduction and recopy are made possible by the data stored in the permissible generational field, present generational field, reproducible number field, and maximum reproducible time field contained in the CPTC information.

Referring to Fig. 20, yet another embodiment of the program receiving portion is to receive and process data transmitted through a recording medium, performing conditional access and CP. This embodiment is made in such a manner that in case of using the same CPTC information as the broadcasting medium, the smart card is commonly used.

As shown in Fig. 20, the fourth embodiment of the program receiving portion comprises a DVCR 253 for reproducing the scrambled digital audio/video data and CPTC information recorded on cassette tape through a recording medium, and outputting them to IRD 252, an IRD 252 for decoding/descrambling the bit stream transmitted from DVCR 253, and outputting analog audio/video data to be displayed, and a smart card 251 for decrypting the bit stream output from IRD 252, detecting/analyzing the CPTC, and outputting the control word and signals for controlling copying to IRD 252 to thereby perform CA and CP. The operation of the third embodiment of the program receiving portion will be explained below.

In case that the bit stream is received via a recording medium, that is, through rental tape and the DVCR, the reproduced audio/video data is scrambled digital audio/video data.

The scrambled digital audio/video data and CPTC information reproduced from DVCR 253 are decoded in IRD 252 and decrypted in smart card 251. The CPTC information is detected and analyzed so that the control word and signal for controlling copying are output back to IRD 252.

IRD 252 descrambles the decoded bit stream using the CPTC information output from smart card 251 and signals for controlling illegal view/copy so that analog audio/video data is output to be displayed.

IRD 252 outputs the scrambled digital audio/video data and CPTC information to DVCR 253 in order to record them on cassette tape. The scrambled digital audio/video data and CPTC information output from IRD 222 are recorded on cassette tape in DVCR 253.

Here, reproduction and recopy are made possible by the data stored in the permissible generational field, present generational field, reproducible number field, and maximum reproducible time field contained in the CPTC information.

IRD 222, 242, or 252 shown in Fig. 17, 19 or 20 is made in the following configuration as shown in Fig. 21.

Referring to Fig. 21, IRD 222, 242 or 252 comprises a recording/digital output controller 262 for decoding the bit stream transmitted from the broadcasting medium and DVCR, outputting to smart card 221, receiving the control word and signals for controlling illegal view/copy output from smart card 221, and controlling the output of the scrambled digital audio/video data for the purpose of recording and displaying; a descrambler 263 for descrambling the scrambled digital audio/video data output from recording/digital output controller 262 according to the control word output from recording/digital output controller 262, and a display processing portion 264 for processing and outputting the digital audio/video data output from descrambler 263 to be displayed. Here, DVCR 265 performs reproduction mainly. DVCR 223 of the program receiving portion of Fig. 18 combines recording therewith. The operation of IRD 266 will be described below.

The signal output to smart card 261 from recording/digital output controller 262 of IRD 266 is ECM, EMM and CPTC information. The signals output from smart card 261 to IRD 266 are the control word used to descramble and display the bit stream, and a signal for controlling copy protection.

Recording/digital output controller 262 communicates with the smart card, performs recording according to the signals of copy protection, outputs them to the digital output port in order to record them in another set, and outputs the control word and bit stream to descrambler 263.

When output to the recording/digital output port, updated ECM, EMM and CPTC information are output in addition to the scrambled data from recording/digital output controller 262 so that a copy different from the original script, that is, the broadcast or rental tape.

The ECM, EMM and CPTC are transmitted in various combinations. For the first combination, the ECM, EMM and CPTC are independently combined. The second combination is that the CPTC is included in the ECM and the EMM is independently combined. The third is that the CPTC is included in the EMM and the ECM is independently combined.

IRD 231 and DVCR 232 of Fig. 18 use the smart card, and additionally requires a CPTC detection and processing portion in the DVCR, which will be shown in Fig. 22.

DVCR 232 comprises a CPTC detecting/processing portion 276 for detecting/analyzing the CPTC information from the bit stream transmitted from recording medium 209, and outputting the control word and signals for illegal view/copy, and a reproducing portion 277 for reproducing the bit stream transmitted from recording medium 209 and outputting it to the IRD.

IRD 231 comprises a digital output controller 272 for receiving the control word and signals for controlling

illegal view/copy output from CPTC detecting/processing portion 276, and controlling the output of the scrambled digital audio/video data output from reproducing portion 277 in order to display them, a descrambler 273 for descrambling the scrambled digital audio/video data output from digital output controller 262 according to the control word output from digital output controller 262, and a display processing portion 274 for processing and outputting the digital audio/video data output from descrambler 273 in order to display them. The operation of IRD 276 and DVCR 275 will be described below.

CPTC detecting/processing portion 276 operates separately when reproducing portion 277 reproduces the scrambled data so that the CPTC information is detected from the cassette tape.

IRD 276 receives the scrambled data, CPTC information and control word from CPTC detecting/processing portion 276 and reproducing portion 277 from DVCR 275. Therefore, for normal descrambling, the scrambled data and control word are supplied to scramblar 273 from digital output controller 272. To the digital output port, only the scrambled data is output. For this reason, in case that the reproduced data is scrambled, copying is made impossible, and vise versa.

Commonly, in order to control tape copying, the depth of generation copy and the reproduction of tape to be copied are used together. As shown in Fig. 7, this yields the effect of controlling the number of copiable tape.

However, in order to allow copying tape to be reproducible as many as a predetermined number or for a predetermined time, it is necessary to perform communication between the smart card and DVCR.

Referring to Fig. 23, tape state information such as the reproduction number of the current tape is transmitted to smart card 261 from DVCR 265. In order to erase the tape, an enable erase signal is transmitted to DVCR 265 from smart card 261, and the erase head of the DVCR operates.

For tape erasing methods, the whole area of tape is erased by the full-width erase head, or only the control track is erased using the control head. In case that the CPTC is contained in the EMM, signals are input and output between the DVCR and smart card.

As the signals input to IRD 266, there are a broadcasting signal transmitted from a broadcasting medium and a signal reproduced from DVCR 265. The broadcasting signal input to IRD 266 is the scrambled digital data and a control signal having the EMM, ECM and CPTC information. The EMM and ECM are required for CA, the CPTC for copyright protection.

The scrambled digital data is input to descrambler 263. The control signal is input to smart card 261 for performing CA and CP. Using the control signal, smart card 261 restores control word CW and outputs it to descrambler 263. Descrambler 263 descrambles it using the control word.

The ECM output from smart card 261 is output to DVCR 265 or to an external port. This ECM is updated from the ECM input for copyright protection. The output disable signal output from smart card 261 is a signal to instruct IRD 266 to prohibit recording or copying. This signal is input to recording/digital output controller 262. The tape state signal is output to smart card 261 from DVCR 265 in order to inform the state of tape.

The signal output to DVCR 265 from smart card 261 for the purpose of a predetermined-number reproduction or predetermined-time reproduction is an erase enable signal. The signal for allowing recorded and copied tape to be reproducible even though the EMM information of the smart card is changed is an ID signal.

The ID signal is mapped and stored with corresponding EMM in the lookup table of smart card 261. If necessary, the EMM corresponding to the ID signal is output.

As shown in Fig. 24, the smart card comprises an ECM filter 301 for filtering the ECM from the bit stream output from the IRD, a CPTC/tape state signal filter 302 for filtering the CPTC information and the tape state signal indicative of the state of tape from the bit stream output from the IRD, an EMM filter 303 for filtering the EMM from the bit stream output from the IRD, a lookup table 304 for, in case that the EMM is updated for copyright protection by a broadcasting station, storing the previous EMM containing information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction in order to continuously reproduce the program of cassette tape copied, an EMM processing portion 307 for processing the EMM using the EMM output from EMM filter 303 and lookup table 304 and the tape state signal output from CPTC/tape state signal filter 302, a CPTC processing portion 306 for processing the CPTC information using the signals output from CPTC/tape state signal filter 302 and EMM processing portion 307, and a CA processing portion 305 for outputting control word CW using the signals output from ECM filter 301 and EMM processing portion 307.

In case that the CPTC information is contained in the EMM, as shown in Fig. 25, smart card 221 comprises an ECM filter 311 for filtering the ECM from the bit stream output from the IRD, an EMM filter 312 for filtering the EMM containing the EMM from the bit stream output from the IRD, a tape state signal filter 313 for filtering the tape state signal output from the IRD, a lookup table 314 for, in case that the EMM is updated for copyright protection by a broadcasting station, storing the previous EMM containing information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction in order to continuously reproduce the program of cassette tape copied, an EMM processing portion 317 for processing the EMM using the EMM output from EMM filter 312 and lookup table 314 and the tape state signal output from tape state signal filter 313, a CPTC processing portion 316 for processing the CPTC information using the signals

output from EMM filter 312 and tape state signal filter 313, to thereby output ECM, enable erase signal and ID signal, and a CA processing portion 315 for outputting control word CW using the signals output from ECM filter 311 and EMM processing portion 317.

ECM filter 301 or 311, CPTC/tape state signal filter 302, EMM filter 303 or 312, and tape state signal filter 313 extract ECM, CPTC, tape state signal and EMM, respectively. CA processing portion 305 or 315 generates a control word and performs CA. EMM processing portion 307 or 317 outputs the EMM information to CA processing portion 305 or 315 and CPTC processing portion 306 or 316, and additionally stores the received EMM to the lookup table.

In case that the scrambled digital data and encoded CPTC information are recorded on tape and that the EMM information required to decode the CPTC information is changed, the reproduction of tape is made impossible. According to this fact, the previous EMM is stored in a memory such as the EEPROM of the smart card as shown in Figs. 13 and 14, which is the same as described before.

Specifically, the lookup table is divided into two fields and stores ID information and EMM information, as shown in Fig. 13. In recording and copying, the ID information is recorded on tape, as shown in Fig. 14 in order to select corresponding EMM from the ID information recorded in the reproduction of tape.

In other words, referring to Fig. 14, EMM processing portion 307 receives a recording/playback signal indicating that the current DVCR mode is recording or playback, ID, and tape state signal having information of reproduction number of tape, selects a proper EMM from the lookup table, outputs it to CPTC processing portion 306 or 316 and CA processing portion 305 or 315, and transmits the ID information for the purpose of recording and copying to record it on tape.

Referring to Fig. 11, CPTC processing portion 306 or 316 performs copyright protection for recording or copying. The CPTC information or ECM containing the CPTC information is input to output the output disable signal, enable erase signal, and the CPTC or ECM containing the CPTC.

In order to control generation copy, CPTC processing portion 306 or 316, in case that the permissible generation of the permissible generational field is greater than the present generation recorded on tape, the present generational field is increased by 1 and encrypted again. If not, the output disable signal is generated to prohibit recording and copying.

In order to control reproduction, in case that the reproducible number of tape is greater than the reproducible number of the reproducible number field or the maximum reproducible time of the maximum reproducible time field is greater than the current time, CPTC processing portion 306 or 316 generates enable erase signal to operate the erase head of the DVCR.

In case that time delay produced when the CPTC or the ECM containing the CPTC is encrypted again becomes a problem to solve, CPTC processing portion 306 or 316 transmits the current generation signal to the DVCR and records it on tape, not modifying the CPTC or the ECM containing the CPTC.

The illegal view/copy protection apparatus for a digital broadcasting system embodying the present invention has means for recording and reproducing the reproduction number information of tape in the DVCR in order to implement the predetermined-number reproducibility of recorded or copied tape. Here, the reproduction number information of tape is updated and recorded again during tape reproduction.

As shown in Fig. 26, the DVCR comprises a deck mechanism 406, a recording/reproducing portion 405 for recording digital data on cassette tape according to the deck mechanism and reproducing the digital data recorded on cassette tape, a reproduction number detecting/updating portion 401 for detecting/updating the reproduction number from the digital data reproduced from recording/reproducing portion 405, and outputting it to the IRD in order to rerecord it in recording/reproducing portion 405, a digital data processing portion 402 for processing the digital data reproduced from recording/reproducing portion 405, outputting it to the IRD, and outputting switching position information for recording and reproducing, a recording/playback switching portion 404 for outputting a switching signal for controlling the reproduction number, the reproduction of digital data and the recording of the updated reproduction number using the switching position information output from digital data processing portion 402, and an error correction encoder/decoder 403 for correcting the error of data output from digital data processing portion 402, and encoding and decoding the data to be output to digital data processing portion 402.

In order to update and rerecord the reproduction number information of tape during playback, the reproduction number information of tape is recorded using an encoding algorithm. Otherwise, the information is recorded as clear data not encoded.

The recording position of the reproduction number information of tape uses part of audio, control and video tracks. For error correction to the reproduction number information of tape, a repetition coding is employed. The operation of the DVCR will be described below.

When reproduced by recording/reproducing portion 405 with the cassette tape loaded on deck mechanism 406, the reproduced digital data is input to reproduction number detecting/updating portion 401 and digital data processing portion 402 so that its reproduction number is detected and the digital data is processed and output.

The reproduction number detected in reproduction number detecting/updating portion 401 is updated, that is, increased by 1, and applied to recording/reproducing portion 405.

Digital data processing portion 402 applies the reproduced digital data output from recording/reproducing

portion 405 to error correction encoder/decoder 403 to perform error correction, encoding and decoding. The result is output to the IRD to be displayed or recorded. At the same time, the switching position information is output to recording/reproducing switching portion 404 in order to output a switching signal.

The switching signal output from recording/reproducing switching portion 404 controls recording/reproducing portion, to thereby record the updated reproduction number output from reproduction number detecting/updating portion 401, that is, the reproduction number added by 1, on tape.

Recording/reproducing switching portion 404 controls the reproduction number, the reproduction of digital data recorded on tape, and the recording of the updated reproduction number.

In another method of implementing the predetermined-number reproducibility of recorded or copied tape, an identifier is given to all tape used for a user to record broadcast programs, and the identifier given to tape and the reproducibility number information of tape corresponding to the identifier are handled together in the smart card.

Here, the smart card has a memory device which can be updated, such as EEPROM. The identifier and corresponding reproducible number information are stored in the memory device. For every reproduction of tape, the reproducible number information is updated and whether to playback is determined.

In conclusion, the described embodiments have the following advantages.

First, by adding CPTC information to data supplied, and by allowing a digital program to be normally viewed only when a CPTC detecting/analyzing means and descrambling/decrypting means are present at the receiving stage, illegal viewing is prohibited.

Second, to enhance copyright protection, data recorded on cassette tape is always scrambled digital data, and its CPTC information is encrypted to be recorded on cassette tape. A code for prohibiting viewable data from being restored from the cassette tape only with the scrambled data and CPTC information, and allowing the data to be viewable is provided in a device excluding the cassette tape. Otherwise, restoring of viewable data is made possible only with the scrambled data and CPTC information, making illegal copy impossible.

Third, using a method of restoring the viewable data only with the scrambled digital data and CPTC, rental tape is made to supply tape. Otherwise, using a method of prohibiting the viewable data from being restored only with the scrambled digital data and CPTC, rental tape is made to supply tape and smart card peculiar to a program provider as one set. Using the smart card for broadcasting medium, the rental tape is made to prohibit the viewable data from being restored only with the scrambled digital data and CPTC. Among the three methods of supplying tape only, one method is selected. Digital hardware for reproducing the data outputs only the scrambled digital data to an external port, making impossible the restoring of viewable data from the output data, without the smart card.

Fourth, the described embodiment prohibits illegal recording and copying of a program protected by copyright law, collects fee for recording or copying, and freely controls the reproducible number of copied tape which can be made from a program supplied by a program supplier, protecting copyright.

Fifth, the described embodiment can be used as a copyright protection system having a high security and multifunction with respect to a program through a broadcasting medium such as satellite and terrestrial broadcastings, or, at the same time, as a copy protection system having a high security to a program through a recording medium such as rental tape.

Sixth, the descibed embodiment is employed to digital hardware such as broadcasting receiver and digital VCR, to thereby perfectly protect a program supplier's copyright and activates digital media because of various software supplied through the digital media.

Claims

1. An illegal view/copy protection method for a digital broadcasting system comprising:

   an audio/video signal transmission step for multiplexing and transmitting audio/video bit stream scrambled in control words and information where the control words and CPTC information for illegal view/copy protection are encrypted; and

   an audio/video reception step for decrypting the transmitted bit stream to analyze the CPTC information and control words, deciding whether recording is allowed or not to be recorded on cassette tape, and using the control words, performing descrambling and decoding to output audio/video signals to a monitor.

2. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1, wherein said CPTC information is formatted in a generational copy control field for limiting the number of copy available, and a reproducibility control field for limiting the reproduction of a copied program.

3. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 2, wherein said CPTC information is formatted further containing a descrambling information field where part of the control words for descrambling are recorded.

4. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 2, wherein said CPTC information is formatted further contain-

ing a CA field where CA information for conditional access is recorded.

5. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 2, wherein said generational copy control field is made up of a permissible generational field for limiting the number of copy permissible and a present generational field for indicating the present generation of a program copied.

6. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 2, wherein said reproduction control field is made up of a reproducible number field for limiting the number of reproducing a copied program, and a maximum reproducible time field for limiting time to reproduce the copied program.

7. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1, wherein the data recorded on cassette tape contains scrambled audio/video bit stream and CPTC information.

8. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 7, wherein said CPTC information is overwritten on the scrambled audio/video bit stream for the error effect and recorded on cassette tape.

9. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 7, wherein said CPTC information is recorded on a portion of any of the audio track of cassette tape, the control track of cassette tape, or the video track of cassette tape.

10. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1, wherein said audio/video signal transmission step comprises the steps of: encoding the audio/video bit stream;

generating a control word for scrambling;
scrambling for the encoded audio/video bit stream using the generated control word;
generating CPTC information for illegal view/copy protection;
encrypting for encrypting the control word and CPTC information; and
multiplexing and transmitting the scrambled audio/video bit stream and encrypted CPTC information.

11. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1, wherein said audio/video signal transmission step comprises the steps of:

encoding the audio/video bit stream;
generating a control word for scrambling;
scrambling for the encoded audio/video bit stream using the generated control word;
generating CPTC information for illegal view/copy protection;
generating conditional access information for conditional reception;
encrypting for encrypting the CPTC information and CA information; and
multiplexing and transmitting the scrambled audio/video bit stream and encrypted CPTC information and conditional access information.

12. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1 or claim 11, wherein said audio/video reception step comprises the steps of:

filtering the transmitted bit stream and decrypting the CPTC information;
analyzing the CPTC information to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information;
deciding whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmitted bit stream on cassette tape; and
descrambling and decoding the transmitted bit stream in the control word and outputting an audio/video signal.

13. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 12, wherein said all of the control word is contained in the CPTC information.

14. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 1, wherein said bit stream transmitted contains ECM and EMM.

15. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 14, wherein said audio/video reception step comprises the steps of:

filtering the transmitted bit stream and decrypting the CPTC information and control word;
filtering the control word;
analyzing the CPTC information to generate a control word and a signal for controlling the protection of copyright and to update the CPTC information;
deciding whether to allow recording according to the signal for controlling the protection of copyright to record the scrambled and transmit-

ted bit stream on cassette tape; and
descrambling and decoding the transmitted bit stream in control words and outputting an audio/video signal.

16. An illegal view/copy protection method for a digital broadcasting system as claimed in any of claims 12, 14 or 15, wherein said CPTC information analyzing step comprises the steps of:

generating a control word;
detecting a permissible generation of a permissible generational field for limiting the available number of copy of a program of the CPTC information and the present generation of the present generational field indicating the present generation of the program copied, to thereby perform copy-impossible and update the CPTC information; and
detecting the reproducible number of the reproducible number field for limiting the number of reproduction of copied programs of the CPTC information, the maximum reproducible time of the maximum reproducible time field for limiting time to reproduce the copied program, and the number and time of reproduction of tape, to thereby process reproduction-impossible.

17. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 12 or claim 16, wherein said copying number limiting step comprises the steps of:

comparing the permissible generation of the permissible generational field and the present generation of the present generational field and deciding whether the permissible generation is below the present generation;
if the permissible generation is below the present generation, generating an output disable signal to make copying impossible and destroying the control word; and
if the permissible generation is not below the present generation, increasing the present invention by '1' and recording the result on cassette tape.

18. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 17, wherein said copying number limiting step further comprises the step of, if the permissible generation is not below the present generation, updating the CPTC information.

19. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 16 or claim 17, wherein said reproduction limiting step comprises the steps of:

comparing the reproducible number of the reproducible number field and the reproduction number of tape and deciding whether the reproducible number is below the reproduction number of tape;
if the reproducible number is not below the reproduction number of tape, comparing the maximum reproducible time and reproduction time of tape, and deciding whether the maximum reproducible time is below the reproduction time of tape;
if the maximum reproducible time is not below reproduction time of tape, turning off an enable erase signal to thereby enable the copied program to be reproduced; and
if the reproducible number is below the reproduction number of tape or the maximum reproducible time is below the reproduction time of tape, turning on the enable erase signal to make the reproduction of the copied program impossible so that part of or the whole program recorded on cassette tape is erased.

20. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 14 or claim 15, wherein part of the control word is contained in the CPTC information.

21. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 20, wherein the remainder of the control word is contained in the ECM.

22. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 20, wherein the remainder of the control word is contained in the EMM.

23. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 14 or claim 15, wherein the whole control word is contained in the ECM.

24. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 14 or claim 15, wherein the whole control word is contained in the EMM.

25. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 14, further comprising a reproduction and rerecording step of: decrypting the bit stream recorded and reproduced on cassette tape, analyzing the CPTC information, deciding whether to allow rerecording, recording the result on cassette tape, filtering the control word, and performing descrambling and decoding to output an audio/video signal.

26. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 25, wherein said audio/video reproduction and rerecording step comprises the steps of:

filtering the bit stream recorded and reproduced on video tape, and decrypting the CPTC information;
analyzing the CPTC information to generate control words and a signal for controlling the protection of copyright and update the CPTC information;
deciding whether to allow recording according to the signal of controlling the protection of copyright, and recording the scrambled and transmitted bit stream on cassette tape; and
descrambling and decoding the transmitted bit stream in control words to output an audio/video signal.

27. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 26, wherein said audio/video reproduction and rerecording step comprises the step of deciding whether to allow post-reproduction according to the signal for controlling the protection of copyright to thereby erase part of or the whole data recorded on cassette tape.

28. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 25, wherein said EMR contains information required for decoding information

29. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 28, further comprising the step of storing and processing EMM in which, in case that the EMM is updated by a broadcasting station for the purpose of copyright protection, the EMM having information required to decode the CPTC information is stored in order to continuously reproduce programs of copied cassette tape.

30. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 29, wherein an ID number indicative of updating the EMM is recorded on said cassette tape.

31. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 30, wherein the EMM is stored to which the updating state and the ID number of cassette tape are mapped.

32. An illegal view/copy protection method for a digital broadcasting system as claimed in claim 31, wherein said EMM storing and processing step comprises the steps of:

storing all EMM to be updated and corresponding ID information;
selecting the latest EMM in recording cassette tape;
recording a corresponding ID number; and
selecting an EMM corresponding to the ID number recorded on cassette tape in reproducing the cassette tape.

33. An illegal view/copy protection apparatus for a digital broadcasting system comprising:

a program producing portion for multiplexing information encrypted both with the control word for scrambling and the CPTC information for prohibiting illegal view/copy, and the audio/video bit stream scrambled in control words, to thereby make a program;
a distribution medium portion for distributing programs made in said program producing portion through a transmission medium; and
a program receiving portion for detecting and analyzing the CPTC information from the bit stream transmitted from said distribution medium portion and the bit stream reproduced from cassette tape, and descrambling and decoding the bit stream transmitted from said distribution medium portion.

34. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein said program producing portion comprising:

a control word generator for generating a control word for scrambling;
a CPTC generator for generating the CPTC information for prohibiting illegal view/copy;
a scrambling portion for scrambling the audio/video bit stream using the control word output from said control word generator;
an encrypting portion for encrypting the control word output from said control word generator and the CPTC information output from said CPTC generator; and
an adder for multiplexing the signals output from said scrambling portion and encrypting portion and transmitting them to said distribution medium portion.

35. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein said distribution medium portion comprises:

a broadcasting medium for distributing the program made by said program producing portion through cable, satellite or terrestrial broadcast-

ing; and
a recording medium for distributing the program made by said program producing portion through cassette tape.

36. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 35, wherein said program receiving portion comprises:

a decrypting portion for decrypting the bit stream transmitted from said broadcasting medium;
a CPTC detecting/analyzing portion for detecting and analyzing the CPTC information from the bit stream output from said decrypting portion and recording medium, and outputting signals for controlling the control word and illegal view/copy;
a descrambling portion for descrambling the bit stream transmitted from said broadcasting medium and recording medium and the bit stream reproduced from cassette tape;
a decoding portion for decoding and displaying the signal output from said descrambling portio; and
a recording/reproducing portion for recording the bit stream transmitted from said broadcasting medium and recording medium according to the signal output from said CPTC detecting/analyzing portion, and reproducing cassette tape, to thereby output the result to said descrambling portion and CPTC detecting/analyzing portion.

37. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein said CPTC information is formatted in a generational copy control field for limiting the number of copy available, and a reproducibility control field for limiting the reproduction of a copied program.

38. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 37, wherein said CPTC information is formatted further containing a descrambling information field where the whole or part of the control words for descrambling are recorded.

39. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 37, wherein said CPTC information is formatted further containing a CA field where CA information for conditional access is recorded.

40. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 37,wherein said generational copy control field is made up of a permissible generational field for limiting the number of copy permissible and a present generational field for indicating the present generation of a program copied.

41. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 37, wherein said reproduction control field is made up of a reproducible number field for limiting the number of reproducing a copied program, and a maximum reproducible time field for limiting time to reproduce the copied program.

42. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein the data recorded on cassette tape contains scrambled audio/video bit stream and CPTC information.

43. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 42, wherein said CPTC information is overwritten on the scrambled audio/video bit stream for the error effect and recorded on cassette tape.

44. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 42, wherein said CPTC information is recorded on a portion of any of the audio track of cassette tape, the control track of cassette tape, or the video track of cassette tape.

45. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein said all of the control word is contained in the CPTC information.

46. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 33, wherein said bit stream transmitted contains ECM and EMM.

47. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 46, wherein part of the control word is contained in the CPTC information.

48. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 47, wherein the remainder of the control word is contained in the ECM.

49. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 47, wherein the remainder of the control word is contained in the EMM.

50. An illegal view/copy protection apparatus for a dig-

ital broadcasting system as claimed in claim 46, wherein the whole control word is contained in the ECM.

51. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 46, wherein the whole control word is contained in the EMM.

52. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 46, wherein said program receiving portion comprises:

an IRD for receiving, decoding and descrambling the bit stream transmitted from said broadcasting medium, outputting analog audio/video data to be displayed and outputting scrambled digital audio/video data to be recorded on cassette tape; and
a smart card for decrypting the bit stream output from said IRD, detecting/analyzing the CPTC information, and outputting the control word and signals for controlling illegal view/copy to said IRD in order to perform conditional access and copy protection.

53. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 52,wherein said program receiving portion further comprises a lookup table for, in case that the EMM is updated by a broadcasting station for the purpose of copyright protection, storing EMM having information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction to said smart card in order to continuously reproduce the program of copied cassette tape.

54. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 52, wherein said program receiving portion further comprises a DVCR for recording the digital audio/video data and CPTC information scrambled and output from said IRD on cassette tape, and reproducing the scrambled digital audio/video data and CPTC information recorded on cassette tape to be output to said IRD.

55. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 54, wherein said smart card comprises:

an ECM filter for filtering the ECM from the bit stream output from said IRD;
a CPTC/tape state signal filter for filtering the CPTC information and the tape state signal indicative of the state of tape from the bit stream output from said IRD;
an EMM filter for filtering the EMM from the bit stream output from said IRD;
a lookup table for, in case that the EMM is updated for copyright protection by a broadcasting station, storing the previous EMM containing information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction in order to continuously reproduce the program of cassette tape copied;
an EMM processing portion for processing the EMM using the EMM output from said EMM filter and lookup table and the tape state signal output from said CPTC/tape state signal filter;
a CPTC processing portion for processing the CPTC information using the signals output from said CPTC/tape state signal filter and EMM processing portion; and
a CA processing portion for outputting control word CW using the signals output from said ECM filter and EMM processing portion.

56. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 54, wherein said smart card comprises:

an ECM filter for filtering the ECM from the bit stream output from said IRD;
an EMM filter for filtering the EMM containing the EMM from the bit stream output from said IRD;
a tape state signal filter for filtering the tape state signal output from said IRD;
a lookup table for, in case that the EMM is updated for copyright protection by a broadcasting station, storing the previous EMM containing information required to decode the CPTC information, and outputting CPTC information corresponding in reproduction in order to continuously reproduce the program of cassette tape copied;
an EMM processing portion for processing the EMM using the EMM output from said EMM filter and lookup table and the tape state signal output from said tape state signal filter;
a CPTC processing portion for processing the CPTC information using the signals output from said EMM filter and tape state signal filter, to thereby output ECM, enable erase signal and ID signal; and
a CA processing portion for outputting control word CW using the signals output from said ECM filter and EMM processing portion.

57. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 54, wherein said DVCR comprises:

a deck mechanism;
a recording/reproducing portion for recording digital data on cassette tape according to said deck mechanism and reproducing the digital data recorded on cassette tape;
a reproduction number detecting/updating portion for detecting/updating the reproduction number from the digital data reproduced from said recording/reproducing portion, and outputting it to said IRD in order to rerecord it in said recording/reproducing portion;
a digital data processing portion for processing the digital data reproduced from said recording/reproducing portion, outputting it to said IRD, and outputting switching position information for recording and reproducing;
a recording/playback switching portion for outputting a switching signal for controlling the reproduction number, the reproduction of digital data and the recording of the updated reproduction number using the switching position information output from said digital data processing portion; and
an error correction encoder/decoder for correcting the error of data output from said digital data processing portion, and encoding and decoding the data to be output to said digital data processing portion.

58. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 35, wherein said program receiving portion comprises:

a DVCR for detecting/analyzing the CPTC information from the bit stream transmitted from said recording medium, outputting a control word and signals for controlling illegal view/copy, and reproducing scrambled digital audio/video data; and
an IRD for receiving the control word and signals for controlling illegal view/copy output from said DVCR 232, descrambling the scrambled digital audio/video data, and outputting analog audio/video data to be displayed or recorded.

59. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 58, wherein said DVCR comprises:

a CPTC detecting/processing portion for detecting/analyzing the CPTC information from the bit stream transmitted from said recording medium, and outputting the control word and signals for illegal view/copy; and
a reproducing portion for reproducing the bit stream transmitted from said recording medium and outputting it to said IRD.

60. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 59, wherein said IRD comprises:

a digital output controller for receiving the control word and signals for controlling illegal view/copy output from said CPTC detecting/processing portion, and controlling the output of the scrambled digital audio/video data output from said reproducing portion in order to display them;
a descrambler for descrambling the scrambled digital audio/video data output from said digital output controller according to the control word output from said digital output controller; and
a display processing portion for processing and outputting the digital audio/video data output from said descrambler in order to display them.

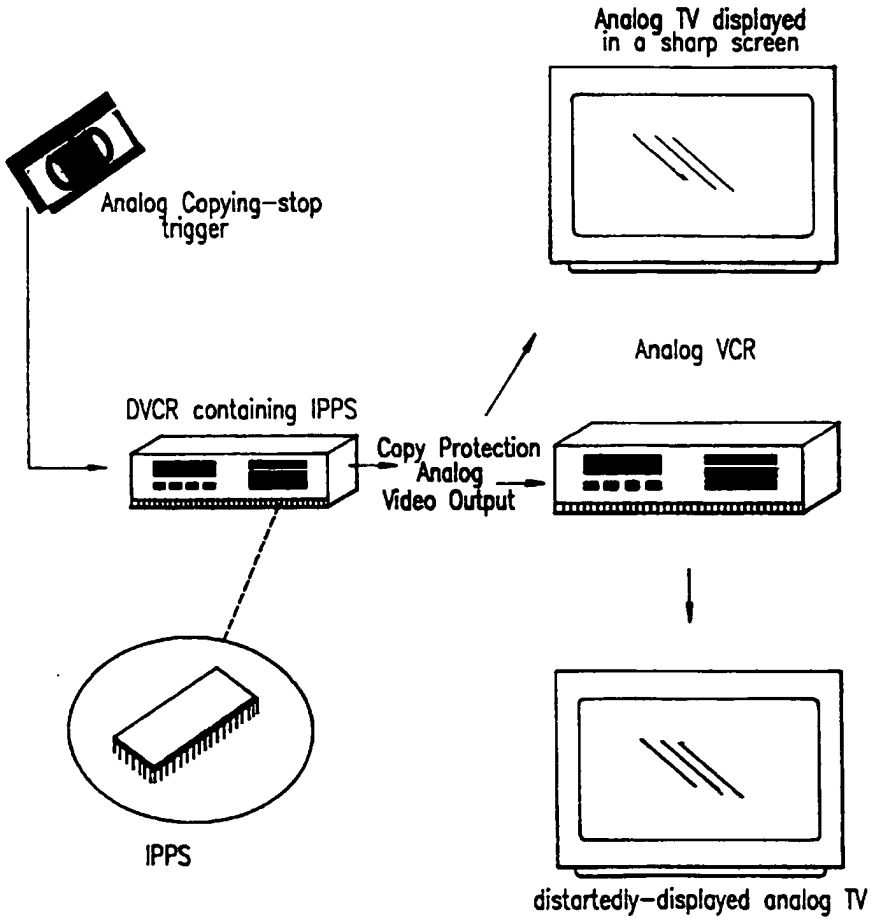
61. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 35, wherein said program receiving portion comprises:

a DVCR for reproducing the scrambled digital audio/video data and CPTC information recorded on cassette tape through a recording medium, and outputting them to said IRD;
an IRD for decoding/descrambling the bit stream transmitted from said DVCR, and outputting analog audio/video data to be displayed; and
a smart card for decrypting the bit stream output from said IRD, detecting/analyzing the CPTC, and outputting the control word and signals for controlling copying to said IRD to thereby perform copy protection and/or conditional access.

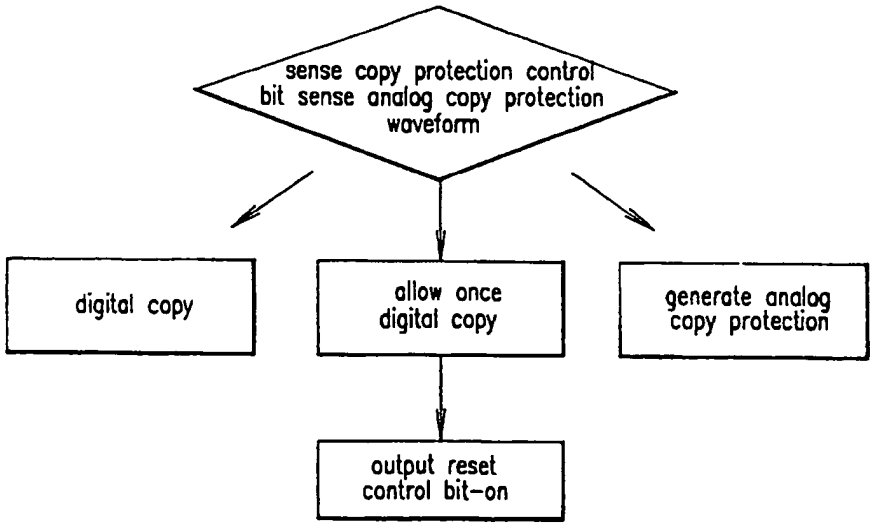
62. An illegal view/copy protection apparatus for a digital broadcasting system as claimed in claim 54 or claim 61, wherein said IRD comprises:

a recording/digital output controller for decoding the bit stream transmitted from the broadcasting medium and DVCR, outputting to said smart card, receiving the control word and signals for controlling illegal view/copy output from said smart card, and controlling the output of the scrambled digital audio/video data for the purpose of recording and displaying;
a descrambler for descrambling the scrambled digital audio/video data output from said recording/digital output controller according to the control word output from said recording/digital output controller; and
a display processing portion for processing and outputting the digital audio/video data output from said descrambler to be displayed.

F I G.1
(conventional art)



F I G.2
(conventional art)



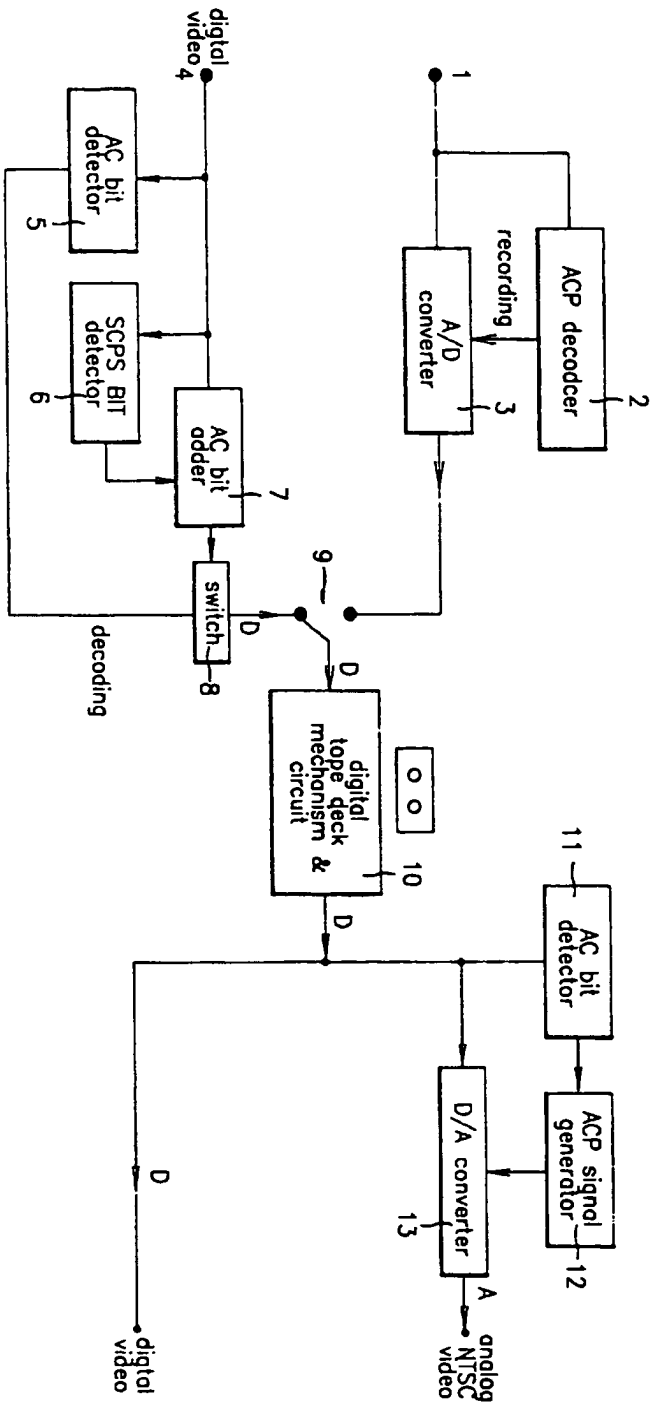


FIG.3

FIG.4

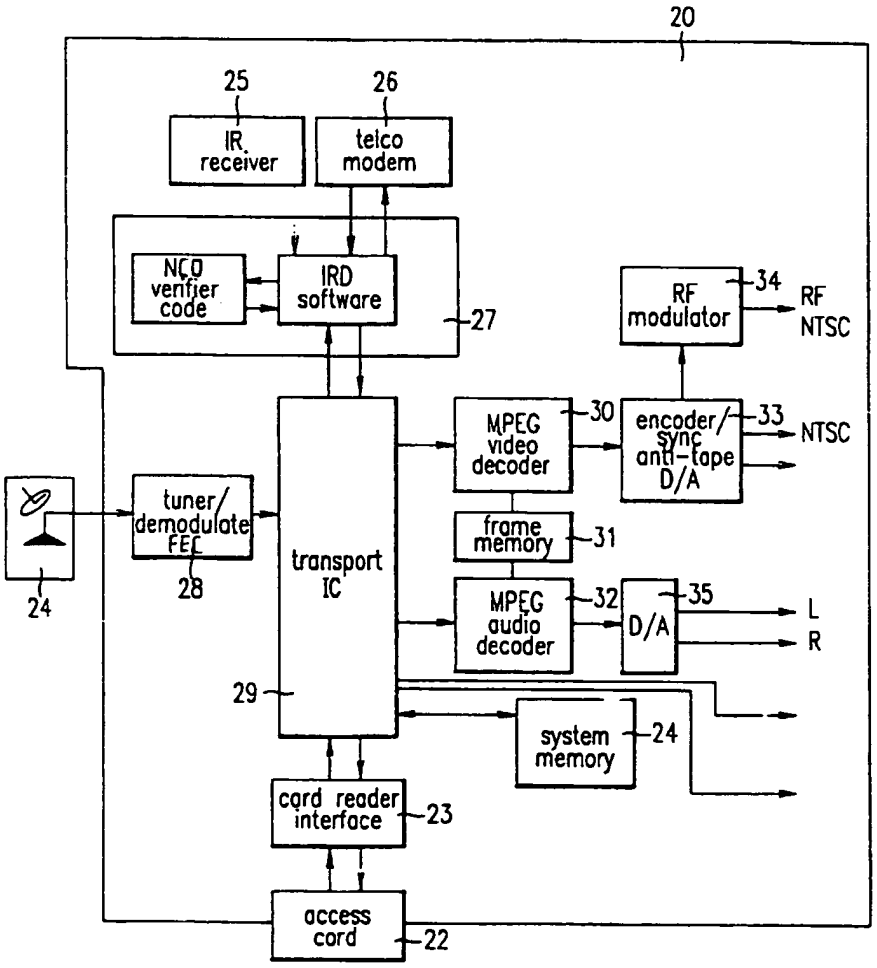


FIG.5

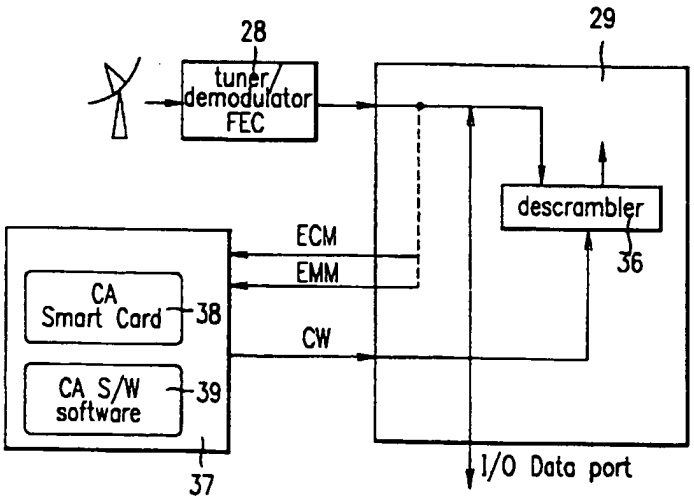


FIG.6a

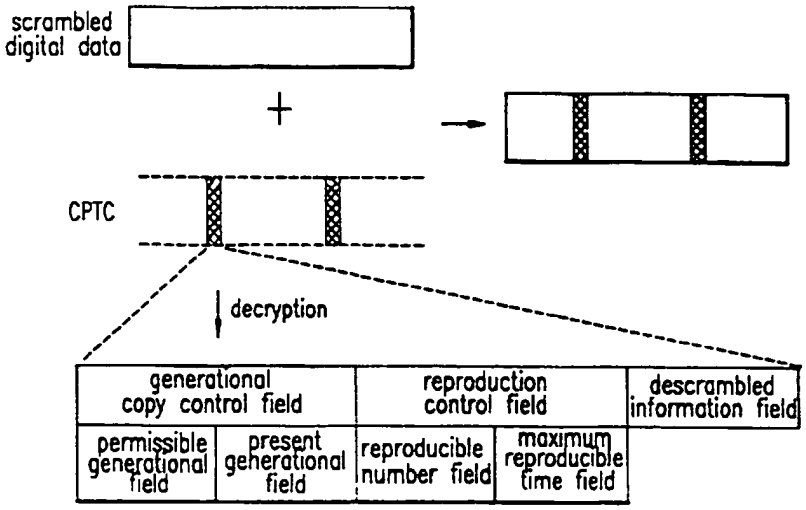


FIG.6b

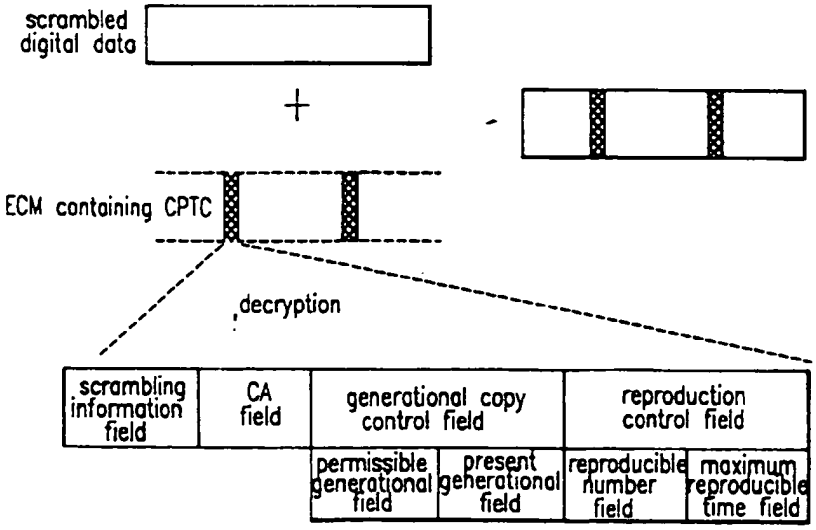


FIG.7

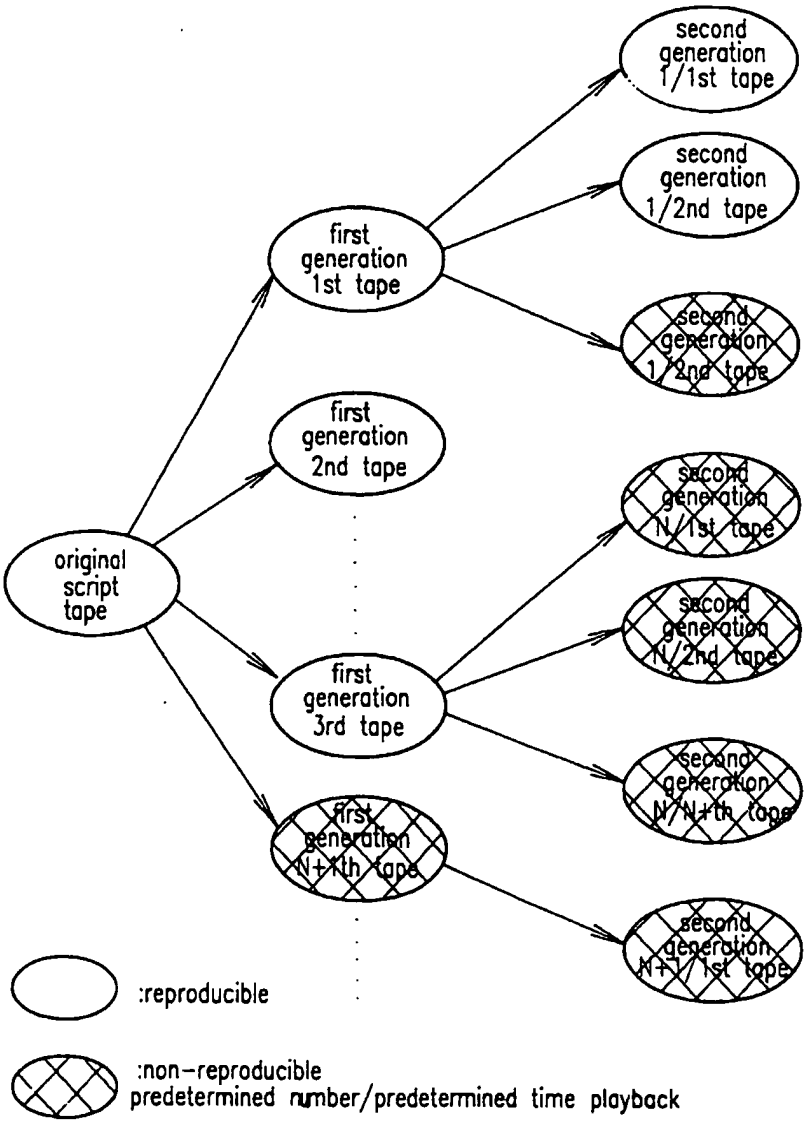


FIG.8a

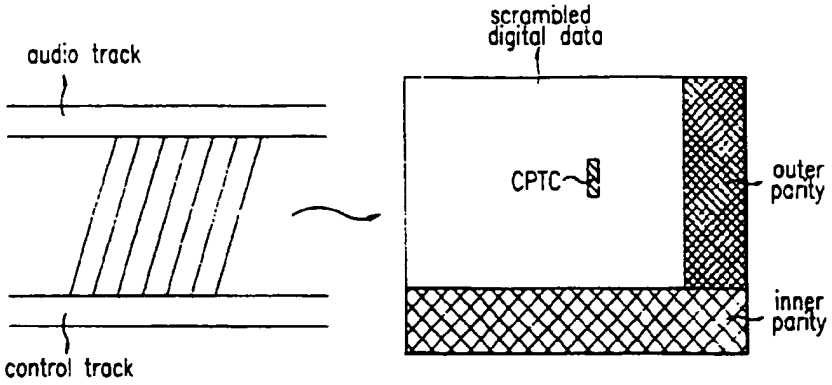


FIG.8b

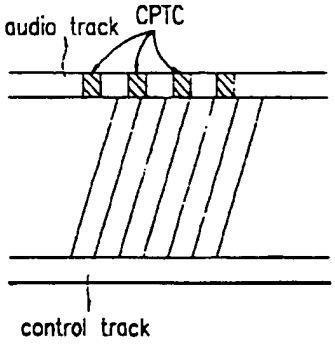


FIG.8c

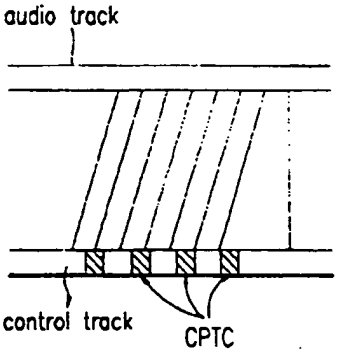
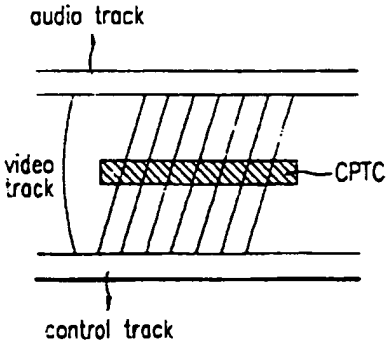


FIG.8d



F I G.9

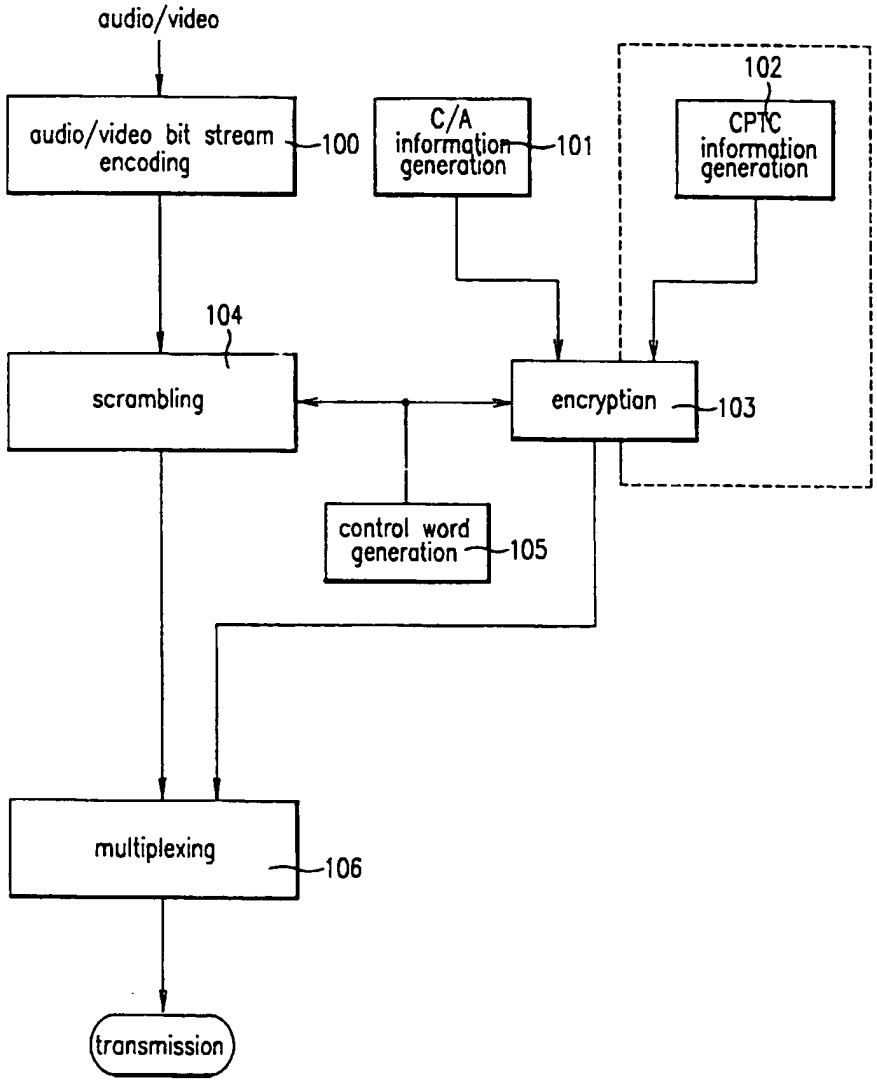
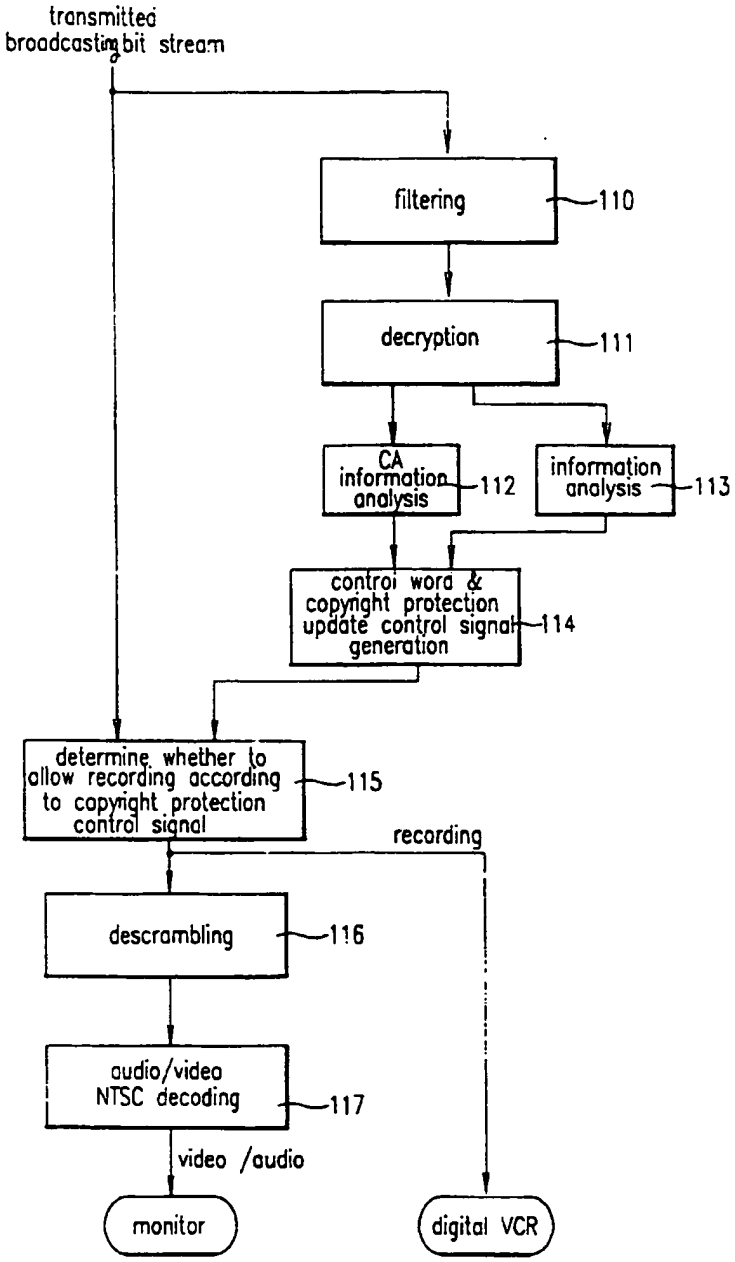
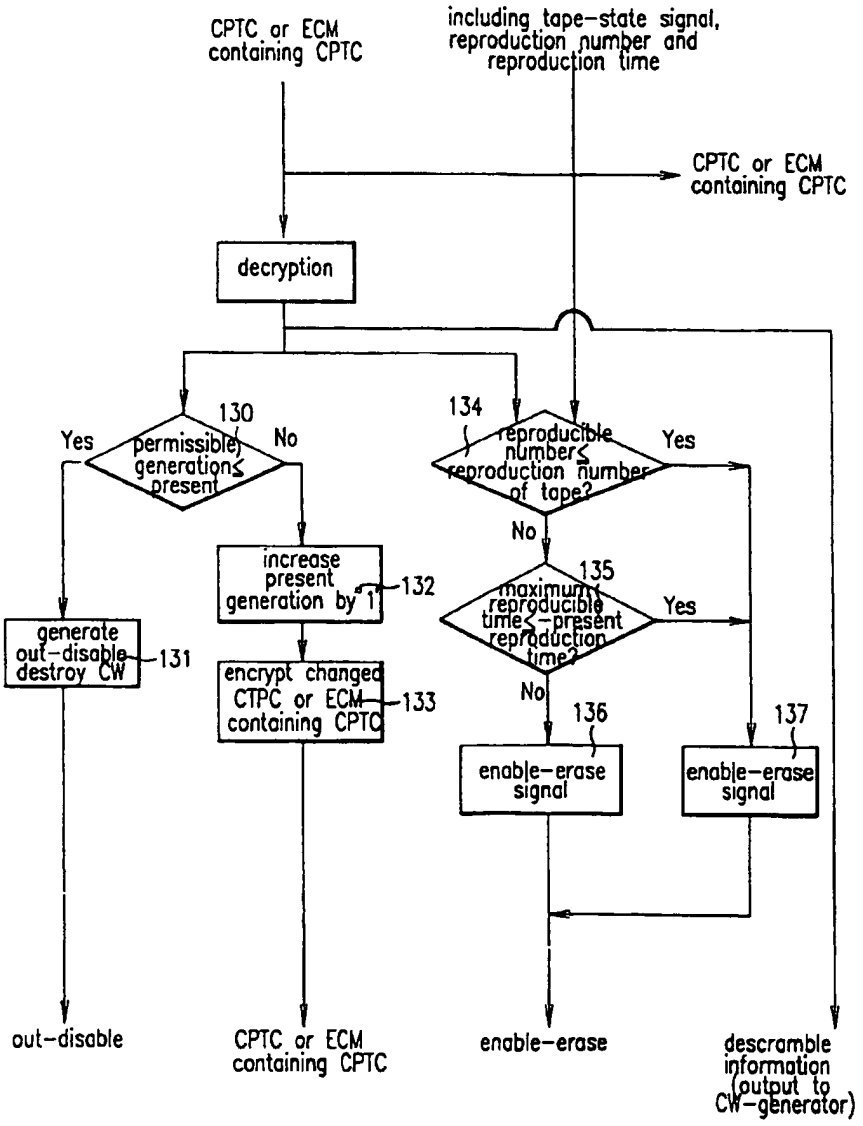


FIG.10



F I G.11



F I G.12

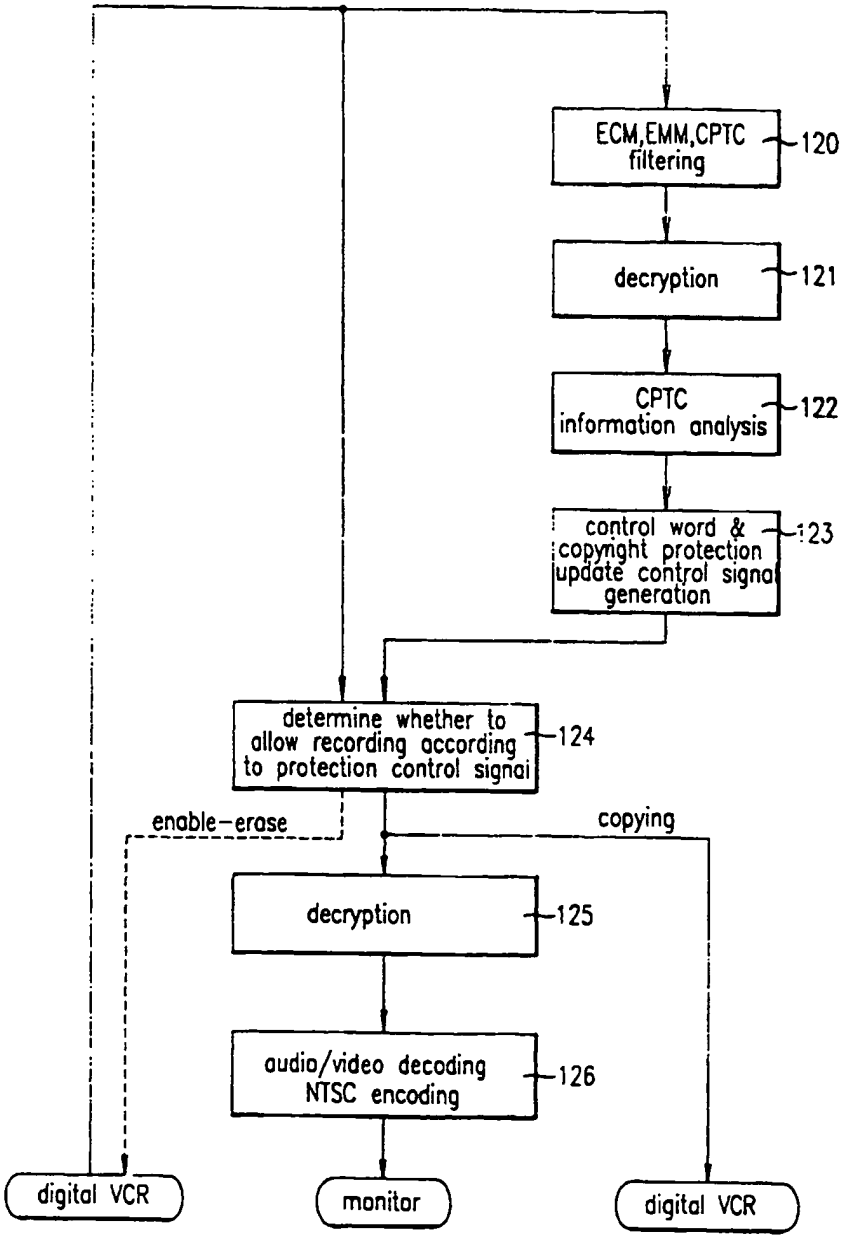


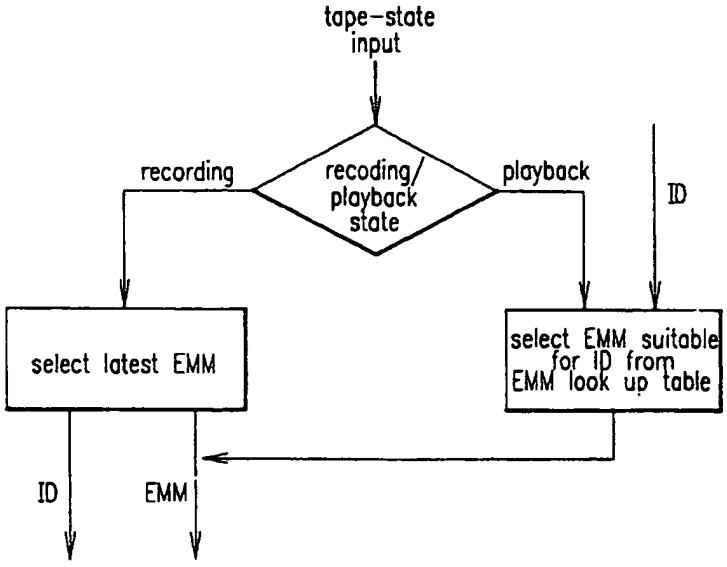
FIG.13

ID_1	EMM_1
ID_2	EMM_2
ID_3	EMM_3
. . .	. . .
ID_n	EMM_n

FIG.14

recording/reproduction state	ID	reproduction number

FIG.15



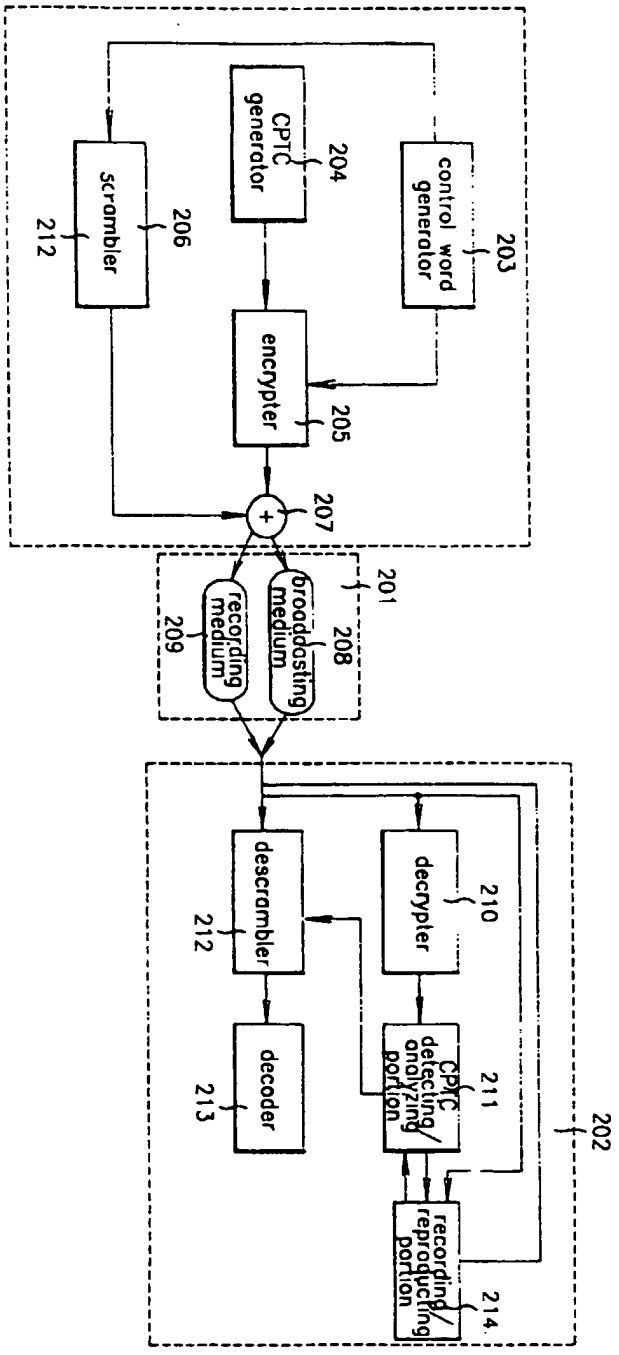


FIG.16

FIG.17a

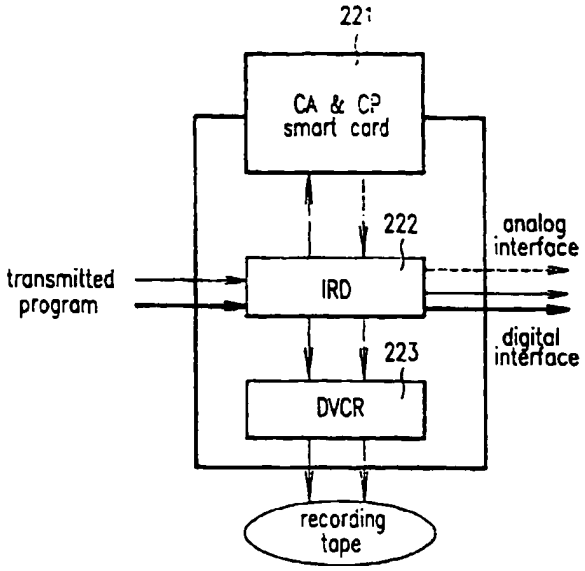


FIG.17b

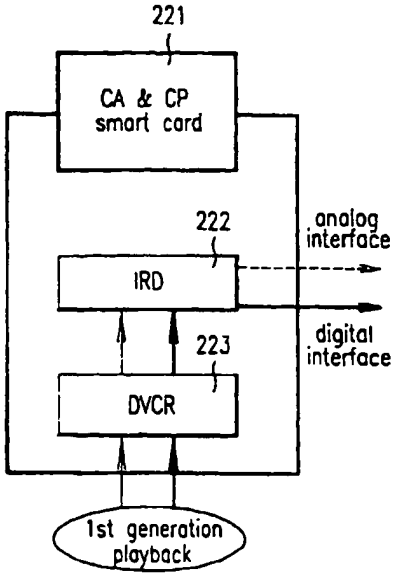


FIG.18

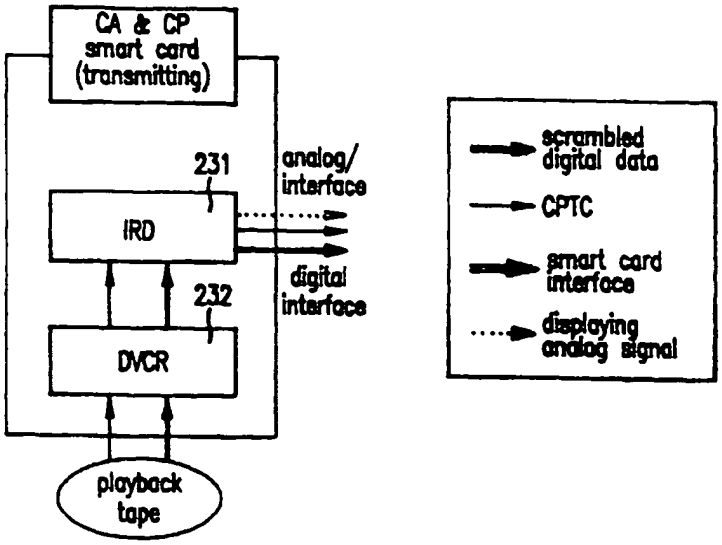


FIG.19

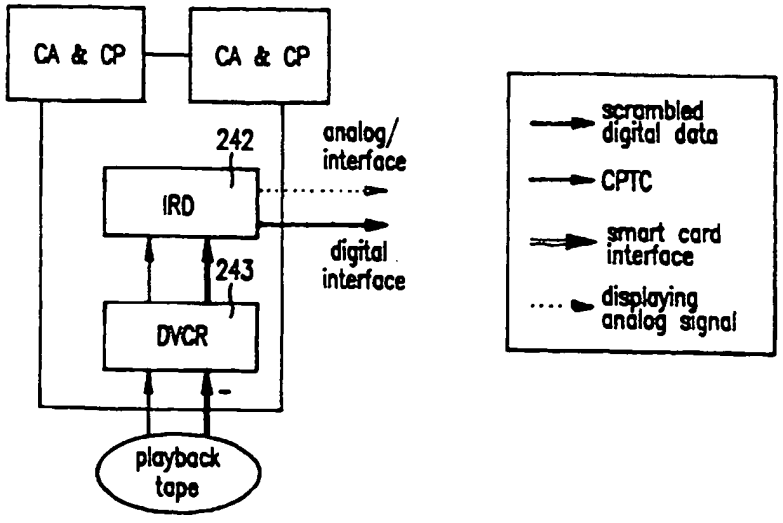


FIG.20

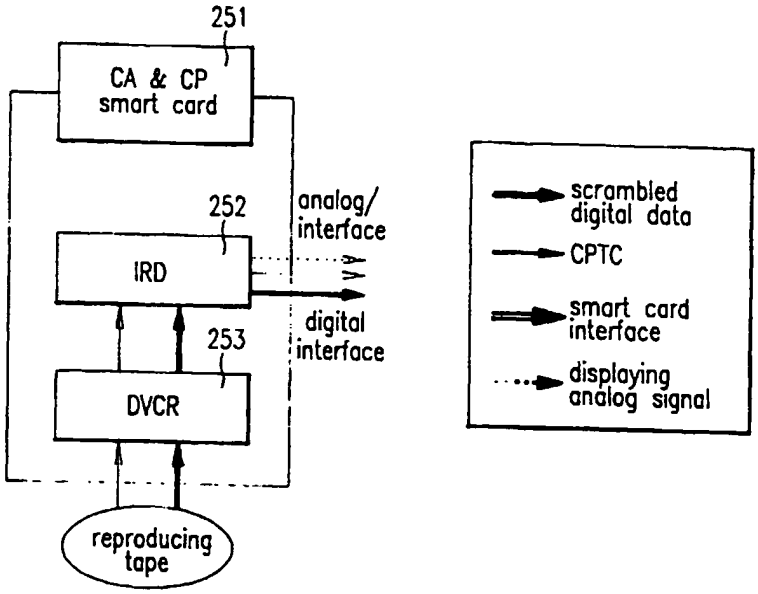


FIG.21

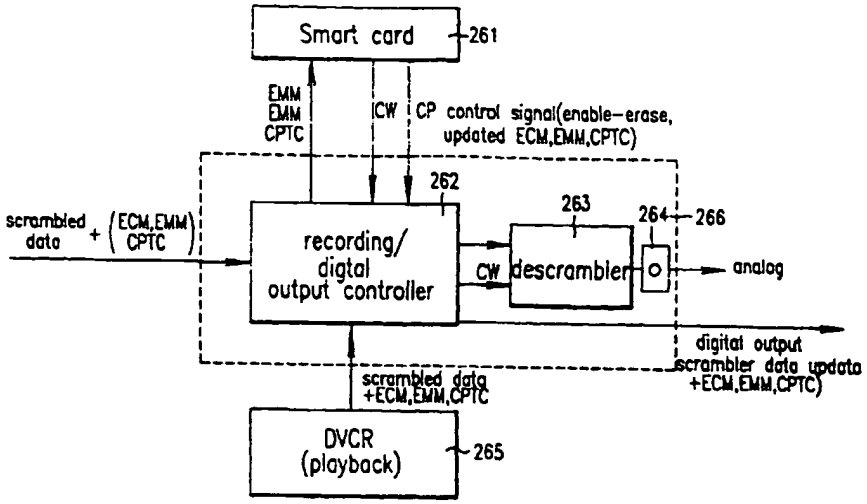


FIG.22

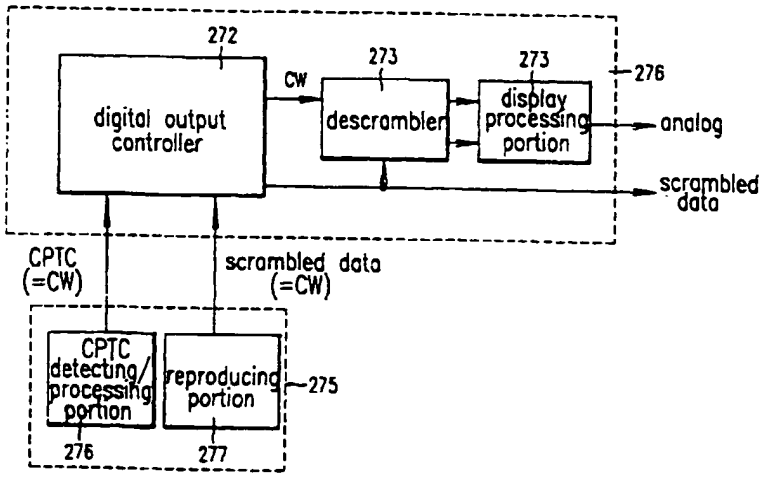


FIG.23

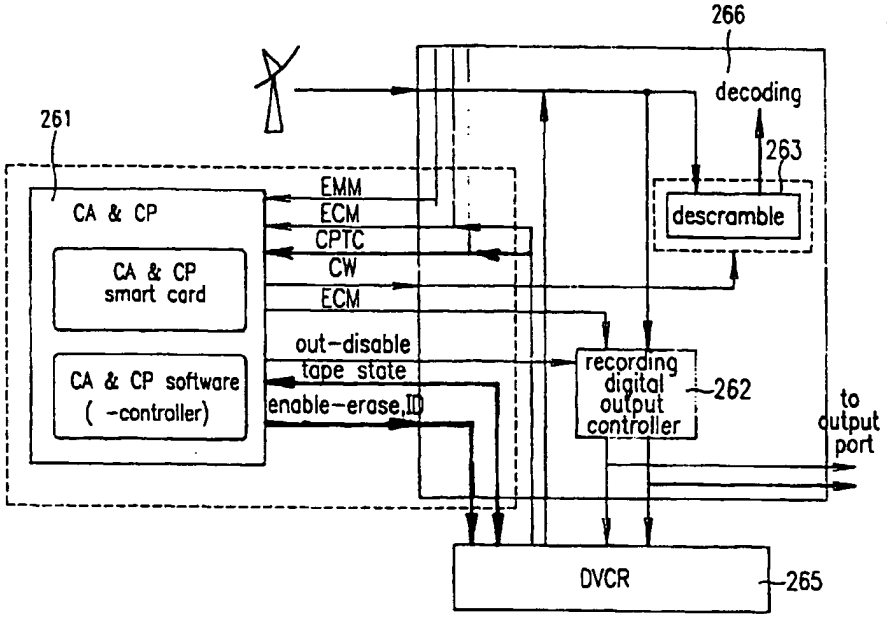
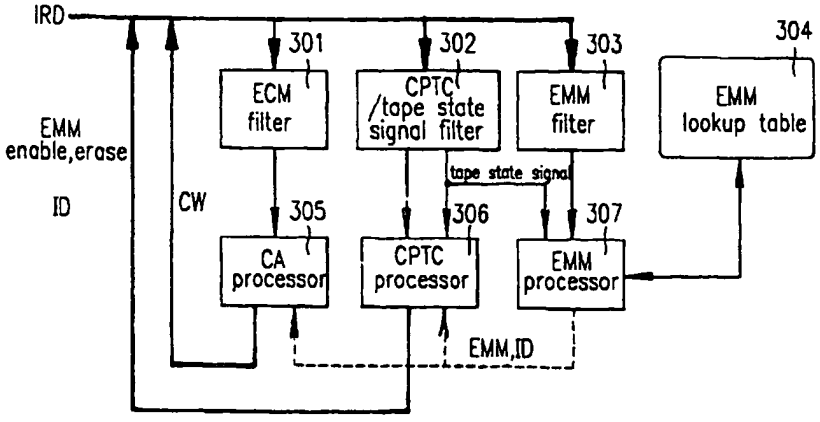
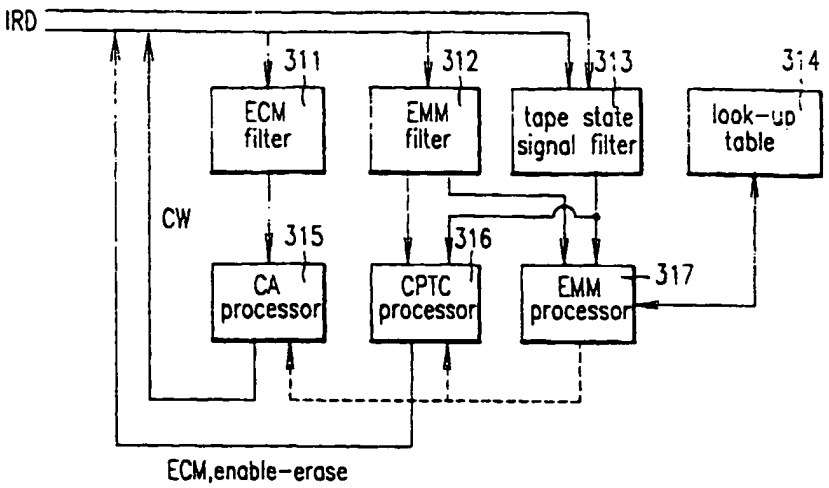


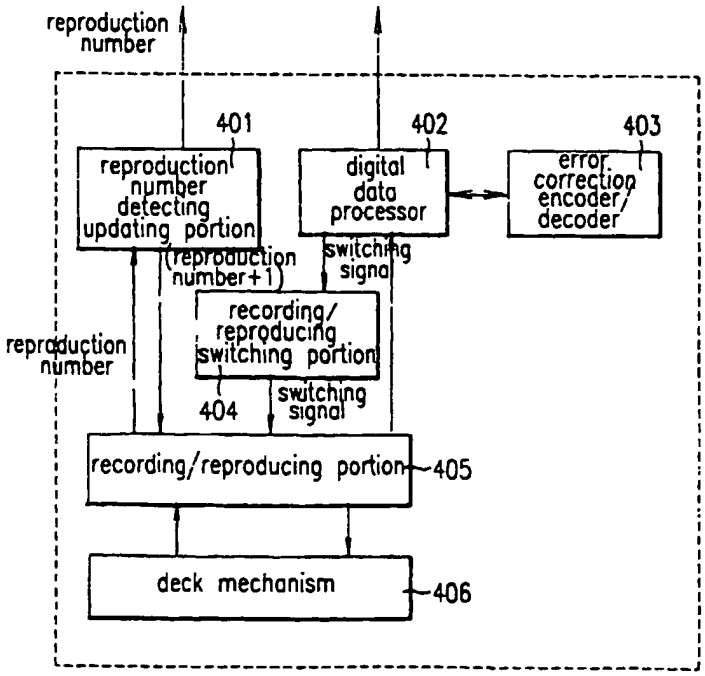
FIG.24



F I G.25



F I G.26



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

(11) EP 0 818 748 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.01.1998 Bulletin 1998/03

(51) Int Cl.[6]: **G06F 17/60**

(21) Application number: **97304946.3**

(22) Date of filing: **07.07.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE

(30) Priority: **08.07.1996 JP 178130/96**
21.05.1997 JP 130626/97

(71) Applicant: **Murakoshi, Hiromasa**
Koriyama-shi, Fukushima, 963 (JP)

(72) Inventor: **Kanno, Kazuhiro**
Koriyama-shi, Fukushima, 963-02 (JP)

(74) Representative:
Cross, Rupert Edward Blount et al
BOULT WADE TENNANT,
27 Furnival Street
London EC4A 1PQ (GB)

(54) Software management system and method

(57) An operation management system for managing the operation of a managed software product. When a management target function is executed, reference is made to a battery value and, if the value is zero or greater, the function is allowed to be executed. The battery value is decremented as the function is executed. A charge value is supplied on a charge disk, such as a floppy disk, to allow the user to increase the battery value and to extend the usage period of the managed software product. The charge value may be supplied over a communication line.

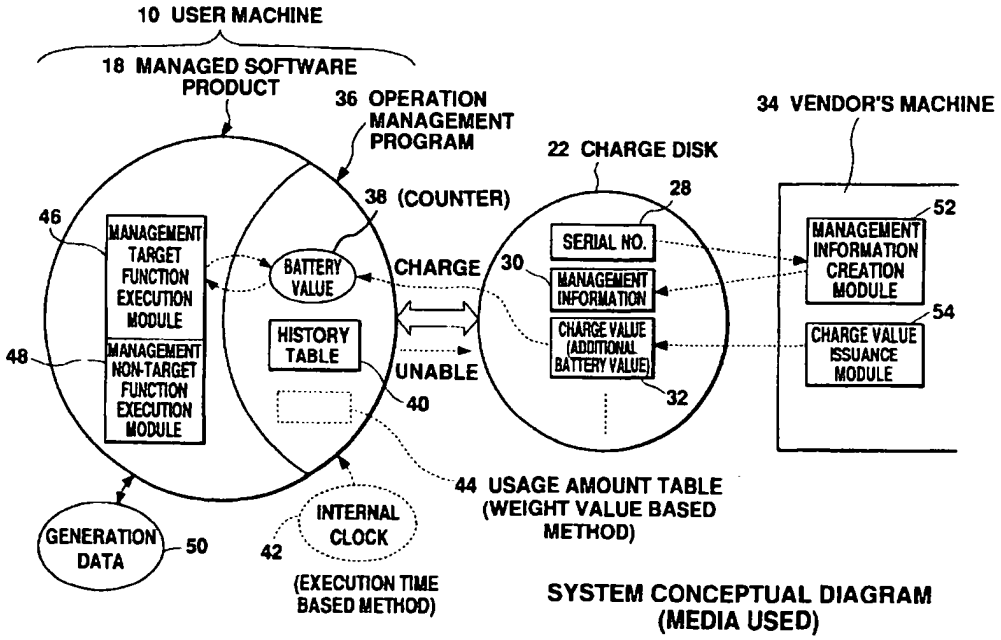


Fig. 3

EP 0 818 748 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to an operation management system and an operation management method, and more particularly to software operation management or execution management.

Description of the Related Art

As computers and computer use become more common, more advanced technology is introduced and a variety of software products are developed for use in various fields. However, in many cases, the user finds it difficult to select a product from among a variety of software products that seem to meet the user's requirements; often, the user cannot find the best tool for his needs.

To reduce such a risk, a service has been available that supplies the user with a trial-use software product free of charge. However, most of these trial-use software products contain only function descriptions or provide the user with limited functions (e.g., save function and/or output function is/are not included). This makes it difficult for the user to evaluate the actual product (all the functions) correctly.

A sales system which charges the user according to how long the user actually uses a software product (including a trial use) would allow him to buy the product anytime he wants, to fully evaluate the product, and to precisely determine the requirements for continued use (including payment for it). Many users would find this type of sales system appealing and economical.

In Japanese Patent Laid-Open Publication No. Sho 59-41061 and Japanese Patent Laid-Open Publication No. Sho 63-153633, a system is disclosed that automatically prevents a program from being used when the usage count reaches a specified value. In Japanese Patent Laid-Open Publication No. Hei 1-147622 a system is disclosed which accumulates program execution time (total program execution time) and prevents the program from being used when the accumulation time reaches a specified amount. However, these systems do not disclose means for extending the program usage period. Japanese Patent Laid-Open Publication No. Hei 5-134949 discloses a system in which a program and expiry of the program are downloaded from a host computer to a user computer via a communication line. Also disclosed is a system in which a new expiry of the program is downloaded from the host computer to the user computer in order to update the expiry. However, the system only measures the execution time taken for executing the entire program, and does not include any means for changing the expiry on the user computer.

In Japanese Patent Laid-Open Publication No. Hei 7-234785, a system is disclosed that relates to a software rental system. This system connects a computer in a rental company to a user computer on which a rental software product is running over a communication line. When the time elapsed from the rental start time reaches the rental limit time, the system makes the program unavailable for use. (For example, the program is deleted.) To allow the user to update the rental period, the rental company sends a rental period extension program to the user's computer over a communication line. The user runs this program to extend the rental period of the program. A drawback of this system is that the user must pay for the software product regardless of whether the user has used it frequently or not. This means that the amount of money the user has to pay depends, not on how often he has used it, but on how long he has used it.

In Japanese Patent Laid-Open Publication No. Hei 7-244585, a system is disclosed that manages the program usage period. This system assigns a usage limit date to a program and, when the current date becomes greater than the limit date, the program product is made unavailable. To extend the usage limit date, the system reads update limit data from a recording medium containing that data and re-assigns a usage limit date based on the update limit data. This system is not reasonable because the amount of money the user has to pay does not depend on whether or not the user actually uses the program.

For example, during execution of a Computer Aided Design (CAD) software product, the user often spends much time thinking without entering data. In the system disclosed by the above mentioned Japanese Patent Laid-Open Publication No. Hei 7-234785 or Japanese Patent Laid-Open Publication No. Hei 7-244585, the user must pay for this thinking time. This places unwanted pressure on the user, especially when he must think carefully during program execution.

SUMMARY OF THE INVENTION

The present invention seeks to solve the problems associated with the art described above. In view of the foregoing, it is an object of the present invention to provide an operation management system and method which reasonably manage the operation of a managed software product.

It is another object of the present invention to provide an operation management system and method which levy a charge according to the actual usage amount of the managed software product (or the amount of the result generated by the managed software product).

It is still another object of the present invention to provide an operation management system and method which manage the operation according to the property of each function of the managed software product.

(1) To achieve the above objects, an operation manage-

ment system for managing the operation of a managed software product according to the present invention comprises: battery value management means for decrementing a battery value according to the operation amount of the managed software product; operation limit means for limiting the operation of the managed software product when the battery value has decreased to a specified limit value; and charge means for adding a charge value to the current battery value when the charge value is entered from external means.

The "battery value" mentioned above is a "virtual battery" which drives a managed software product. This battery value is preferably the value of a counter.

The battery value management means decrement the battery value according to the operation amount of the managed software product. When the battery value has reached a specified limit value (for example, 0), the operation limit means limit all of or a part of the operation of the managed software product. Upon receiving a charge value (additional battery value) from the external means, the charge means add the received value to the current battery value, thus extending the operation period. That is, the battery value is incremented, just as a battery is charged, to allow the continued use of the managed software product.

The managed software product described above is preferably a packaged application software program including a CAD program, game program, video program, language processor, music program, communication program, or a measurement program.

The battery value management means, operation management means, and charge means described above should be implemented preferably as software programs (management software programs) that run on a computer. The managed software product and the management software product may be separate, or the whole or a part of the management software product may be included in the managed software product.

A system according to the present invention is implemented on a general-purpose computer or special-purpose computer having such peripheral units as a disk drive, display, and input unit. The external means described above include recording media such as a magnetic disk or an optical disk and other host computers connected over a network.

(2) An operation management system according to the present invention may be applied to an application software product sales system. The following explains an example:

A vendor sells an application software product containing the operation management program according to the present invention. The operation management program has a battery value defined as the initial value. In addition to this product, the vendor sells recording media containing charge values (e.g., floppy disk (FD)). In this case, it is desirable that a variety of recording media, each containing a unique charge value, be supplied.

On the other hand, a user who bought the application software product may use the product until the battery value reaches zero. This allows the user to fully evaluate and examine the product. A user who wants to use the product after the battery value becomes zero must buy a recording medium containing a charge value to charge the battery. This enables him to add a charge value to the battery value and to use the product continuously.

If the specifications of the application software product do not satisfy the user's request, the user does not buy the recording medium. This prevents additional charges and reduces the cost to the user.

Considering an increase in the sales profit in recording media that will be produced in the future, a combination of a managed software product and the operation management program will lower prices significantly. The operation management system according to the present invention will increase the profits of both the user and the vendor, making it possible to build a very reasonable, economical system.

(3) In a preferred embodiment of the present invention, the battery value management means calculate the operation amount of each function of the managed software product, and subtracts a value corresponding to the operation amount from the battery value.

A continuous decrease in the battery value during execution of a managed software product, as in a conventional system, decrements the value even when the user is idle (input wait time), which places pressure on the user.

Calculating the operation amount of each function during execution of a managed software product, as in a system according to the present invention, decreases the battery value only when the managed software product is actually used, enabling the user to do operation without having to worry about time elapsed while thinking.

(4) In a preferred embodiment of the present invention, function category determination means are also available which determine if an execution instruction from the user activates a management target function or a management non-target function. And, the battery value management means decrement the battery value only when the management target function is executed.

For example, with the data generation function defined as a management target function and with other functions as management non-target functions, a cost can be levied only when new data are generated.

(5) In a preferred embodiment of the present invention, the battery value management means have a weight table containing an operation amount weight value for each of the management target functions. When any of the management target functions is executed, the battery value management means decrement the battery value by the weight value corresponding to the management target function.

In a preferred embodiment of the present invention,

the battery value management means measure the execution time of each of the management target functions and decrement the battery value by the value corresponding to the execution time.

This weight value system is able to calculate the operation amount regardless of the computer speed, which may differ among computers. In addition, by measuring time in this manner, the execution time is directly monitored and therefore the operation mount becomes proportional to the CPU load.

(6) In a preferred embodiment of the present invention, the operation limit means prevent only the management target functions from being executed when the battery value has decreased to a specified limit value; management non-target functions are executed.

For example, forcing a game program used at home to terminate when the battery value has reached a specified value does not cause a serious problem.

However, for a CAD program used in an office, forced termination when the battery value has reached a specified value may make already-produced data unavailable, possibly interrupting a job. Therefore, considering user's advantage and convenience, the embodiment keeps some functions operable even when the battery value has reached a specified value.

(7) A preferred embodiment of the present invention has remainder warning means for issuing a remainder warning message when the battery value has decremented to a specified warning value because a sudden inoperable condition in the managed software product without prior notice may cause the user unexpected damage. The remainder warning means alert the user to that condition before it occurs. In other words, the warning message prompts the user to determine whether to charge the battery value.

A preferred embodiment of the present invention has remainder display means for displaying the battery value on the screen during execution of the managed software product. This remainder display information keeps the user informed of the amount by which the managed software product will be able to continue operation without being charged.

It is also possible to program the system so that, upon detecting that the battery value has been charged to a specified value, the system can automatically disable operation management through the battery value to allow the user to use the product indefinitely.

(8) To achieve the above objects, a method for managing the operation of a managed software product according to the present invention comprises: a count value management step for changing a count value according to the operation amount of the managed software product; an operation limit step for limiting the operation of the managed software product when the count value has reached a specified limit value; and a charge step for charging the current count value or the limit value when a charge value is entered from external means.

The above count value is incremented or decremented according to the operation amount of the managed software product. When the count value is incremented, a charge value is added to the limit value; when the count value is decremented, a charge value is added to the current count value. In either case, the usage period is extended by charging the battery value.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram showing a user machine used in the operation management system according to the present invention.

Fig. 2 is a diagram showing the data structure of a charge disk.

Fig. 3 is a diagram showing the concept of the operation management system according to the present invention.

Fig. 4 is a diagram showing an example of the history table.

Fig. 5 is a diagram showing an example of the usage amount table.

Fig. 6 is a flowchart showing the processing of the system when a management target function is executed in the execution time based method.

Fig. 7 is a flowchart showing the processing of the system when a management target function is executed in the weight value based method.

Fig. 8 is a flowchart showing the charge disk read processing.

Fig. 9 is a flowchart showing the charge processing.

Fig. 10 is a diagram showing a user machine used in another embodiment.

Fig. 11 is a diagram showing the structure of data sent from the host machine to a user machine.

Fig. 12 is a diagram showing the concept of the system in another embodiment.

Fig. 13 is a diagram showing an example of the user registration table.

Fig. 14 is a flowchart showing the operation of the user machine and a user machine in another embodiment.

Fig. 15 is a diagram showing another configuration of the system.

Fig. 16 is a diagram showing an example of an application according to the present invention.

Fig. 17 is a flowchart showing the function category determination processing.

DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 shows a user machine 10. This user machine 10 is a computer which executes various types of application programs under control of the operation system (OS). The user machine 10 is composed of a system unit 12, display 14, keyboard (not shown in the figure), output unit (not shown in the figure) such as a printer or plotter, and so forth. The system unit 12 contains a CD-ROM disk drive 16 which accesses a CD-ROM and

reads data from it and a floppy disk drive 20 which accesses a floppy disk (FD) and reads data from it.

The CD-ROM shown in Fig. 1 contains a managed software product 18. In this embodiment, the managed software product 18, such as a CAD software product, has an operation management program built in. The operation management program, designed for managing the operation of the managed software product 18, manages the operation using a "battery value" which will be described below. In the example shown in Fig. 1, the managed software product 18 is installed from the CD-ROM to the user machine 10; it may be installed from any other recording medium or via a communication line.

A charge disk 22, containing specified data (including a charge value) on a floppy disk, functions as a battery value charger. Inserting this charge disk 22 into the floppy disk drive 20 causes a charge value to be read and enables the user to extend the allowable operation period of the managed software product 18. In this embodiment, several charge disks 22, each containing a unique charge value, are supplied to allow the user to select or buy a desired charge disk 22 to add a desired charge value to the battery value.

The managed software product 18 and the charge disk 22 are usually supplied from the same vendor. In this embodiment, the managed software product 18 includes the operation management program. Of course, the managed software product 18 and the operation management program may be separately loaded into the user machine 10.

In Fig. 1, the display 14 has a remainder information area 24 where remainder information is displayed and a remainder warning area 26 where a warning message is displayed when the remainder drops below the specified amount. These areas will be described later.

Fig. 2 shows the data structure of the charge disk 22. As shown in Fig. 2, the charge disk 22 contains a serial number 28, management information 30, and charge value (additional battery value) 32. The serial number 28 is a unique identification number that is assigned when the floppy disk is formatted. Usually, this number is not copied when the disk is copied. The management information 30 is created when the serial number 28 is encrypted. This management information 30 is copied when the disk is copied. Therefore, when the disk is copied illegally, the serial number 28 and the management information 30 do not match, thereby making it easy to determine that the disk is copied illegally. Of course, any other conventional security system may also be used instead of this method.

The charge value 32 is an additional charge value to be added to the battery value that is decremented as the user uses the managed software product 18. Charging the battery value with this charge value enables the user to extend the usage period.

When the battery value is managed in the "execution time based method" in which the battery value is decremented by the execution time of each function, an additional time is recorded as the charge value 32. On the other hand, when the battery value is managed in the "weight value based method" in which the battery value is decremented by the weight value of each function, the additional value is recorded as the charge value 32. These methods will be described in more detail later.

Although a floppy disk is used as the charge disk 22 in the embodiment shown in Fig. 1, other types of recording media may also be used. Also, as shown in another embodiment that will be explained later, a charge value may be sent over a communication line.

Fig. 3 shows the concept of the operation management system which uses the charge disk 22. The system is composed primarily of the user machine 10, charge disk 22, and vendor's machine 34. In this embodiment, the managed software product 18 including the operation management program 36 is installed in the user machine 10.

The charge disk 22 is generated on the vendor's machine 34 owned by the vendor which sold the managed software product 18. More specifically, the vendor's machine 34 has two software modules: the management information creation module 52 and the charge value issuance module 54. The management information creation module 52 encrypts the serial number 28 recorded on the charge disk 22, and writes the resulting management information 30 back onto the charge disk 22. Note that the operation management program 36, which contains the encryption condition or the decryption condition, can check whether or not the serial number 28 agrees with the management information 30. The charge value issuance module 54 records the charge value 32, which has been set by the vendor, onto the charge disk 22. In the execution time based method, the charge value 32 is recorded, for example, as 100 hours, 200 hours, or 500 hours. Note that the operation management program 36 contains an initial battery value (for example, 100 hours).

The operation management program 36 has a counter 38 which decrements the battery value (battery value management function). In this embodiment, the operation management program 36 decrements the counter 38 each time a "management target function" provided by the managed software product 18 is executed. When the battery value, i.e., the counter value, has decremented to the limit value of 0, the operation management program 36 prevents management target functions from being executed. That is, in this embodiment, when the battery value has reached a specified limit value, the execution of the managed software product 18 is limited and, when the battery value is charged with the charge value 32 contained on the charge disk 22, the charge value is added to the battery value and the resulting value is used as a new battery value. The usage period of the managed software product 18 is thus extended.

A history table 40 managed by the operation man-

agement program 36 contains history information on charge values recorded on the charge disk 22. Fig. 4 shows an example. As shown in Fig. 4, the history table 40 is composed of three columns: FD serial number column 40A, charge data/time column 40B, and charge value column 40C. The table may have other columns as necessary.

Referring to Fig. 3 again, the following explains how the battery value is managed. When the battery value is managed in the "execution time based method" described above, the execution time of each management target function, measured based on the internal clock 42, is subtracted from the battery value. On the other hand, when the "weight value based method" described above is used, the battery value is managed based on the usage amount table 44. Fig. 5 shows an example of the usage amount table 44. In this embodiment, the table contains entries, each consisting of a function name 44A and the corresponding usage amount 44B. It should be noted that each usage amount is used as a weight value. For example, a weight value is pre-defined according to the processing time of each function. Therefore, when a management target function is executed, the corresponding usage amount (weight value) is subtracted from the battery value.

The managed software product 18 shown in Fig. 3 has many user interface programs as well as many internal functions and common functions used by the programs. These functions are classified roughly into two: management target functions and management non-target functions. Whenever the managed software product 18 attempts to execute a management target function, the operation management program 36 references the battery value and, when it is zero or greater, allows the managed software product 18 to execute that function. When the managed software product 18 attempts to execute a management non-target function, the operation management program 36 does not check the battery value. For example, when input/output function for processing generated data 50 from the managed software product 18 is defined as a management non-target function, the input/output processing is always executed on the generated data 50, even if the usage period of the managed software product 18 has expired. This ensures that the generated data 50 are always processed, thus protecting user assets. Examples of management non-target functions include the data display function, data print function, and data plotter output function.

Management target functions include the data generation function. For example, when the managed software product is a CAD software product, the data generation function includes the straight-line drawing function, curved-line drawing function, circle drawing function, area fill-in function, area hatching function, and character insertion function.

Fig. 3 conceptually shows management target function execution module 46 which executes management target functions and management non-target function execution module 48 which executes management non-target functions. In this embodiment, the battery value is decremented only when a management target function is activated. Note that the battery may be decremented when both a management target function and a management non-target function are activated.

In addition to the data described above, the charge disk 22 may contain other types of data. For example, it may contain the name of the managed software product 18 which accepts a charge value. In this case, the name of the managed software product 18 is used as follows. When the charge disk 22 is read, the operation management program 36 checks whether or not the name of the managed software recorded on the charge disk 22 matches that of the managed software product 18 installed in the user machine 10 and, only when they match, accepts the charge value 32.

The battery value described above is stored on the hard disk and then copied into the computer's RAM. The battery value in the RAM is decremented whenever a management target function is executed. Also, at an interval or as necessary, the battery value in the RAM replaces the battery value on the hard disk. This means that, even when the computer fails, the battery value is not erased. The battery value may also be maintained in some other way.

Fig. 17 is a flowchart showing how the operation management program operates when it accepts an instruction requesting the execution of a managed software product function. The following explains this processing in more detail.

Upon receiving from a user an instruction requesting the execution of a function of the managed software product while the managed software product is in execution (S601), the operation management program checks whether the requested function is a management target function or a management non-target function (S602). When the function is a management target function (S603), the operation management program performs the processing shown in Fig. 6 or Fig. 7 (S604). When the function is a management non-target function (S603), the program executes the function immediately. (S605). This processing is repeated whenever an execution instruction is received.

Next, referring to Fig. 3, the execution of a management target function in the execution time based method is explained with the use of Fig. 6.

When the user requests the execution of a management target function while the managed software product 18 shown in Fig. 3 is in execution, the routine shown in Fig. 6 is started. First, the management target function execution module 46 or the operation management program 36 reads the battery value to check if it is greater than zero. If the battery value is zero or less, the routine is terminated. That is, the requested management target function cannot be started. Note that a management non-target function is started even if the battery value is

zero.

In S102, the routine gets the start time from the internal clock 42 before starting the requested management target function and, in S103, starts the management target function. In S104, the routine gets the end time from the internal clock 42 and, in S105, subtracts the start time from the end time to calculate the processing time (execution time) of the processing executed in S103.

In S106, the routine subtracts the processing time calculated in S105 from the battery value. In S107, the routine checks if the resulting battery value is equal to or less than the warning value and, if so, displays a message in the remainder warning area 26 shown in Fig. 1. If the resulting battery value is greater than the warning value, the routine does not display the message. As shown in Fig. 1, the remainder information area 24 is displayed during execution of the managed software product 18 (see Fig. 1) to allow the user to check the remaining amount. This helps the user determine how long he can execute the managed software product 18.

Fig. 7 shows the processing of a management target function in the weight value based method.

When the execution of a management target function is requested as described above, the routine references the battery value in S201 to check if it is equal to or greater than 0. If it is, the routine executes the requested management target function in S202 and, in S203, references the usage amount table 44 shown in Fig. 5 to find the usage amount (weight value) of the executed management target function. Then, in S204, the routine subtracts the processing amount found in S203 from the battery value to find a new battery value. In S205, the routine checks if the battery value is less than the warning value and, if so, displays a message in the remainder warning area 26 in S206.

The "execution time based method" shown in Fig. 6 allows the user to manage operation using a physical amount that is easy to understand. In addition, the user can manage operation in a relatively simple configuration. On the other hand, the "weight value based method" shown in Fig. 7 gives the user the same result regardless of the CPU speed of the user's machine.

Next, referring to Fig. 3, the charge disk 22 read processing is explained with the use of Fig. 8.

This processing is started when the charge disk 22 is inserted into the floppy disk drive 20 as shown in Fig. 1. The routine reads the serial number in S301, and the management information in S302, both from the charge disk 22. In S303, the routine encrypts the serial number according to the encryption condition, or decrypts the management information according to the decryption condition, and compares the serial number with the management information. This comparison determines whether or not the charge disk 22 is legal. For example, when the disk is illegally copied, the management information 30 is copied, but the serial number 28 is not copied but replaced. This results in a mismatch between the serial number 28 and the management information 30, thereby making it possible to find an illegal copy.

In S304, the routine checks if the charge disk 22 is valid and, if it is not valid, terminates processing in S308. If it is valid, the routine references the history table 40, containing past charge history data, in S305 to check the validity of the charge value 32 recorded on the charge disk 22. To do so, the routine first checks to see if the serial number 28 of the charge disk 22 is in the history table 40. If the serial number is found, the routine takes the following steps to check if the charge value 32 recorded on the charge disk 22 is valid. The routine finds the charge value initially recorded on the charge disk 22 and, from that initial value, subtracts the actual charge value to find the remainder. The next time the battery value is charged, the routine compares the remainder with the charge value currently recorded on the charge disk. If the charge value on the charge disk 22 is greater than the remainder, the routine determines in S306 that the charge disk is not valid and terminates processing in S308. If the routine finds that the charge value 32 on the charge disk 22 is valid, it performs the charge processing, shown in Fig. 9, in S307.

Fig. 9 shows an example of charge processing. In S401, the routine references the counter 38 to read the current battery value and, in S402, reads the charge value from the charge disk 22. In S403, the routine asks the user to type an actual charge value that does not exceed the charge value 32 recorded on the charge disk 22. The user types the charge value, for example, from the keyboard. In S404, the routine checks that the specified charge value is less than the charge value on the charge disk 22. If the specified charge value is greater than the charge value on the charge disk 22, the routine asks the user to retype the charge value.

In S405, the routine adds the specified charge value to the battery value, thus charging the battery value. In S406, the routine subtracts the specified charge value from the initial charge value and writes the resulting value on the charge disk 22 as a new charge value 32 . If the initial charge value 32 is exhausted, the routine writes the value of 0 on the charge disk 22 to virtually erase the charge value. The value of 0 prevents the charge disk 22 from being re-used. In S407, a record relating to the charge processing is added to the history table 40.

In the above embodiment, the user specifies an actual charge value. Instead of having the user specify a value, a pre-defined charge value may be added to the battery value at that time.

Fig. 10 shows another embodiment according to the present invention. In the embodiment described above, the battery value is charged using a recording medium. In this embodiment, the battery value is charged via a communication line 60. For the same components as those used in the above embodiment, the same numbers are assigned and their descriptions are omitted.

The user machine 10 in Fig.10 is connected to the

host machine 62 via the communication line 60. From this host machine 62, send data 64 shown in Fig. 11 are sent to the user machine 10 to charge the battery value.

In Fig. 11, address information 68 specifies the address of the user machine 10. Management information 70 is created by encrypting the serial number on the recording medium containing the managed software product 18. A charge value 72, a value to be added to the battery value as with the above embodiment, is an additional period of time in the execution time based method, and is an additional amount in the weight value based method.

Fig. 12 illustrates the system concept of this embodiment.

As described above, the user machine 10 is connected to the host machine 62 via the communication line 60. That is, this host machine 62 is connected to each of a number of user machines 10 for integrated operation management. This host machine 62 has a management information creation module 76, charge value issuance module 78, user registration table 80, and billing module 82. The management information creation module 76 creates the management information 70 shown in Fig. 11, and the charge value issuance module 78 issues a charge value 72 in response to a request from the user machine 10. As shown in Fig. 13, the user registration table 80 is composed primarily of the user ID column 80A, user name column 80B, and request charge value column 80C. The billing module 82 references the user registration table 80 to automatically issue a bill for a requested amount whenever a charge value is issued, or at some specified interval.

Next, referring to Fig. 12, the operation of this embodiment is explained with the use of Fig. 14. The operation of the user machine 10 is shown in the left side of Fig. 14, while that of the host machine 62 is shown on the right.

First, in S501 and S502, the user machine 10 is connected to the host machine 62 via a communication line. In S503, the user machine 10 generates a request for a charge value that will be sent to the host machine 62. In this case, the request contains at least the serial number of the CD-ROM containing the managed software product 18 and information on the charge value. In S504, the user machine sends the request to the host machine and, in S505, the host machine receives the request.

In S506, the host machine checks the user registration table 80. If the host machine finds, in S507, that the requesting user is registered in the host machine 62, the management information creation module 76 creates management information based on the serial number in S508, and the charge value issuance module 78 generates a charge value in response to the request from the user. In S509, the host machine 62 sends the management information and the charge value to the user machine 10 as the send data 64 shown in Fig. 11. In S510, the user machine 10 receives the send data 64. In S511 and S512, the user machine 10 and the host machine 62 are disconnected.

In S513, the operation management program 36 compares the serial number 74 with the management information 70 to check to see if the data received by the user machine 10 are valid. This prevents the user from illegally charging the battery value. If it is found in S514 that the send data are valid, the charge processing is performed in S515. This charge processing is the same as that in Fig. 9.

As shown in Fig. 12, this embodiment may also use the execution time based method or the weight value based method in order to manage the battery value.

Although the battery value is charged over a communication line such as a telephone line in the above embodiment, it may also be charged over a communication satellite (satellite line).

In the above embodiments, the operation management program 36 is included in the managed software product 18. Of course, an external program can manage the operation of the managed software product 18. Fig. 15 shows the concept of such an embodiment.

As shown in Fig. 15, the operation system (OS) 83 is located between the hardware 81 and each of application programs 84, 86, and 88. The operation management program 36 according to the present invention may be located between the operation system 83 and the application program 84.

Operation management program 36 therefore functions as an interface program. Messages are exchanged between the operation management program 36 and the application program 84 according to some specific rule. Messages are also exchanged between the operation management program 36 and the operation system 83 according to a specific rule.

To execute a management target function in this configuration, the operation management program 36 references the battery value when it receives an execution request from the application program 84. If the battery value is not zero, the operation management program 36 sends an instruction to the operation system 83 while simultaneously decrementing the battery value by a value corresponding to the function. If the battery value is zero, the operation management program 36 sends a message back to the application program 84, indicating that the instruction cannot be executed.

To execute a management non-target function, the operation management program 36 does not reference the battery value when it receives an execution request from the application program 84 but instead sends the instruction directly to the operation system 83.

The battery value is decremented as management target functions are executed. Charging the battery value allows the user to extend the usage period of the application program 84, which may be supplied separately from the application program 84.

In the above embodiments, one operation management program manages one operation management program. It is also possible for one operation manage-

ment program to manage several application programs.

Fig. 16 shows an application of the present invention. The system shown in Fig. 16 is composed of one host machine 90 and several user machines 92. Within each user machine 92 are a managed software product 18 and the operation management program 36, which, in turn, contains the counter 38 where the battery value to be decremented is stored. In other words, the operation of the managed software product 18 is controlled by the value stored in the counter 38. To execute the managed software product 18 in this system, it is necessary to insert a battery disk 96 into the user machine 92 and to move the battery value from the battery disk 96 into the counter 38. The battery value is decremented as the operation of the managed software product 18 proceeds. When the user finishes the managed software product 18, a sequence of operations are executed to move the current counter value from the counter 38 to the battery disk 96. This initializes the counter 38 to zero just as it was before the battery disk 96 was inserted.

The host machine 90 has several disk drives into which a battery disk 96 is inserted to read the battery value that was returned to the battery disk 96. This host machine 90 is also used to charge the battery value on the battery disk 96.

Integrated management of the battery values on several battery disks 96 through the host machine 90 brings a benefit of integrally managing several managed software products 18.

This type of system may be used, for example, in a school or a business where many computers are installed. With an individual carrying his or her own portable battery disk 96, it is possible to check and control the software usage amount of each person. In this case, either the "execution time based method" or the "weight value based method" may be used.

Claims

1. An operation management system for managing the operation of a managed software product, comprising:

   battery value management means for decrementing a battery value according to the operation amount of said managed software product;
   operation limit means for limiting the operation of said managed software product when said battery value has decremented to a specified limit value; and
   charge means for adding a charge value to the current battery value when the charge value is entered from external means.

2. An operation management system according to claim 1, wherein said battery value management means find the operation amount for each execution of a function owned by said managed software product and subtract a value corresponding to said operation amount from said battery value.

3. An operation management system according to claim 2, further comprising:

   function category determination means for determining if a function to which an execution instruction is issued is a management target function or a management non-target function, wherein said battery value management means decrement said battery value only when said management target function is executed.

4. An operation management system according to claim 3, wherein

   said battery value management means has a weight table containing pairs of said management target function and a weight value representing said operation amount thereof, and
   said battery value management means subtract a weight value corresponding to said management target function from said battery value when said management target function is executed.

5. An operation management system according to claim 3, wherein, when said management target function is executed, said battery value management means measure the execution time and subtracts the execution time from said battery value.

6. An operation management system according to claim 3, wherein said operation limit means prevent said management target function from being executed but allows said management non-target function to be executed when said battery value has reached a limit value.

7. An operation management system according to claim 3, wherein said managed software product has a data generation function and a data output function and wherein said function category determination means determine said data generation function as said management target function and determine said data output function as said management non-target function.

8. An operation management system according to claim 1, further comprising remainder warning means for issuing a remainder warning when said battery value has decremented to a warning value.

9. An operation management system according to claim 1, further comprising remainder display

means for displaying said battery value during execution of said managed software product.

10. An operation management system for managing the operation of a managed software product, comprising:

battery value management means for decrementing a battery value according to the operation amount of said managed software product;
operation limit means for limiting the operation of said managed software product when said battery value has decremented to a specified limit value;
read means for reading a charge value from a recording medium containing the charge value thereon; and
charge means for adding said charge value to the current battery value.

11. An operation management system according to claim 10, further comprising erase means for erasing the charge value from said recording medium after said charge value is added.

12. An operation management system according to claim 10, further comprising:

specification means for allowing a user to specify an actual charge value by which the current battery value is to be actually charged, the actual charge value not exceeding the charge value recorded on said recording medium; and
rewrite means for rewriting the charge value on said recording medium with a remainder value after said actual charge value is added to the current battery value.

13. An operation management system according to claim 10, in which said recording medium contains not only said charge value, but also the identification number of the recording medium and management information generated through encryption of the identification number, said operation management system further comprising:

validity determination means for comparing said identification number with said management information considering the condition of said encryption to determine the validity of said recording medium.

14. An operation management system comprising:

a managed machine containing a managed software product; and
a managing machine connected to said managed machine with a communication line, wherein
said managed machine comprises:
battery value management means for decrementing a battery value according to the operation amount of said managed software product;
operation limit means for limiting the operation of said managed software product when said battery value has decremented to a specified limit value;
charge value receive means for receiving a charge value from said managing machine; and
charge means for adding said charge value to the current battery value, and wherein
said managing machine comprises:
charge value send means for sending said charge value to said managed machine.

15. An operation management system according to claim 14, wherein said managed machine further comprises:

notification means for notifying said managing machine of the identification number of a portable recording medium initially containing said managed software product; and
validity determination means for comparing management information sent from said managing machine with said identification number to determine the validity of the recording medium; and wherein said managing machine further comprises:
management information creation means for creating said management information generated by encrypting said notified identification number and for sending the management information to said managed machine.

16. An operation management system comprising:

at least one managed machine containing a managed software product; and
a managing machine for managing the operation of said managed machine, wherein said managed machine comprises:
a counter containing a battery value changing according to the operation amount of said managed software product;
first charge means for reading a battery value from a portable recording medium to store the battery value into said counter; and
first return means for writing the current battery value on said recording medium, and wherein, said managing machine comprises:
second charge means for writing said battery value on said recording medium; and
second return means for reading said battery value from said recording medium.

17. An operation management method comprising:

a count value management step for changing a count value according to the operation amount of a managed software product;
an operation limit step for limiting the operation of said managed software product when said count value has reached a specified limit value; and
a charge step for charging the current count value or said limit value when a charge value is entered from external means.

18. A medium containing a management software product for managing the operation of a managed software product, wherein said managed software product and said management software product are executed on computers, said management software product comprising:

a module for changing a count value according to the operation amount of said managed software product;
a module for limiting the operation of said managed software product when said count value has reached a specified limit value; and
a module for charging the current count value or said limit value when a charge value is entered from external means.

19. A medium containing a charge value read by a management software product for use in managing the operation of a managed software product, wherein said managed software product and said management software product are executed on computers, said management software product comprising:

a module for changing a count value according to the operation amount of said managed software product;
a module for limiting the operation of said managed software product when said count value has reached a specified limit value; and
a module for charging the current count value or said limit value when said charge value is entered.

20. A computer system having an interface software product between an operation system and at least one application software product, wherein said interface software product comprises:

a module for changing a count value according to the operation amount of said application software product;
a module for limiting the operation of said application software product when said count value has reached a specified limit value; and
a module for charging the current count value or said limit value when a charge value is entered from external means.

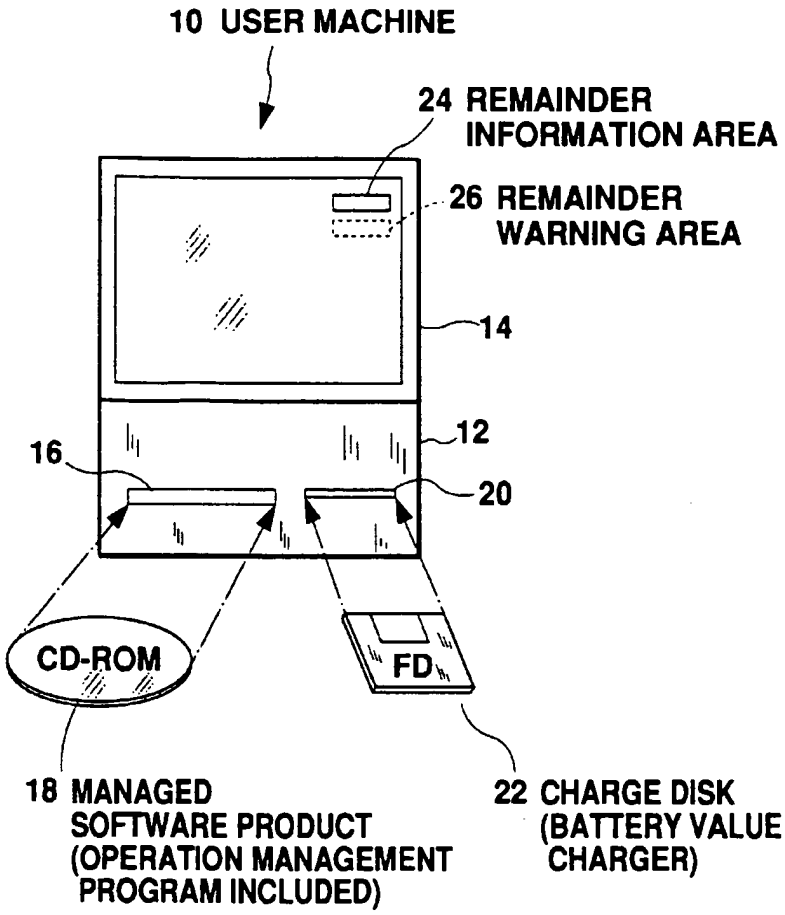


Fig. 1

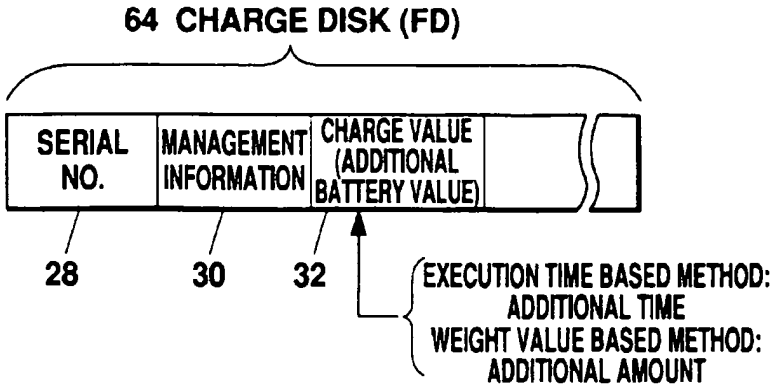


Fig. 2

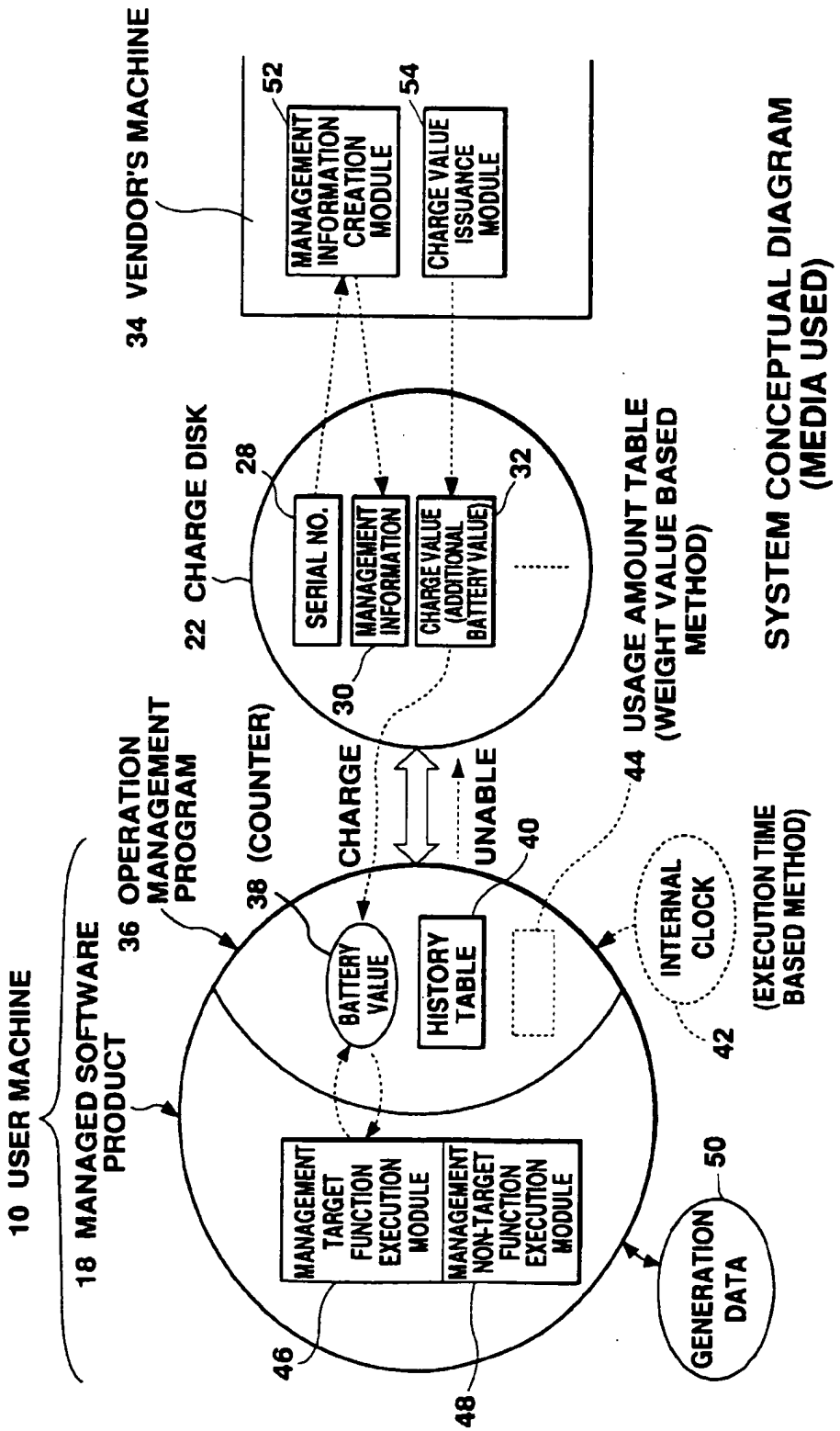


Fig. 3

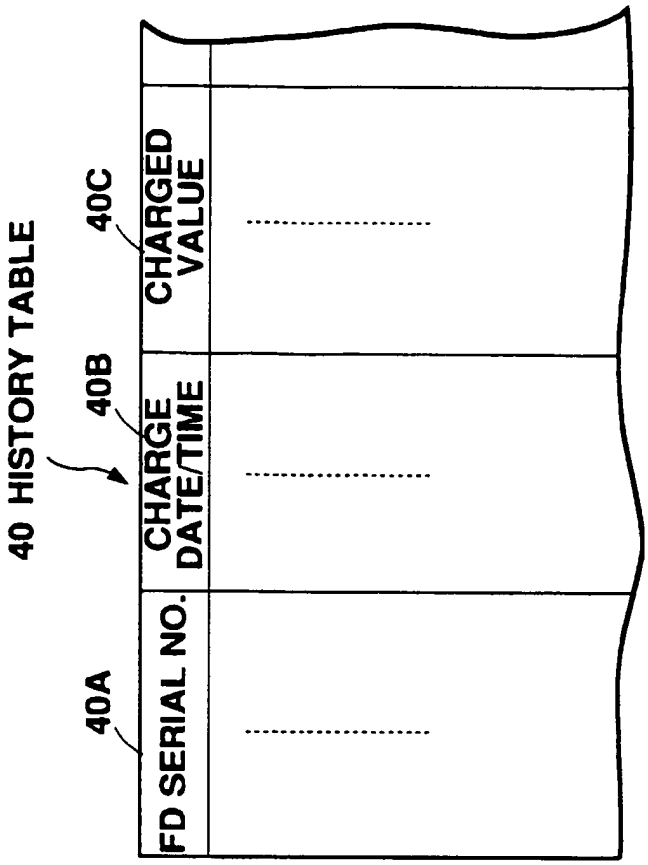


Fig. 5

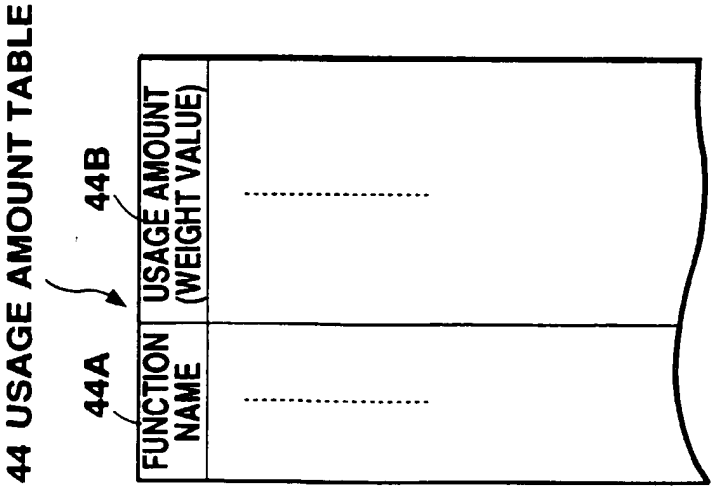


Fig. 4

MANAGEMENT TARGET FUNCTION EXECUTION
(EXECUTION TIME BASED METHOD)

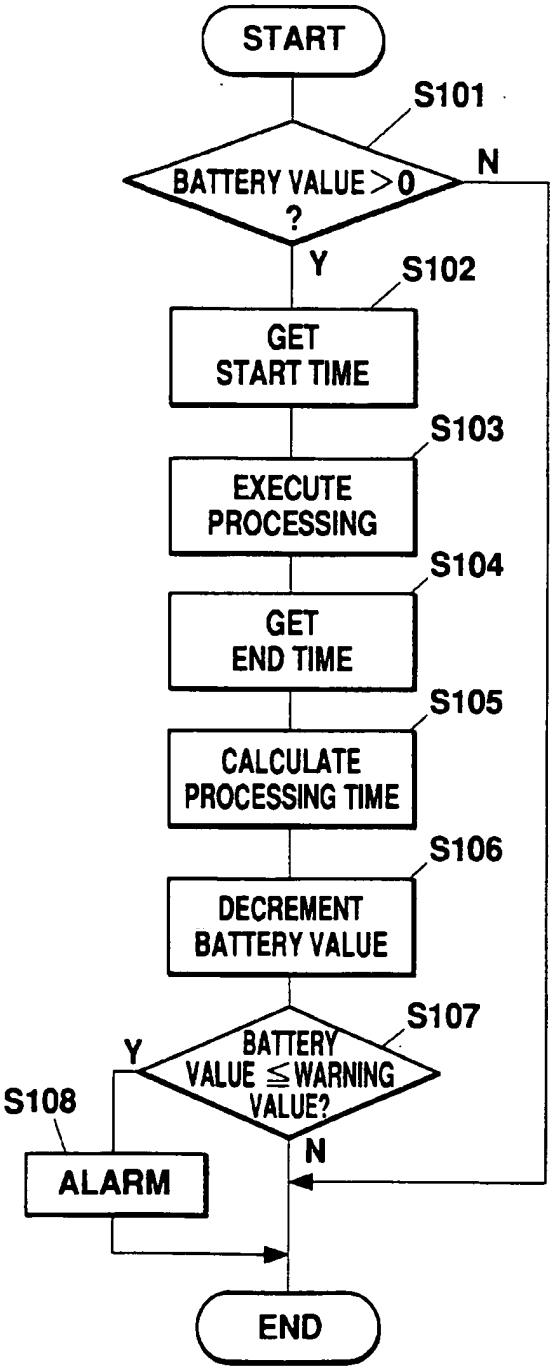


Fig. 6

MANAGEMENT TARGET FUNCTION EXECUTION
(WEIGHT VALUE BASED METHOD)

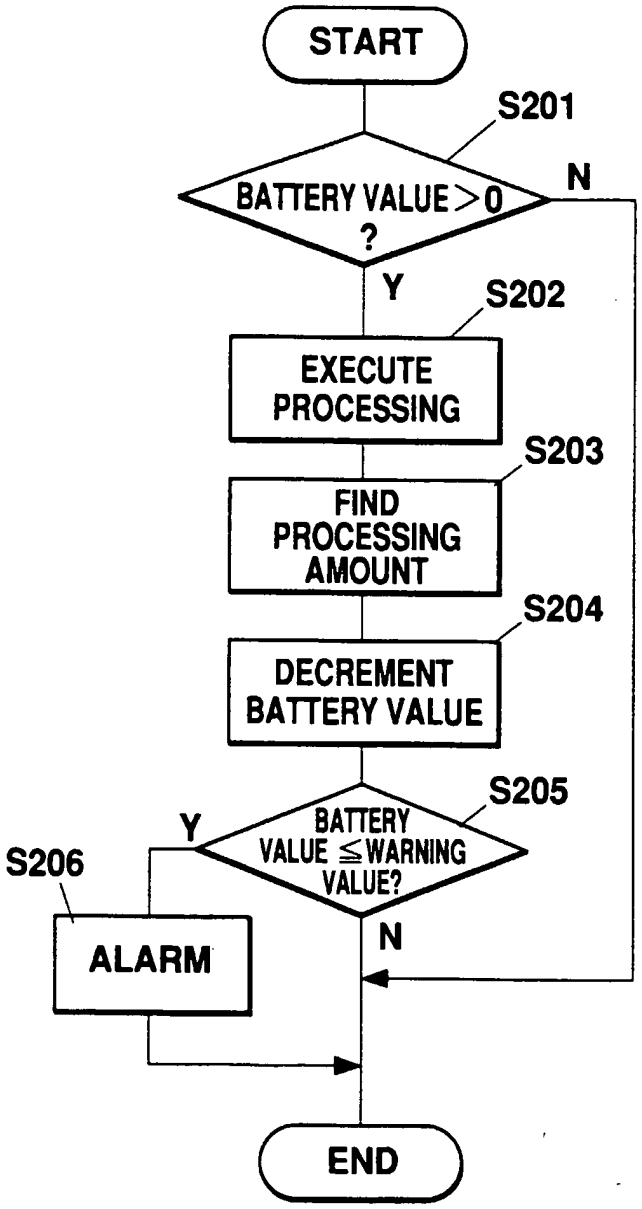


Fig. 7

CHARGE DISK READ PROCESSING

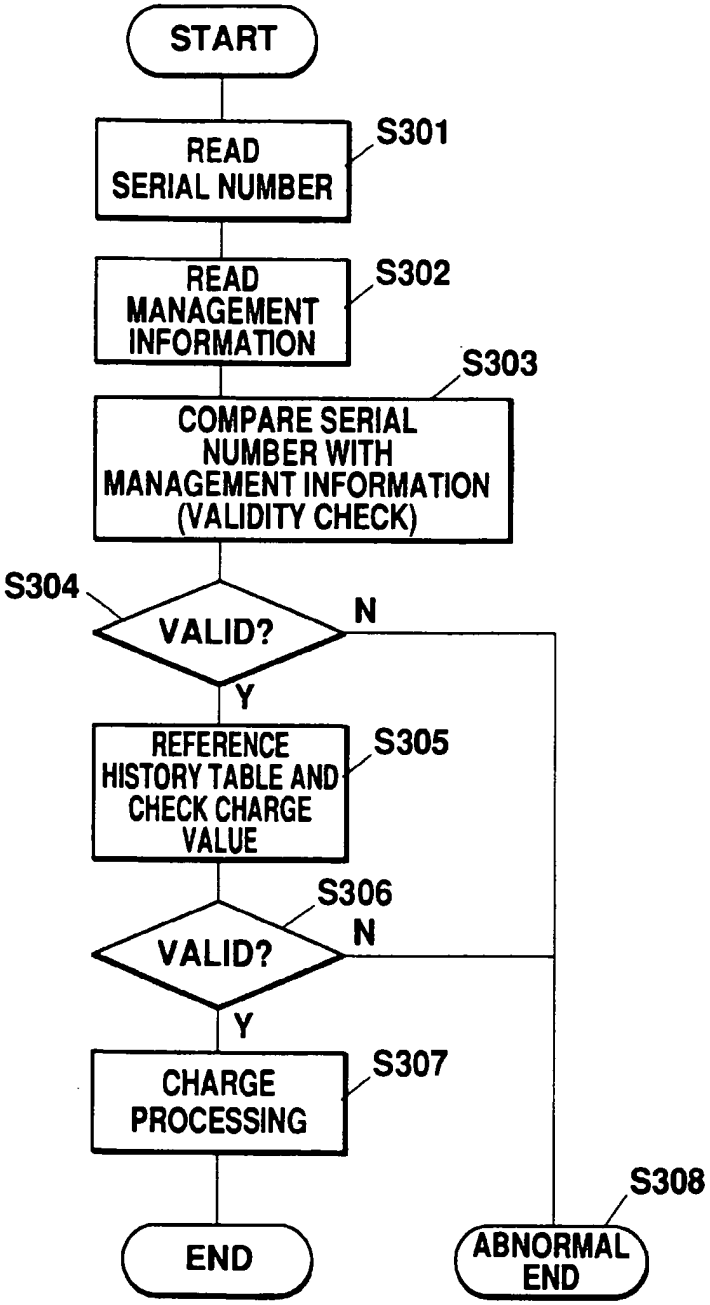


Fig. 8

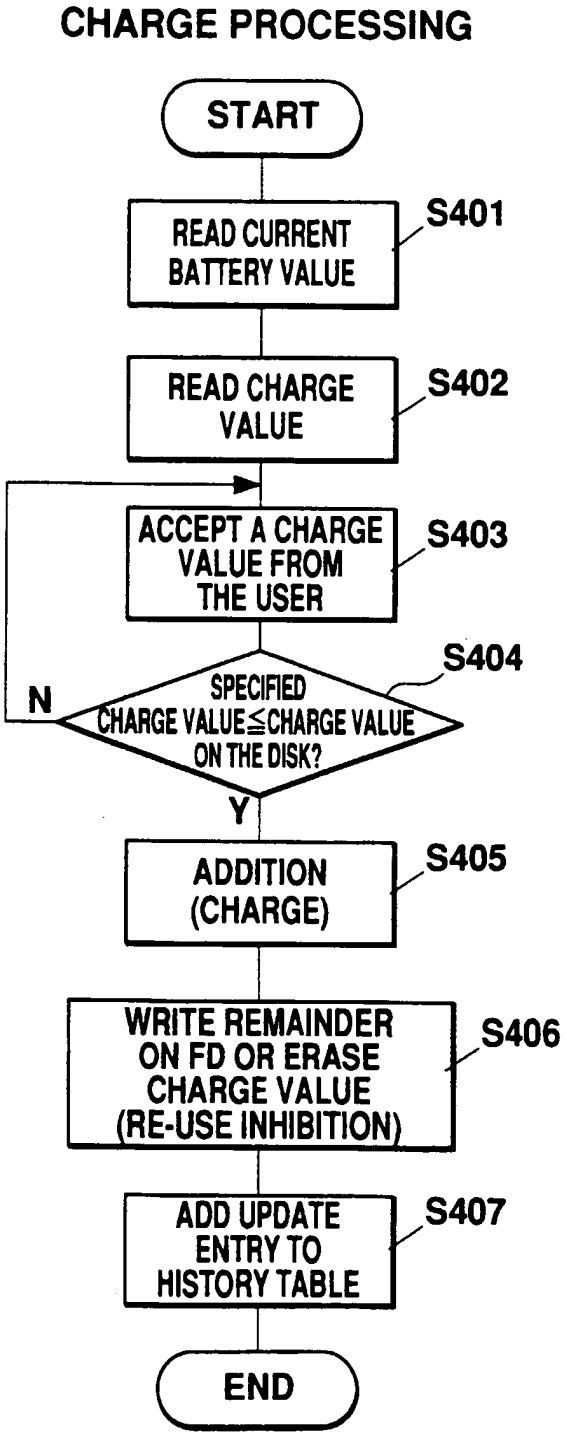


Fig. 9

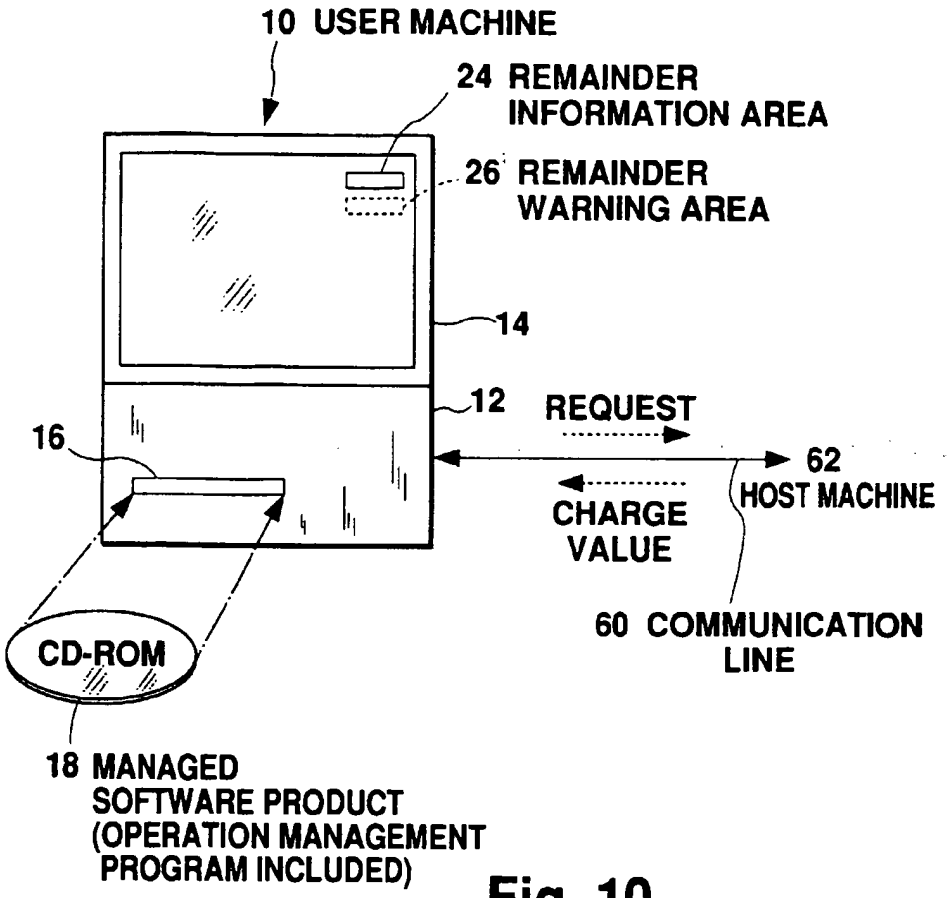


Fig. 10

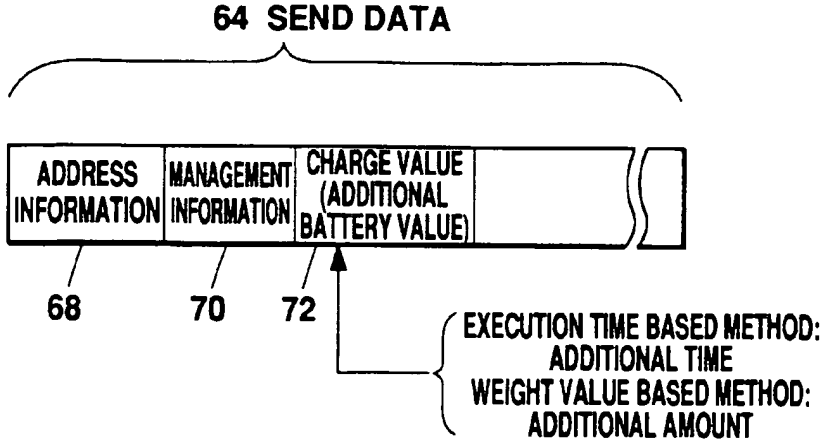
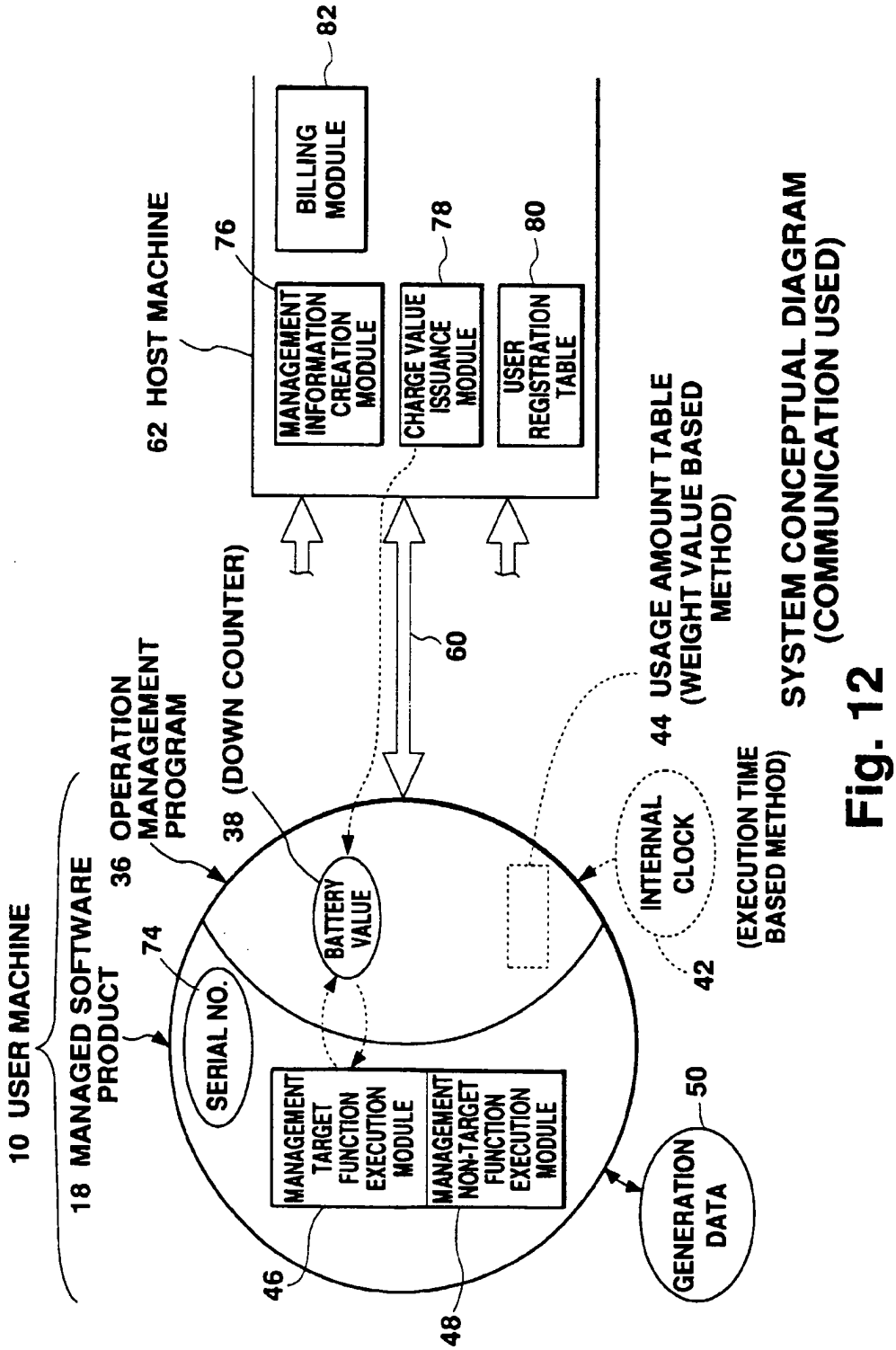


Fig. 11



SYSTEM CONCEPTUAL DIAGRAM (COMMUNICATION USED)

Fig. 12

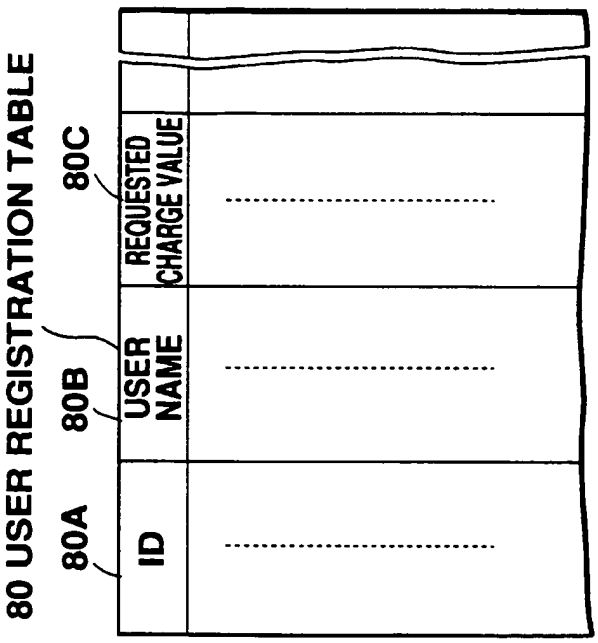


Fig. 13

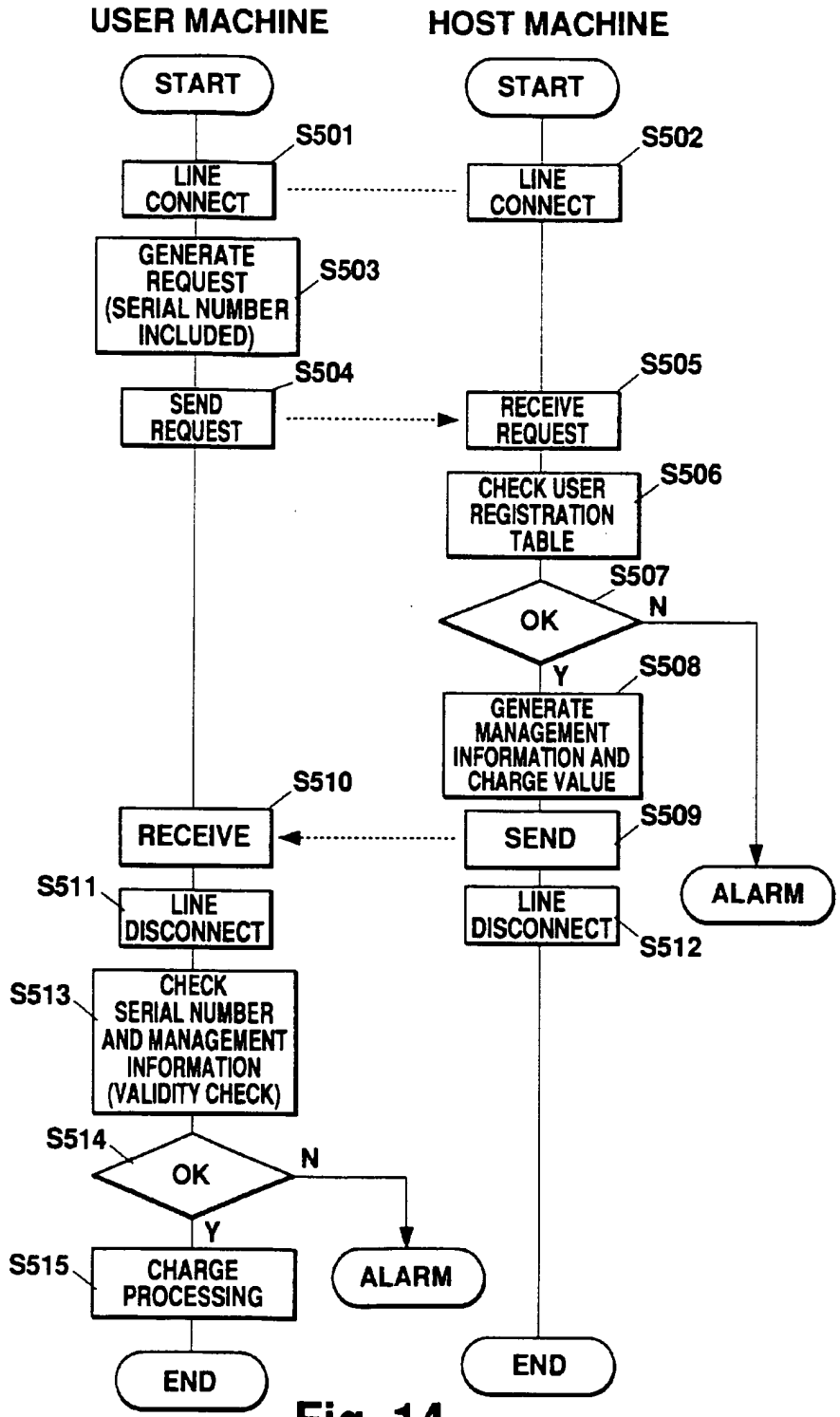


Fig. 14

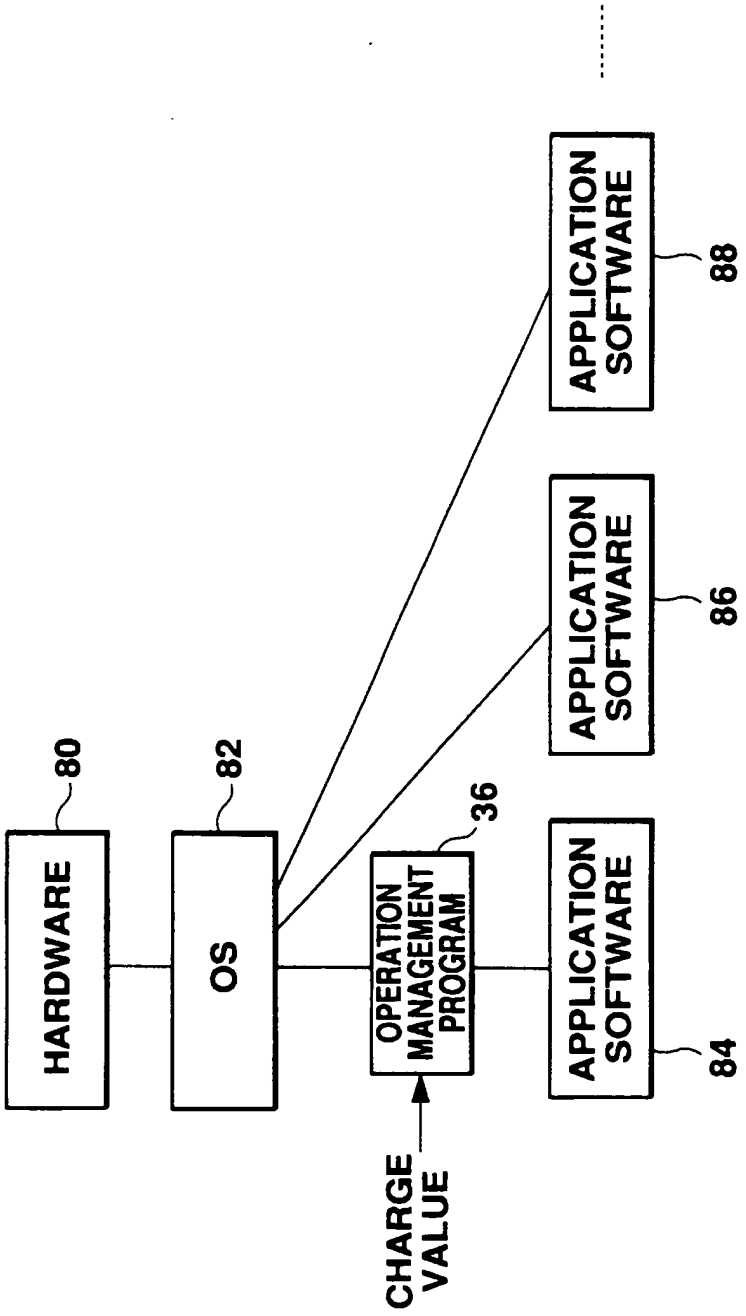


Fig. 15

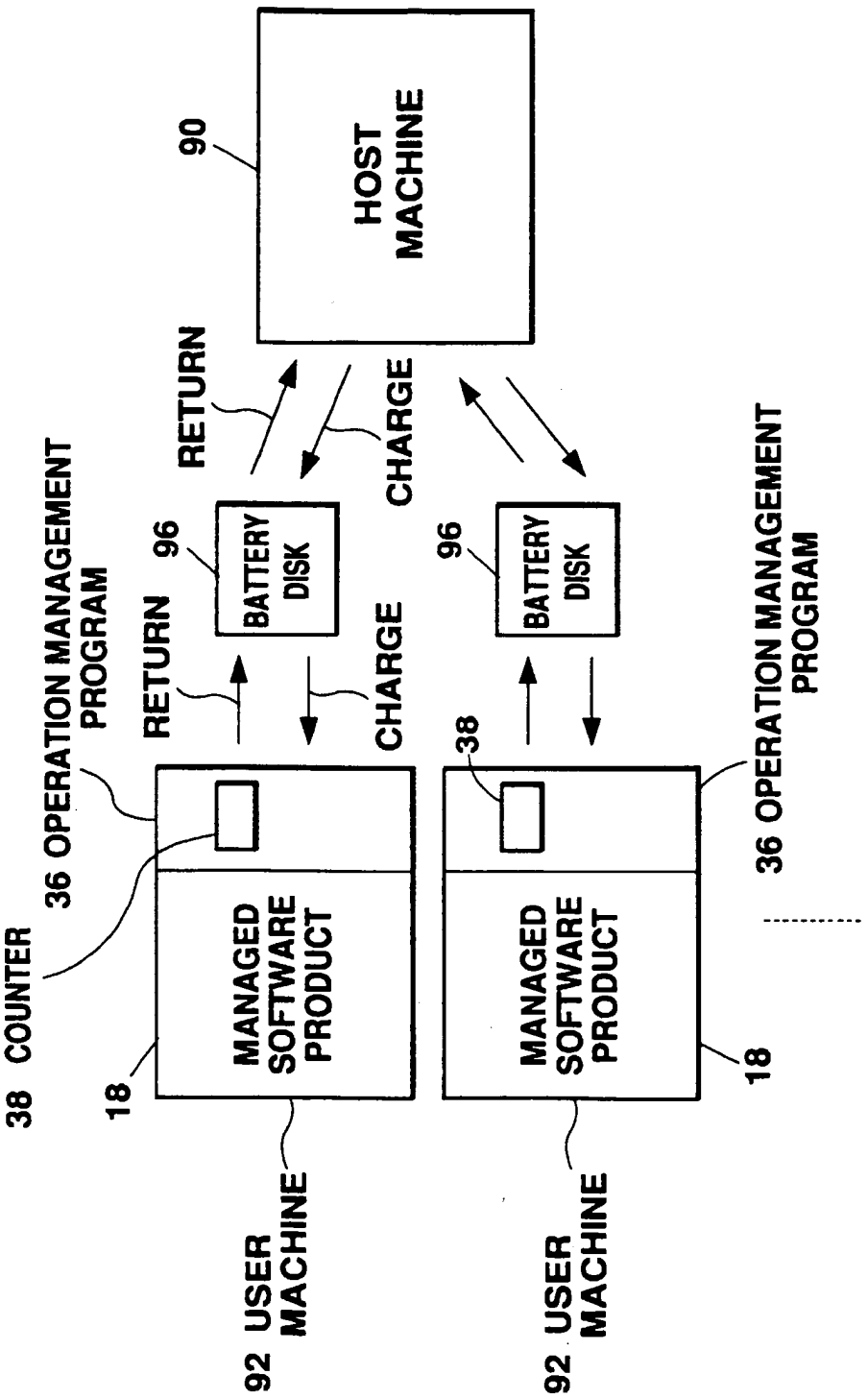


Fig. 16

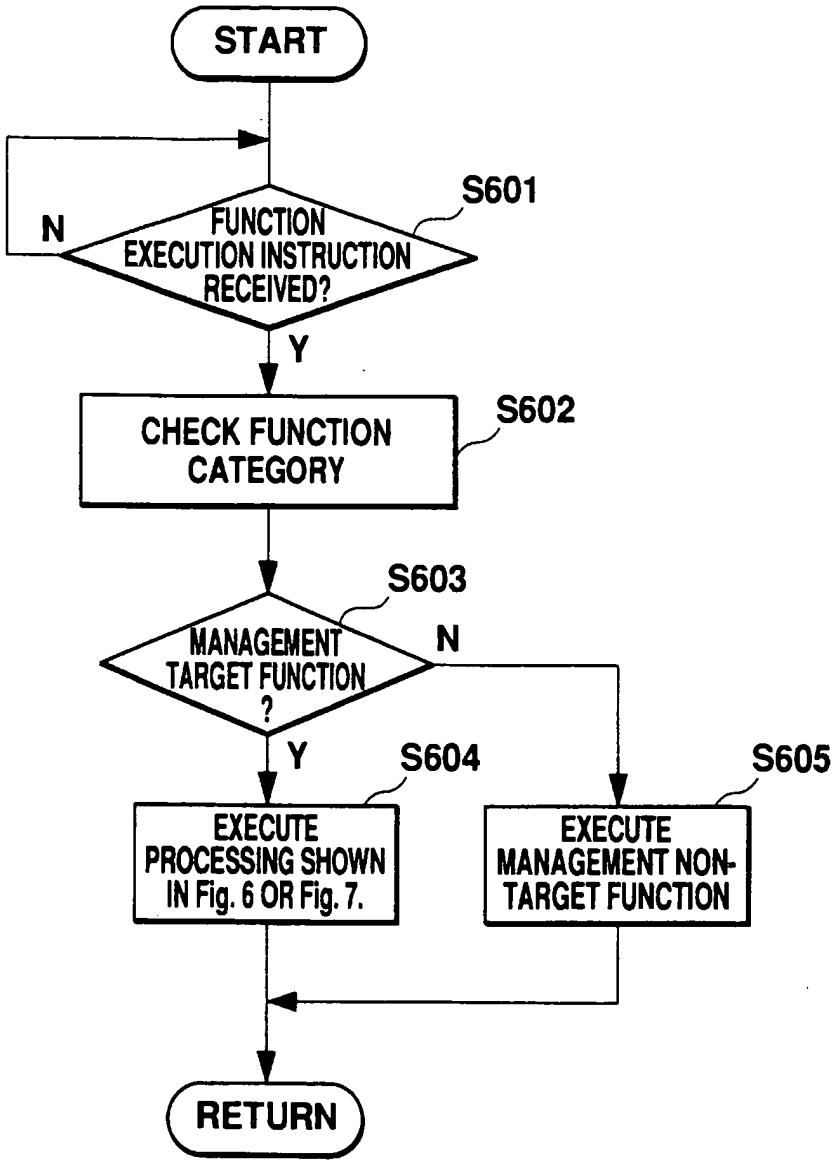


Fig. 17

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 9/08 — A1

(11) International Publication Number: WO 99/34553

(43) International Publication Date: 8 July 1999 (08.07.99)

(21) International Application Number: PCT/US98/27531

(22) International Filing Date: 30 December 1998 (30.12.98)

(30) Priority Data:
09/001,463 31 December 1997 (31.12.97) US

(71) Applicant: V–ONE CORPORATION [US/US]; Suite 300, 20250 Century Boulevard, Germantown, MD 20874 (US).

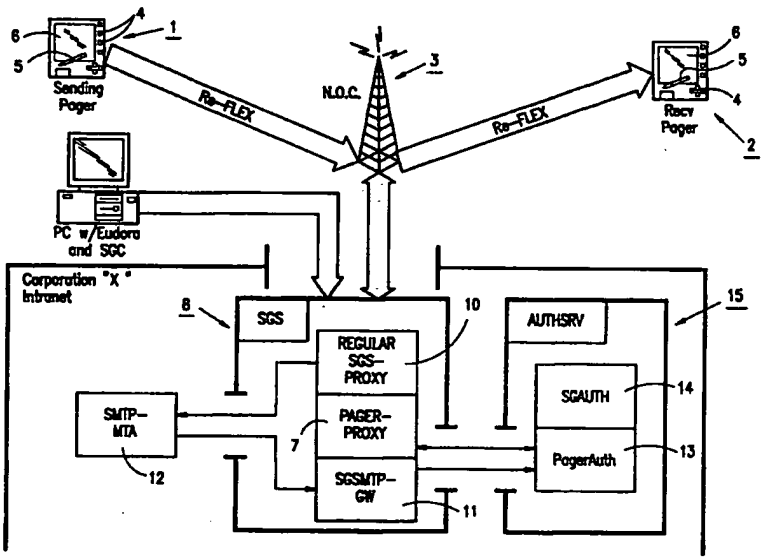
(72) Inventors: WRIGHT, Steven, R.; Apartment 21, 12010 Waterside View Drive, Reston, VA 20194 (US). BROOK, Christopher, T.; 7308 Pomander Lane, Chevy Chase, MD 20815 (US).

(74) Agents: URCIA, Benjamin, E. et al.; Bacon & Thomas, PLLC, 4th floor, 625 Slaters Lane, Alexandria, VA 22314 (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, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, 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: KEY ENCRYPTION SYSTEM AND METHOD, PAGER UNIT, AND PAGER PROXY FOR A TWO–WAY ALPHANUMERIC PAGER NETWORK



(57) Abstract

A method and system allows encryption services to be added to an existing wireless two–way alphanumeric pager (4) network by providing a pager proxy (7) which is arranged to receive an encrypted message from a sending pager (1) and re–packages it for retransmission to the destination pager (2). The sending pager encrypts the message using a session key, and encrypts the session key so that it can only be recovered by a secret key of the pager proxy. Authentication (13) of the sending pager and proxy server is provided by encryption of the session keys together with identifying data, and authentication of the message is provided by a message authentication code generated by computing a message authentication code based on the session key, identifying data, and the message.

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		

KEY ENCRYPTION SYSTEM AND METHOD, PAGER UNIT, AND PAGER PROXY FOR A TWO-WAY ALPHANUMERIC PAGER NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a system and method of encrypting messages for transmission and/or receipt by a pager, and in particular to a system and method for which uses a standard two-way wireless pager protocol to send encrypted messages over an existing paging infrastructure. The invention also relates to a pager unit capable of sending and receiving encrypted alphanumeric messages over a wireless pager network, and to a pager proxy server which provides key management functions for enabling transmission of encrypted alphanumeric messages over the wireless pager network.

2. Description of Related Art

Paging systems capable of transmitting simple alphanumeric messages and displaying the messages on a miniature two-way pager are becoming increasingly popular. Such two-way paging systems enable messages like "Meet me at the gym at 6:00" or "I love you" to be both transmitted and received by equipment that is smaller, less complex, and less intrusive than a wireless telephone. The messages are transmitted as packets containing source and destination address data formatted for transmission over

the response channel of a wireless paging network, using a protocol that allows users to respond to messages directly from their pager units without having to use a telephone.

Two-way pagers are currently offered by Motorola and Wireless Access, with national paging services being provided by MTEL, which uses Motorola's Re-FLEX™ paging protocol. The Re-FLEX™ paging protocol allows users to respond to messages using a selection of pre-programmed responses or by formatting a free-form text reply, and in addition includes a TCP/IP protocol stack that allows the user to initiate messages to subscribers on wired networks, including e-mail and fax machine addresses.

The present invention concerns a method and system for encrypting and authenticating messages transmitted over the existing pager system, using the Re-FLEX™ protocol, or over other yet-to-be-implemented paging systems in the U.S. and elsewhere which may or may not use the Re-FLEX™ protocol. Unlike previously proposed arrangements, which either rely on complex encoding schemes and sophisticated hardware at the sending and destination ends of a transmission, over transfer of keys and authentication of keys using a telephone rather than the wireless network, the present invention offers the advantages of (i) providing authenticable key encryption of messages at the source of the transmission and key decryption at the destination, with protection of the communication by keys that are unique to each pager, rather than shared, and yet with no need for a key exchange between the sending and destination pagers, (ii) using existing two-way pager designs and paging system infrastructure, and (iii) providing the encryption capabilities without adding to carrier overhead. The addition of full key encryption and authentication capabilities to an existing pager system without adding to carrier overhead or capital costs distinguishes the system and method of the invention from all previously proposed pager encryption schemes.

An example of a previously proposed pager encryption scheme is described in

U.S. Patent Nos. 5,452,356 and 5,481,255, assigned to Data Critical Corp. Although the term "encryption" is used in these patents, the patents are directed primarily to a data compression and encoding protocol for enabling transmission of large volumes of data over a wireless pager network using modified transmitting and receiving hardware, including separate computers at each end of the transmission. The only discussion of encryption in these patents is a cursory reference to "encryption" as an added security layer provided by utilizing a "commercially available algorithm" (see, *e.g.*, col. 11, lines 15-32 of U.S. Patent No. 5,452,356) during encoding of the files by a computer connected to the pager. Because all encryption and decryption in the Data Critical patents is disclosed as being carried out by software on computers connected directly to the sending and receiving pagers, the only possible ways that true key encryption could be provided for would be to use encryption keys corresponding to decryption keys common to all possible recipients of the message, to use unique keys for each potential recipient but to store the corresponding encryption keys in the sender's computer, or to exchange keys prior to a transmission. While these alternatives might be reasonable in the context of, for example, a medical paging system in which all transmissions are between doctors or trusted medical personnel, none of them are suitable for use in connection with a paging system designed to transmit simple text messages using miniature handheld paging units and which is open to the general public, both because of the hardware intensive nature of the encoding scheme and the problem of key management.

In addition to the wireless pager protocol described in the Data Critical patents the prior art includes a number of patents describing authentication or encryption schemes that are used in connection with wireless paging, but are carried out over a telephone line. The systems described in these patents are more suited to traditional one-way paging environments than with two-way protocols, even though one of the patents issued only recently, and none disclose systems that can be practically applied to the current two-way paging networks.

U.S. Patent No. 5,668,876, for example, discloses a modified pager which provides authentication of a caller. The modified pager calculates a unique response code based on a transmitted challenge code, an input personal identification number, and an internal key. The resulting response code is converted into DTMF tones and transmitted by telephone to a central computer which authenticates the caller. This system does not provide for encryption of messages, or authentication by the receiving party of communications forwarded by the central computer, and yet requires a challenge response form of authentication which requires simultaneous two-way communications, which is currently neither possible nor required by existing two-way wireless pager protocols.

U.S. Patent No. 5,285,496 describes a paging system with two options: the first is to send and receive encrypted messages using private key encryption by transmitting a clear text message over a private communications line to a local client of the pager network where the message is encrypted using a private key, and broadcast over a pager network, and the second is to send the message in clear text by telephone directly to the central control system of the pager network, where the message is encrypted. However, neither of the two options provides for encryption of the original pager transmission, which must be sent in clear text form over a telephone line, and which, in the case where a local client computer is used, provides no way to maintain centralized control. In addition, for the local client computer option, in which the address is encrypted together with the message, the destination pager must decrypt every message sent over the system in order to determine whether a message is addressed to it, which is only possible in pager networks with a very limited number of participants.

In the system described in U.S. Patent No. 5,638,450, on the other hand, reception by a pager of encrypted messages over a radio frequency pager network is made possible by having the pager transmit an encryption key via DTMF tones over a telephone line to a central office, the central office then encrypting the messages before forwarding them

to the recipient. This system does not permit outgoing messages to be encrypted, and provides no way of key encrypting messages between two pagers on the network, and again is not applicable in the context of the present invention.

It will be appreciated that none of the above patents, representing the known pager message protection proposals, describes a system that enables true key encryption and authentication capabilities to be added to a conventional two-way wireless alphanumeric paging system of the type with which the present invention is concerned, using existing pager protocols and equipment, and in which any individual can send a simply alphanumeric message by keying the message into a miniature two-way pager (or choosing from a menu of pre-stored messages), entering a destination address, and pressing a send button, the message then being retrievable by the intended recipient by a simple keystroke on the recipient's pager, with the message being encrypted by a key unique to the sending pager and decrypted by a key unique to the destination pager. In contrast, the present invention not only provides these capabilities, but adds further levels of security by using strong secret or private key based encryption algorithms, with multi-tier authentication of a transmitted packet, while permitting central registration and billing for encryption services and recovery of messages by legal authorities without adding to carrier overhead or increasing the costs of the paging service for users who do not require encryption.

All of the above advantages of the system and method of the invention are made possible through the use of a proxy server to intercept an encrypted message and repackage it for delivery to the intended recipient in a form that the intended recipient is capable of reading, thus eliminating the need for shared keys or key exchange between the sender and ultimate recipient of the message or complex, hardware-intensive encoding schemes, and allowing encrypted messages to be transmitted using existing two-way alphanumeric pager protocols. Because the invention involves key encryption and not encoding of the message, and requires knowledge by the sending and receiving

units of only one or two keys (for example, a private key unique to the pager and a server's public key), encryption being simpler to implement than encoding since it merely involves performing arithmetically combining the message with the key, the present invention can be used with existing pager hardware and protocols, and by avoiding the need for challenge/response authentication, the present invention can be used with existing channels and therefore with the existing pager infrastructure. None of the previously proposed systems and methods has these capabilities.

Not only does the use of a proxy server relieve the sending and receiving pagers of key management functions, but the manner in which the invention utilizes strong encryption capabilities, by separately encrypting the session key, further minimizes the processing resources required by the sending and receiving pagers. Essentially, encryption of the message itself can be carried out with a relatively short session key to minimize usage of the processor, while the relatively short session key can be protected by a strong encryption algorithm. Because the session key is not re-used, key integrity can easily be maintained.

SUMMARY OF THE INVENTION

It is accordingly a first objective of the invention to provide a system of adding full key encryption services to a pager network, allowing key encrypted alphanumeric messages to be sent by any pager unit registered with the encryption service provider to any other registered pager unit via the network, as well as to e-mail addresses, fax machines and other destinations capable of receiving text messages.

It is a second objective of the invention to provide a method of adding full key encryption services to a pager network, allowing key encrypted messages to be sent by any pager unit registered with the encryption service provider to any other registered pager unit via the network, as well as e-mail addresses, fax machines and other

destinations capable of receiving text messages.

It is a third objective of the invention to provide a system which allows encryption of alphanumeric messages by a paging unit for wireless transmission over a paging network in a manner which is transparent to the person sending the message, and which allows decryption and display of the messages by a receiving pager in a manner which is transparent to the person receiving the message.

It is a fourth objective of the invention to provide a method which allows encryption of messages by a paging unit for wireless transmission over a paging network in a manner which is transparent to the person sending the message, and which allows decryption and display of the messages by a receiving pager in a manner which is transparent to the person receiving the message.

It is a fifth objective of the invention to provide a system and method of adding encryption capabilities with centralized key management and unique secret keys for each user, without modification of existing pager network infrastructure or paging transmission protocols.

It is a sixth objective of the invention to provide a system and method of encrypting text messages capable of being transmitted over a pager network, which can be provided as an add-on or option to the services provided by the pager network, and which can be centrally managed using a proxy server connected to the network to provide the encryption services to subscribers who select the encryption option.

It is a seventh objective of the invention to provide a system and method of authenticating messages transmitted in encrypted form over a pager network, without the need for an authentication channel or challenge/response protocol.

It is an eighth objective of the invention to providing a standard alphanumeric pager unit with the capability of encrypting, decrypting, and authenticating messages transmitted using a two-way alphanumeric pager protocol, with minimal or no hardware modification.

It is a ninth objective of the invention to provide a proxy server arrangement which can be connected to the network operations center of a pager network in order to manage transmission of key encrypted messages over the network.

These objectives are achieved, in accordance with the principles of a preferred embodiment of the invention, by using a pager proxy server to carry out decryption of a message encrypted by a session key and received from the sending pager, and to have the pager proxy generate a new session key for re-encryption of the message transmitted to the receiving pager, with the original session key being encrypted at least by a secret key shared by the sending pager and the pager proxy server or by a public key corresponding to a private key of the pager proxy server, and the new session key being encrypted either by a secret key shared by the pager proxy server and the destination pager or a public key corresponding to a private key held by the destination pager, thereby freeing the sending and destination pagers from having to store more than one secret key or of having to carry out a direct exchange of keys, and allowing each pager on the network to be provided with a unique key.

In accordance with the principles of an especially preferred embodiment of the invention, in order to encrypt a message, the sending pager must have hard-coded into memory a unique identification number and a secret key associated with the identification number. When a user is ready to send an encrypted message, he or she begins by entering the message to be sent, after which the user is prompted for an access code to gain access to the encrypted shared key, the encrypted shared is decrypted, and a session key is generated. The message that was entered by the user is then encrypted with the

session key, and the session key is encrypted with the public key of the pager proxy server, or a shared secret key of the sending pager, and appended to the encrypted message for transmission via the network operations center to the pager proxy server.

Pager messages are formatted in accordance with standard pager protocols to include a destination header, which is generally the address or telephone number of the receiving pager, and with an additional space in the header to indicate that the message is encrypted, as will be explained in more detail below. When the network operations center receives a message that is in encrypted form, it forwards it to the encryption service center, which must at least include a pager proxy server, using an appropriate protocol, examples of which include but are not limited to TME-X and TNPP. In the illustrated embodiment, the pager proxy server is included in a gateway server in order to enable the system to package e-mail messages for transmission in encrypted form to pagers on the pager network, or to package pager messages according to an e-mail protocol for transmission over a wired network such as the Internet to an e-mail address, but it will be understood by those skilled in the art that the pager proxy may be operated as a separate unit.

In the illustrated embodiment of the invention, the pager proxy server has the role of verifying the authenticity of the message sent by the sending pager, decrypting the data with its private key or alternatively with a secret key shared with the sending pager to obtain the session key that was generated by the sending pager, and decrypting the message with the session key generated by the sending pager. Once this is accomplished, the server generates a new session key to encrypt the message with, and then encrypts the session key with a secret key shared with the destination pager or with a public key corresponding to the private key of the destination pager, or alternatively with a secret key shared with the destination pager, the two entities being appended together and sent to the recipient pager. The destination pager, after receiving the encrypted message, alerts the user and, when the user is ready to read the encrypted page, prompts him or her

for the access code to begin decryption of the appropriate shared secret key or private key, which is then used to decrypt the session key used to decrypt the message.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic diagram showing the principal elements of a pager encryption system constructed according to the principles of a preferred embodiment of the invention.

Fig. 2 is a schematic illustration summarizing the operation of the two-way pager for sending an encrypted message over a wireless network in accordance with the principles of a preferred embodiment of the invention.

Fig. 3 is a functional block diagram of a module used by a two-way pager to encrypt a message and package it for wireless transmission over a pager network to a network operations center.

Fig. 4 is a functional block diagram of a module used by a pager proxy server to authenticate the sender of an encrypted message, authenticate the message, and extract information from the message which can be used to re-package the message for transmission a destination address.

Fig. 5 is a functional block diagram of a module used by the pager proxy server to repackage a message and send it to the network operations center for transmission for re-transmission over the wireless pager network to a destination pager.

Fig. 6 is a functional block diagram showing the principal elements of a module used by a destination pager to decrypt and display a message received in encrypted form from the network operations center over the wireless paging network.

Fig. 7 is a flowchart of a preferred process corresponding to the functional block diagram of Fig. 3.

Fig. 8 is a flowchart of a preferred process corresponding to the functional block diagram of Fig. 4.

Fig. 9 is a flowchart of a preferred process corresponding to the functional block diagram of Fig. 5.

Fig. 10 is a flowchart of a preferred process corresponding to the functional block diagram of Fig. 6.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

As illustrated in Fig. 1, the system of the preferred embodiment of the invention allows encrypted communications between a sending pager and a receiving pager via a two-way wireless paging system such as M-TEL's system, using two-way alphanumeric pagers such as, but not limited to, the Motorola and Wireless Access pagers. The basic elements of the system are a sending pager 1, a receiving pager 2 which may be identical to the sending pager, and a network operations center (NOC) 3 which provides basic message forwarding and subscription management services for all communications carried by the system.

As is conventional, the sending and receiving or destination pagers (or pager units) 1 and 2 include function and data entry keys 4, and/or a stylus 5 or other data entry device, for allowing a user to input and send alphanumeric messages, and an LCD or other device 6 which allows received alphanumeric messages to be displayed. The pagers can also provide other functions such an alarm function to alert the user that a message has been received, and includes a microprocessor and circuitry capable of formatting an

input message and transmitting it to the network operations center according to an appropriate protocols, including but not limited to the ReFLEX™ protocol. The sending and receiving or destination pagers also include a memory for storing a unique user identification number (UID) that identifies a particular pager for addressing purposes, and other information such as a password that can be used to prevent unauthorized users from accessing the transmission or message display functions of the pager, as well as an addressing mode (AM) generator that is used in the pager protocol to indicate the type of addressing used by the paging system, and a timer that can be used to generate a message number.

In order to be used with the system and method of the illustrated embodiment of the invention, the pager memory must also have stored therein at least a private key of the pager unit, a corresponding public key of the pager unit, and a public key corresponding to a private key of the server, for encrypting either the message itself or a session key used to encrypt the message, and software capable of running on the included processor for performing an encryption algorithm and a decryption algorithm. In addition, according to the preferred embodiment of the invention illustrated in Figs. 2-10, the pager must be capable of generating a session key for each message to be transmitted, storing a private key unique to the pager which is used to authenticate the pager, and computing a message authentication code which is used to authenticate the message being transmitted or received.

It will be appreciated by those skilled in the art, however, that whenever a public key or private key is required, a shared secret key could be substituted using an appropriate algorithm, and that while the use of session keys is highly advantageous, the session key could also be eliminated in favor of public-private key encryption. In addition, while the illustrated system provides both encryption and decryption capabilities in at least two pagers, so that each pager can send or receive messages, the system and method of the invention could also be applied to systems in which some or

all of the pagers have reception capabilities only, *i.e.*, in which some or all of the pagers are designed to allow the pagers to receive encrypted messages originating from e-mail addresses and/or two-way pagers, but not to originate messages. Conceivably, the system and method of the invention could even be applied to systems in which at least some of the pagers are capable of sending encrypted messages, but not receiving and decrypting them, although such a system would seem to make little commercial sense. In any case, it will be appreciated that the system and method illustrated in Figs. 2-10 are intended as being illustrative in nature only, and should not be interpreted as being limitative of the scope of the invention.

As indicated above, the number of keys required of a pager to encrypt and decrypt messages is at most two, so that the key storage requirements are minimal. The encryption algorithms themselves simply involve a series of mathematical steps, and are well within the capabilities of the microprocessors used in the conventional pagers, as are message authentication code generating techniques such as CRC or SHA1. The session key used in the preferred embodiment to encrypt the message itself consists, in a practical implementation, of just sixteen characters (128 bits), and thus encryption of the alphanumeric message using RC4 or a similar stream cipher or other algorithm which makes use of a shared secret key can be accomplished without a large amount of processing resources, while strong overall protection of the transmission is still provided because the more processor intensive encryption algorithms are reserved for encryption of the relatively small session key rather than the alphanumeric message itself. Of course, the session key is not limited to a particular bit size, and it is possible for example to use 256 bit session keys, or longer or shorter session keys as desired.

In the preferred embodiment, encryption of the session key is carried out by RSA (1024 bits) but other stronger private key algorithms such as ECC PK1 (~2500 bits) can also be used, as well as shared secret key-based encryption methods such as RC4. The public-private key encryption algorithms are preferred not only because of the strong

encryption provided, but also because the permit authentication of the sender, as explained below, but legal or other considerations may also affect the choice of encryption algorithm, and thus the system of the invention is designed to permit the use of different mutually exclusive encryption algorithms by the sending and destination pagers.

The sending pager 1 illustrated in Fig. 1 transmits messages to the network operations center 3 in the form of a packet that includes a clear text applications header that tells the center to forward the text to the pager proxy server 7, which is conveniently though not essentially included in a gateway 8 capable of network communications as well as the pager encryption and decryption functions required by the present invention. Forwarding of the packet to the pager proxy or gateway server preferably involves use of a network data transfer protocol such as TME-X, although the manner in which the packet is forwarded to the proxy will depend on the wireless protocol used by the pager network and the capabilities of the network operations center. TME-X is a preferred transfer protocol for use with Re-FLEX encoded packets because of the presence of a TCP/IP stack in the standard format packets that allows the Re-FLEX™ protocol to communicate directly with computer networks.

The gateway 8 may include a general purpose proxy server 10 such as the one described in U.S. Patent No. 5,602,918, entitled "Application Level Security System And Method," and also in U.S. Patent Application Ser. No. 08/917,341, filed August 26, 1997, entitled "Multi-Access Virtual Private Network," both of which are incorporated herein by reference. The two patent documents describe a system currently available from V-One Corporation of Germantown, Maryland under the name SmartGate™ (SG in the figures) which is especially suitable for use with the pager proxy of the present invention, although the pager proxy server of the invention could also be used with other gateway servers, or without any network connection capabilities.

As illustrated, gateway 8 also includes a dedicated e-mail server or gateway 11, and e-mail protocol message transfer agent (MTA) 12 for transferring messages from the gateway server 10 to the e-mail gateway. Both the e-mail gateway 11 and pager proxy 7 may be physically incorporated in the gateway server or provided on independent or separate computers, and are connected to a pager authentication module 13 which may be physically incorporated into a general purpose gateway authentication module 14 of a separate authentication server 15, combined with the gateway server, or may be provided as an independent unit.

Computers on the network with capabilities of communicating with the general purpose proxy server are represented in Fig. 1 by computer 16, and include gateway client software that permits the computer to establish a secured communications path to the gateway server, as well as an e-mail program which packages messages in an appropriate format such as that provided by the SMTP protocol for transmission over the secured communications path established by the gateway client software. An example of an e-mail program is "Eudora™," although the use of standard protocols such as SMTP and Re-FLEX™ allows any e-mail program to communicate with the gateway and thence with the pager network, so that the system of the invention is not limited to use in connection with any particular e-mail program, the conventional pager network already being equipped to handle e-mail transmissions to or from the wireless network. The invention may be considered to apply equally to pager-to-pager communications, pager-to-email communications, and email-to-pager communications. In addition, it is possible that the invention could be adapted to communications originating from a fax machine, in which case the clear packet transmitted by the fax machine over a telephone line would be processed by a facsimile proxy for packaging and encryption by the pager proxy, and messages addressed to the fax machine would be decrypted by the pager proxy and forwarded to the facsimile proxy for transmission as clear text over a telephone line, the principles of the invention still being applicable to the encryption and decryption of the messages by the pager proxy and sending or receiving pagers.

Turning to the specific embodiment illustrated in Figs. 2-10, the system and method of the invention take the form of modifications to the header of the transmission packet sent by the sending pager 1 and/or the pager proxy 7. Essentially in order to send messages over the paging system, the sending pager and pager proxy, (or pager proxy alone in the case of a message originating from computer 16 or a source of clear text messages such as a facsimile machine) generates a header which includes the information necessary to enable processing by the recipient of the packet, and in the case of the pager proxy, for forwarding of a repackaged packet to a destination address. The header should at least include the session key encrypted message, the encrypted session key, a sender identification number, and a destination header or address, but because the header format will vary if a protocol other than Re-FLEX™ is used, it should be appreciated that the other information contained in the illustrated header, and the position of the information, can be varied without departing from the scope of the invention, and the invention is intended to encompass headers formatted for other alphanumeric wireless paging protocols, as well as for encryption algorithms and authentication protocols other than the specific algorithms and protocols indicated.

Fig. 2 illustrates the format of the preferred header, which is divided into three fields. It is to be understood that while the illustration refers to the communication between the sending pager and the pager proxy, the same header will be used for the communication between the pager proxy and the destination pager, with appropriate substitutions of addresses and keys as explained in more detail below. As shown in Fig. 2, the first field is a clear text field that contains the encryption method indicator EM, pager addressing mode (AM), and user identification number (UID) (sometimes referred to as a PIN, but not to be confused with the password entered by the user to access pager functions), while the second field contains the encrypted session key (SESKey1) and various data referred to as "header data" (HdrData) including the destination header or address (DH) and a message authentication code (MAC), the information in the second field being encrypted by the unique private key of the sending pager (pv.sender) in order

to authenticate the sender, and by a public key corresponding to a private key held by the server (pb.server) in order to protect the contents of this field. The third field contains the message encrypted by the session key.

The various fields illustrated in Fig. 2 may be formatted in any convenient manner permitted or required by the protocol used to package the data in the fields for transmission, but in the illustrated example most or all of the data in at least fields one and two can conveniently be in hexadecimal format. Whenever the drawings illustrate a hexadecimal number, the number ## will be preceded by a "0x" to form 0x##.

The encryption method indicator EM indicates which of the possible encryption methods handled by the server is being used to encrypt the session key and other information in field 2, so that the session key can be recovered and used to decrypt the encrypted message in field 3. As indicated above, possible encryption methods include the RC4 secret key encryption method, which requires the parties to the communication to have a shared secret key that is used for both encryption or decryption, and the RSA public key encryption method, which is the method illustrated in Fig. 2. The indicator itself is simply a number assigned to the encryption method. While any given pager will generally have only a single encryption method stored in memory, it is possible for the pager proxy to be arranged to handle multiple different methods and thus need to have an indication of the type of encryption method, to accommodate different pager systems or legal requirements, particularly if international pager traffic is involved.

The addressing mode (AM) indicates the type of address involved. For example, in the U.S., pager addressing modes are assigned one application header, while e-mail addressing modes are assigned another application header. This indicator may not be necessary in all protocols since the destination header may be unique to a specific type of address, but is included in field 1 as part of the Re-FLEX™ protocol.

The user identification number (UID) included in clear text in field 1 and in encrypted form in field 2, is the unique address assigned to the pager, and is used to indicate the source of the message so as to enable the pager proxy to retrieve the appropriate public decryption key (pb.sender), and for use in authentication of the sender and for display by a receiving pager. Preferably, this number is hard-coded into memory so that it cannot easily be altered, and in current U.S. paging systems is in the form of a ten digit number.

The header data (HdrData) of the second field includes an application header (AH), which included in a field having variable length and string value, the address mode and destination header (AM/DH), the user identification number (UID), which is the same as the one included in field 1, and a message number (MSGNO) and message authentication code (MAC). In addition, e-mail address protocols require a byte indicative of address length to be added where the address mode indicates an e-mail address.

For purposes of the present invention, the message number can be any arbitrary number, although the use of a time-related reference, as allowed by the Re-FLEX protocol, is useful for account tracking or billing purposes, and in addition can be used to ensure that received message is not a recording of a message sent earlier and intercepted by an unauthorized party. For example, the message number has previously been defined as the number of seconds since January 1, 1970.

The message authentication code is a checksum used to verify that the recovered message is identical to the original message, and may be computed using an error correction code function such the cyclic recovery code (CRC) function, with CRCs being used in the illustrated embodiment or, alternatively, by computing a hash or one-way combination of the header data with the message and the session key, using an algorithm such as SHA1. By combining the message with other data to obtain the message

authentication code in a way that can only recreated if the data used to recreate the code is the same as the data originally used to generate the code, the code can be used to authenticate the message, *i.e.*, to verify that the message has not been altered since the time when the code was first generated, as will be described in more detail below. It will be appreciated that the exact form of the message authentication code is not a part of the present invention, and that any message authentication code may be used so long as it can be used to authenticate the message in the manner described below.

The three blocks above the header data in Fig. 2 indicate the source of the data for the various fields. The manner in which the data is combined to form the fields is described in more detail in connection with Figs. 3-10, but the sources of the data may be summarized as (i) information entered by the user, which consists of the message (MSG) and the recipient address which forms the destination header, (ii) information stored in memory, including private and public keys of the pager, a public key of the pager proxy server, an access code which is to be compared with an access code input by the user, the encryption method indicator (EM), the user identification number (UID), and the application header, and (iii) information generated at runtime, *i.e.*, during assembly of the packet header, including the session key (SESKey), the message number (MSGNO), the addressing mode (AM), and the message authentication code (MAC).

The details of the manner in which the data shown in Fig. 2 is assembled by sending pager 1 to form the header shown in Fig. 2 is illustrated in the functional block diagram of Fig. 3, as well as the flowchart of Fig. 7. As illustrated in Fig. 3, the pager 1 includes a user input 20 connected to keys 4 or stylus 5, which supplies the destination header (DH) to a functional block 21 which assembles the header data (HdrData), and to a functional block 22 which computes the message authentication code (MAC). In addition, the user input 20 supplies the message to functional block 28, the output of which is field 3 of the header.

Pager 1 also includes a memory 24 which stores the encryption method (EM), the application header (AH), the user identification number (UID) and the encryption method identifier (EM), which are supplied directly to functional block 23 for inclusion in field 1, the user identification number and application header being also supplied to functional block 21 for inclusion in the header data, which in turn is supplied to functional block 22 for inclusion in the message authentication code. The address mode (AM), which is associated with the destination header (DH) in the header data is generated by an address mode generator 25 which can be in the form of a look-up table, device that reads a particular identifying bit in the destination header, or other device, and the message number can be generated by a counter, timer, or other device 26 depending on the nature of the message number. Finally, the session key (SESKey1) for this embodiment of the invention is an eight character string generated by a random or pseudorandom number generator 27, which supplies the session key to functional block 28 for use in encrypting the message (MSG), to functional block 22 for inclusion in the message authentication code, and to functional block 29 for encryption together with the header data by the private key of the sender. The output of functional block 29 is supplied to functional block 30 for encryption by the public key of the server, the output of block 30 serving as field 2 of the header for the packet transmitted by the sending pager.

It will be appreciated by those skilled in the art that any of the functional blocks and data or number generators illustrated in Fig. 3, or in Figs 4-6, may be implemented either by hardware or software, and that while distinguishable by function, the functions may be carried out using common subroutines, hardware, or software.

Turning to Fig. 4, the pager proxy 7 includes a database of public keys corresponding to the unique public keys of pagers registered with the encryption service provider that operates the proxy server. The database is accessed by functional block 31 according to the clear text user identification number (UID) present in the header of a packet forwarded to the pager proxy by the network operations center. Field 2 is

decrypted by functional block 32 using the private key of the server (pv.server) and by functional block 33 using the public key of the sender (pb.sender) to recover the session key, and the user identification number (UID) recovered from field 2 is compared by functional block 34 with the user identification number of field 1 to verify the authenticity of field 2 and recover the session key (SESKey1). A functional block 35 then uses the session key to decrypt the message (MSG).

The message recovered by the pager proxy is authenticated in functional block 37, by comparing the message authentication code recovered from field 2 with the output of a functional block 36 that computes the message authentication code based on the destination header (DH), application header (AH), user identification number (UID), message number (MSGNO), and session key (SESKey1) recovered from field 2, and the message recovered from field 3. The message, session key, and header data (HdrData) are then made available by functional block 38 to an encryption or repackaging module, illustrated in Fig. 5, for repackaging in a way that will enable decryption by a destination pager.

As shown in Fig. 5, the application header (AH) and message number (MSGNO) received from functional block 38 is provided to functional blocks 41 and 42 for inclusion in the header data and message authentication code, while the address mode (AM) and encryption method (EM) obtained from field 1 of the packet received from the sender is passed to functional block 43 or regenerated for inclusion as clear text in the packet header. In order to permit decryption and authentication of the repackaged header by the receiving pager, however, the destination header (DH) and user identification number (UID) are swapped, so that the original destination header is supplied by the pager proxy to functional blocks 41, 42, and 43 as the user identification number (UID), and the original user identification number are supplied to functional blocks 41 and 42 as the destination header (DH). Functional block 42 generates a message authentication code based on the new destination header (DH), application header (AH), user

identification number (UID), message number (MSGNO), while a new session key (SESKey2) is generated by functional block 44 in the same manner as functional block 27 shown in Fig. 3, and the resulting message authentication code (MAC) together with the new session key and header data from functional block 41 are encrypted by functional block 45 using the private key of the server (pv.server) before being sealed by functional block 46 using the public key of the destination pager (pb.recipient) and included in the header as field 2. Functional block 47 receives the message and new session key and re-encrypts the message using the new session key and an algorithm such as RC4 to generate field 3, fields 1-3 being assembled into a packet 50 for transmission to the destination pager 2 via the network operations center 3.

Again, those skilled in the art will appreciate that all of the functional blocks illustrated as being present in the proxy server and/or proxy authentication module may be implemented as software, hardware, or a combination of software and hardware, and may be varied depending on the encryption method and requirements of the pager protocol.

In addition, those skilled in the art will appreciate that the illustrated embodiment could be modified by eliminating the session key and instead using public key encryption of the message. Alternatively, instead of having the pager proxy perform any decryption of the message, the original session key could simply be re-encrypted by the pager proxy using at least the public key of the destination pager as described above, or a secret key shared with the destination pager, in which the encrypted message would simply be forwarded to the destination pager unit with the session key re-encrypted so that it can be recovered by the destination pager. While neither of these options is currently preferred because elimination of the session key leaves transmissions vulnerable to recording, and elimination of message decryption by the pager proxy makes message authentication more difficult, they should nevertheless be considered to be within the scope of the invention.

Turning to Fig. 6, the destination pager 2 includes functional blocks mirroring those of the server for decrypting messages and authenticating packets received from the pager proxy 7 via the network operations center 3. These include functional block 51 for retrieving the server public key (pb.server) from memory, functional blocks 52 and 53 for decrypting the field 2 using the recipient private key (pv.recipient) and the server public key, functional block 54 for comparing the user identification number recovered from field 2 with the user identification number in field 1, functional block 56 for decrypting the message (MSG) using the session key (SESKey2) recovered from field 2, and functional blocks 57 and 58 for generating a message authentication code and comparing it with the message authentication code recovered from field 2. It will be noted that functional block 57 may also be used to generate a message authentication code for an outgoing message, avoiding duplication of the hardware or software which performs this function.

Finally, destination pager 2 includes a functional block 59 for displaying the message (MSG) and destination header (DH) corresponding to the user identification number of the sending pager, and for alerting the user as necessary that a message has been received. The display is identical to that used for an unencrypted message, and thus the decryption operation is entirely transparent to the user.

The method steps that implement the functions illustrated in Figs. 3-6 are as follows:

First, as shown in Fig. 7, upon input of a message and destination address by the user of a pager (step 100), which may follow the input and verification of a password (not shown), a message number, address mode, and session key are generated (step 110) and the encryption method identifier, application header, user identification number, server public key, and sender private key are retrieved from memory (step 120). The encryption method identifier, address mode, and user identification number are included in field 1 (step 130), a message authentication code based on the destination header, application

header, user identification number, message number, message, and session key is computed (step 140), and the application header, user identification number, destination header, message number, message authentication code, and session key are encrypted by the private key of the sending pager (step 150) and then by the public key of the pager proxy (step 160) to obtain field 2 of the packet header. Finally, the message is encrypted by the session key (step 170) to obtain field 3, and the packet header is transmitted via the network operations center to the pager proxy (step 180).

Upon receipt by the pager proxy, as shown in Fig. 8, the public key of the sending pager is retrieved based on the user identification number in field 1 (step 200), and field 2 of the packet is decrypted by the private key of the server (step 210) and then by the public key of the sending pager (step 220) based on the encryption method identified by the identifier in field 1. Authentication of the sender is provided by comparing the user identification number recovered from field 2 with the user identification number in field 1 (step 230), the message included in field 3 is decrypted using the session key recovered from field 2 (step 240), and authentication of the message is provided by generating a message authentication code based on the destination header, application header, user identification number, message number, and session key recovered from field 2 together with the decrypted message (step 250), and by then comparing the computed message authentication code with the message authentication code recovered from field 2 (step 260).

As illustrated in Fig. 9, after authenticating the information contained in field 2, the proxy server generates a new session key (step 300), encrypts the message using the new session key (step 310), assigns the original user identification as the new destination header and the original destination header as the new user identification number, computes a new message authentication code (step 330), encrypts the address header, message number, new user identification number, new destination header, new session key, and new message authentication code using the private key of the server (step 340),

encrypts the result of step 340 using the public key of the destination pager (step 350), and assembles the header and packet for RF transmission to the destination pager via the network operations center (step 360).

As illustrated in Fig. 10, upon receipt by the destination pager, as shown in Fig. 8, the public key of the pager proxy server is retrieved based on the user identification number in field 1 (step 400), and field 2 of the packet is decrypted by the private key of the destination pager (step 410) and then by the public key of the pager proxy server (step 420) based on the encryption method identified by the identifier in field 1. Authentication of the sender is provided by comparing the user identification number recovered from field 2 with the user identification number in field 1 (step 430), the message included in field 3 is decrypted using the session key recovered from field 2 (step 440), and authentication of the message is provided by computing a message authentication code based on the destination header, application header, user identification number, message number, and session key recovered from field 2 together with the decrypted message (step 450), and by then comparing the computed message authentication code with the message authentication code recovered from field 2 (step 460). Finally, after authentication of the user identification number and message, the user is alerted that a message has been received and the decrypted message and information contained in the destination header are displayed at the request of the user (step 470).

Having thus described a preferred embodiment of the invention in sufficient detail to enable those skilled in the art to practice the invention, it is nevertheless anticipated that numerous variations and modifications of the invention will occur to those skilled in the art, and it is intended that all such variations and modifications be included within the scope of the invention. For example, although the preferred embodiment of the invention has the pager proxy re-package the message by first decrypting it, and then re-encrypting it using a new session key, it is also within the scope of the invention to have the pager proxy decrypt only the session key and re-encrypt the same session key using

the public key or shared secret key of the destination pager. Accordingly, it is intended that the above description not be taken as limiting, but rather that it be defined solely by the appended claims.

I claim:

1. A system for adding encryption services to an existing pager network, the pager network including a network operations center which provides a means for receiving an alphanumeric message from any of a plurality of handheld pager units and forwarding the alphanumeric message to another of the plurality of handheld pager units, at least two of said pager units comprising:

means for inputting an alphanumeric message and a destination address;

means for including the alphanumeric message in a packet for transmission to the destination address by wireless transmission via the network operations center;

means for receiving an alphanumeric message from the network operations center; and

means for displaying the alphanumeric message received from the network operations center,

wherein the system for adding encryption services comprises:

means in at least one of said pager units for encrypting a message and transmitting the encrypted message via the network operations center to another of said pager units;

means in said another one of said pager units for decrypting and displaying the encrypted message; and

a pager proxy server including means for receiving a packet containing the encrypted message that has been sent to the network operations center, decrypting at least a portion of the packet, and re-encrypting said portion of the packet for delivery to said another of said pager units via said network operations center.

2. A system as claimed in claim 1, wherein said means for encrypting the message comprises means for encrypting the message by a secret key.

3. A system as claimed in claim 2, wherein said secret key is a first session key generated by a sending pager unit, said sending pager unit further comprising means for encrypting said first session key by a public key corresponding to a private key held by the pager proxy server so that the session key can be recovered only by the paging proxy server.

4. A system as claimed in claim 3, wherein said sending pager unit further comprises means for encrypting at least the first session key by a private key of the sending pager unit, and wherein said pager proxy server includes means for retrieving a public key corresponding to the private key of the sending pager unit for use as a first level authentication of the sending pager unit.

5. A system as claimed in claim 4, further comprising means for appending a unique user identification number of the sending pager unit to the header in clear text form, said user identification number being hard-coded into the sending pager unit.

6. A system as claimed in claim 5, wherein said means for encrypting at least the session key by a private key of the sending pager unit also encrypts the user identification number of the sending pager unit, and said paging proxy server includes means for decrypting the encrypted user identification number together with the first session key and comparing it with the clear text user identification number in order to authenticate the contents of the field containing the encrypted user identification number and first session key.

7. A system as claimed in claim 4, wherein the sending pager unit further comprises means for generating a first message authentication code based on various header data and the message and encrypting the various information together with the session key and the first message authentication code using the private key of the sending pager unit, and wherein the pager proxy server further comprises means for decrypting the various header

data, first message authentication code, and session key using a public key corresponding to the private key of the sending pager unit, decrypting the message using the session key, generating a second message authentication code based on the message and various header data, and comparing the first message authentication code with the second message authentication code in order to authenticate the message.

8. A system as claimed in claim 7, wherein said message authentication code is an error correction code function.

9. A system as claimed in claim 7, wherein said various header data includes at least a user identification number of the sending pager and a destination header corresponding to the input address of the destination pager.

10. A system as claimed in claim 9, wherein said various header data further includes a message number and application header.

11. A system as claimed in claim 4, wherein the sending pager further comprises means for adding an encryption method identifier in clear text to the packet header.

12. A system as claimed in claim 4, wherein an encryption algorithm used to encrypt the first session key is a public-private key encryption algorithm.

13. A system as claimed in claim 4, wherein said secret key is a first session key generated by a sending pager unit and said first session key is encrypted by a stream cipher that uses a shared secret key.

14. A system as claimed in claim 2, wherein said sending pager unit further comprises means for generating an address mode and appending the address mode in clear text to the packet header.

15. A system as claimed in claim 14, wherein said address mode indicates an address type selected from the group consisting of pager address types and e-mail address types, and wherein the pager proxy server is connected to a computer network gateway server and includes means for re-packaging said message in an e-mail packet and transmitting the e-mail packet via said computer network server to an e-mail address.

16. A system as claimed in claim 15, further comprising means for receiving e-mail packets from said computer network gateway server, and re-packaging said e-mail packets for transmission to the destination pager unit via said network operation center, and means for repackaging packets received from the network operations center for forwarding to an e-mail server.

17. A system as claimed in claim 1, wherein said means included in the pager proxy server for decrypting at least a portion of the packet includes means for decrypting, using a secret key, a portion of the packet containing a first session key used by a sending pager unit to encrypt said portion of the packet.

18. A system as claimed in claim 17, wherein said pager proxy server further includes means for decrypting said message using said first session key, means for generating a second session key, and means for re-encrypting the message using the second session key.

19. A system as claimed in claim 18, wherein said means for re-encrypting said portion of the packet includes means for encrypting the second session key by a secret key.

20. A system as claimed in claim 19, wherein said means for encrypting said portion of the packet by a secret key includes means for re-encrypting the second session key by a public key corresponding to a private key of a destination pager unit.

21. A system as claimed in claim 20, wherein said means for encrypting said portion of the packet by a secret key further includes means for, before re-encrypting the second session key by the public key corresponding to a private key of the destination pager, encrypting the second session key and various additional data by a private key of the pager proxy server.

22. A system as claimed in claim 21, wherein said additional data includes a second user identification number, said second user identification number corresponding to a first destination header included in said decrypted portion of the packet received from the sending pager unit, and wherein said destination paging unit includes means for comparing said second user identification number encrypted with said second session key to a clear text version of the second user identification number received from the pager proxy server in order to authenticate the pager proxy server.

23. A system as claimed in claim 22, wherein said additional data includes a second destination header corresponding to the first user identification number, and wherein said second pager unit includes means for displaying information included in said second destination header in order to indicate an address of the sending pager unit.

24. A system as claimed in claim 22, wherein said additional data includes a second destination header corresponding to the first user identification number, a message number recovered from said decrypted portion of the packet received from the sending pager unit, and an application number.

25. A system as claimed in claim 22, wherein said pager proxy server further comprises means for generating a message authentication code based on said message, said second session key, and said additional data, and said destination pager unit includes means for recovering said additional data and computing a message authentication code based on the additional data, said second session key, and said message in order to authenticate said message.

26. An encryption method according to which encryption services may be added to an existing two-way wireless pager network, the pager network including a network operations center which provides a means for receiving an alphanumeric message from any of a plurality of handheld pager units and forwarding the alphanumeric message to another of the plurality of handheld pager units, comprising the steps of:

causing one of said pager units to perform the steps of encrypting a message, including the encrypted message in a wireless transmission packet, and transmitting the encrypted message from said one of said pager units to a pager proxy server via the network operations center;

causing the pager proxy server to perform the steps of receiving the encrypted message and repackaging it for transmission to another of said pager units via the network operations center; and

causing said another of said pager units to perform the steps of decrypting and displaying the encrypted message.

27. A method as claimed in claim 26, wherein the step of encrypting the message comprises the step of encrypting the message by a secret key corresponding to a secret key of the pager proxy server so that the session key can only be recovered by the paging proxy server.

28. A method as claimed in claim 26, wherein said secret key is a first session key generated by a sending pager unit, and wherein said sending pager unit further performs the step of encrypting said first session key by a public key corresponding to a private key held by the pager proxy server.

29. A method as claimed in claim 27, wherein said sending pager unit further performs the step of encrypting at least the first session key by a private key of the sending pager unit, and wherein said pager proxy server performs the step of retrieving a public key corresponding to the private key of the sending pager unit for use as a first

level authentication of the sending pager unit.

30. A method as claimed in claim 29, further comprising of the step of appending a unique user identification number of the sending pager unit to the header of the transmission to the paging proxy server in clear text form, said user identification number being hard-coded into the sending pager unit.

31. A method as claimed in claim 30, wherein said step of encrypting at least the session key by a private key of the sending pager unit includes the step of encrypting the user identification number of the sending pager unit, and said paging proxy server further performs the steps of decrypting the encrypted user identification number together with the first session key and comparing it with the clear text user identification number in order to authenticate the contents of the field containing the encrypted user identification number and first session key.

32. A method as claimed in claim 29, wherein the sending pager unit further performs the step of computing a first message authentication code based on various header data and the message and encrypting the various information together with the session key and the first message authentication code using the private key of the sending pager unit, and wherein the pager proxy server further performs the steps of decrypting the various header data, first message authentication code, and session key using a public key corresponding to the private key of the sending pager unit, decrypting the message using the session key, generating a second message authentication code based on the message and various header data, and comparing the first message authentication code with the second message authentication code in order to authenticate the message.

33. A method as claimed in claim 32, wherein said message authentication code is an error correction code function.

34. A method as claimed in claim 32, wherein said various header data includes at least the user identification number of the sending pager and a destination header corresponding to the input address of the destination pager.

35. A method as claimed in claim 34, wherein said various header data further includes a message number and application header.

36. A method as claimed in claim 34, wherein the sending pager further performs the step of adding an encryption method identifier in clear text to the packet header.

37. A method as claimed in claim 29, wherein an encryption algorithm used to encrypt the first session key is a public-private key encryption algorithm.

38. A method as claimed in claim 27, wherein said secret key is a first session key generated by a sending pager unit and said first session key is encrypted by a stream cipher that uses a shared secret key.

39. A method as claimed in claim 37, wherein said sending pager unit further performs the step of generating an address mode and appending the address mode in clear text to the packet header.

40. A method as claimed in claim 39, wherein said address mode indicates an address type selected from the group consisting of pager address types and e-mail address types, and wherein the pager proxy server is connected to a computer network gateway server and further performs the step of re-packaging said message in an e-mail packet and transmitting the e-mail packet via said computer network server to an e-mail address.

41. A method as claimed in claim 40, further performs the steps of receiving e-mail packets from said computer network gateway server, and re-packaging said e-mail

packets for transmission to the destination pager unit via said network operation center.

42. A method as claimed in claim 26, wherein said step of repackaging the encrypted message for transmission includes the step of causing the pager proxy server to encrypt, using a secret key, a portion of the packet containing a first session key used by a sending pager unit to encrypt said portion of the packet.

43. A method as claimed in claim 42, wherein said pager proxy server further performs the steps of decrypting said message using said first session key, generating a second session key, and re-encrypting the message using the second session key.

44. A method as claimed in claim 43, wherein said pager proxy server further performs the step of encrypting the second session key by a secret key.

45. A method as claimed in claim 44, wherein said step of encrypting said portion of the packet by a secret key includes the step of re-encrypting the second session key by a public key corresponding to a private key of a destination pager unit.

46. A method as claimed in claim 45, wherein said step of encrypting said portion of the packet by a secret key further includes the step of, before re-encrypting the second session key by the public key corresponding to a private key of the destination pager, encrypting the second session key and various additional data by a private key of the pager proxy server.

47. A method as claimed in claim 46, wherein said additional data includes a second user identification number, said second user identification number corresponding to a first destination header included in said decrypted portion of the packet received from the sending pager unit, and wherein said destination paging unit perform the step of comparing said second user identification number encrypted with said second session key

to a clear text version of the second user identification number received from the pager proxy server in order to authenticate the pager proxy server.

48. A method as claimed in claim 47, wherein said additional data includes a second destination header corresponding to the first user identification number, and wherein said second pager unit performs the step of displaying information included in said second destination header in order to indicate an address of the sending pager unit.

49. A method as claimed in claim 47, wherein said additional data includes a second destination header corresponding to the first user identification number, a message number recovered from said decrypted portion of the packet received from the sending pager unit, and an application number.

50. A method as claimed in claim 47, wherein said pager proxy server further performs the step of computing a message authentication code based on said message, said second session key, and said additional data, and said destination pager unit further performs the step of recovering said additional data and computing a message authentication code based on the additional data, said second session key, and said message in order to authenticate said message.

51. A two-way alphanumeric pager unit, comprising:

means for inputting a message and a destination address;

means for generating a session key;

means for encrypting the message using the session key;

means for protecting the session key so that it can only be recovered by a pager proxy server;

means for transmitting the message via a wireless pager network to the pager proxy server;

means for receiving an encrypted message transmitted via the wireless pager network from the pager proxy server;

means for decrypting an encrypted session key appended to the message;

means for decrypting the encrypted message transmitted from the pager proxy server using the decrypted session key; and

means for displaying the message.

52. A pager unit as claimed in claim 51, wherein said means for protecting the session key comprises means for encrypting the session key by a secret key.

53. A pager unit as claimed in claim 52, wherein said secret key is a first session key generated by the pager unit, said sending pager unit further comprising means for encrypting said first session key by a public key corresponding to a private key held by the pager proxy server.

54. A pager unit as claimed in claim 53, further comprising means for appending a unique user identification number of the pager unit to the header in clear text form, said user identification number being hard-coded into the pager unit.

55. A pager unit as claimed in claim 54, wherein said means for encrypting at least the session key by a secret key also encrypts the user identification number of the sending pager unit, said encrypted user identification number being compared by the pager proxy server with a clear text version of the user identification number transmitted with a packet header in order to authenticate the pager unit.

56. A pager unit as claimed in claim 55, wherein the pager unit further comprises means for computing a message authentication code based on various header data and the message, and means for encrypting the various information together with the session key and the message authentication code using a private key of the sending pager unit in order to provide a means for authentication by the pager proxy of the message.

57. A pager unit as claimed in claim 56, wherein said message authentication code is an error correction code function.

58. A pager unit as claimed in claim 57, wherein said various header data includes at least the user identification number of the pager unit and a destination header corresponding to the input address of a destination pager.

59. A pager unit as claimed in claim 58, wherein said various header data further includes a message number and application header.

60. A pager unit as claimed in claim 52, wherein the pager unit further comprises means for adding an encryption method identifier in clear text to a packet header.

61. A pager unit as claimed in claim 60, wherein an encryption algorithm used to encrypt the first session key is a public-private key encryption algorithm.

62. A pager unit as claimed in claim 60, wherein said secret key is a first session key generated by a sending pager unit and said first session key is encrypted by a stream cipher that uses a shared secret key.

63. A pager unit as claimed in claim 62, wherein said pager unit further comprises means for generating an address mode and appending the address mode in clear text to the packet header.

64. A pager unit as claimed in claim 62, wherein said address mode is selected from the group consisting of pager address types and e-mail address types, and wherein the pager proxy server is connected to a computer network server and includes means for re-packaging said message in an e-mail packet and transmitting the e-mail packet via said computer network server to an e-mail address.

65. A pager proxy server, comprising:

means for receiving a message encrypted by a session key, the session key being encrypted and appended to the encrypted message, from a network operations center of a pager network;

means for recovering the session key using a secret key of the server;

means for authenticating the sender of the message; and

means for re-transmitting the message encrypted by a session key in a manner which enables decryption of the message only by a holder of a second secret key.

66. A server as claimed in claim 65, wherein said means for re-transmitting the message comprises means for decrypting the message using the first session key, re-encrypting the message using a second session key, and encrypting the second session key.

67. A server as claimed in claim 66, wherein said first secret key is a private key held by the pager proxy server.

68. A server as claimed in claim 67, further comprising means for retrieving a public key corresponding to a private key of a sending pager unit for use as a first level authentication of the sending pager unit.

69. A server as claimed in claim 68, further comprising means for decrypting the a user identification number of the sending pager unit together with the session key and comparing it with a clear text user identification number in order to authenticate the contents of the field containing the encrypted user identification number and session key.

70. A server as claimed in claim 69, further comprising means for decrypting various header data, a first message authentication code, and a session key using a public key corresponding to the private key of the sending pager unit, decrypting the message using

the session key, generating a second message authentication code based on the message and various header data, and comparing the first message authentication code with the second message authentication code in order to authenticate the message.

71. A server as claimed in claim 70, wherein said message authentication code is an error correction code function.

72. A server as claimed in claim 70, wherein said various header data includes at least the user identification number of the sending pager and a destination header corresponding to the input address of the destination pager.

73. A server as claimed in claim 72, wherein said various header data further includes a message number and application header.

74. A server as claimed in claim 73, wherein said encryption method is a public-private key encryption algorithm.

75. A server as claimed in claim 73, wherein said encryption method is RC4 secret key encryption.

76. A server as claimed in claim 72, further comprising means for receiving e-mail packets from said computer network server, and re-packaging said e-mail packets for transmission to the destination pager unit via said network operation center.

77. A system for adding encryption services to an existing pager network, the pager network including a network operations center which provides a means for receiving an alphanumeric message from any of a plurality of handheld pager units and forwarding the alphanumeric message to another of the plurality of handheld pager units, at least one of said pager units comprising:

means for inputting an alphanumeric message and a destination address;

means for including the alphanumeric message in a packet for transmission to the destination address by wireless transmission via the network operations center;

means for receiving an alphanumeric message from the network operations center; and

means for displaying the alphanumeric message received from the network operations center,

wherein the system for adding encryption services comprises:

means in at least one of said pager units for decrypting and displaying an encrypted message; and

a pager proxy server including means for receiving a packet containing the encrypted message, decrypting at least a portion of the packet, and re-encrypting said portion of the packet for delivery to said at least one of said pager units via said network operations center.

78. An alphanumeric pager unit, comprising:

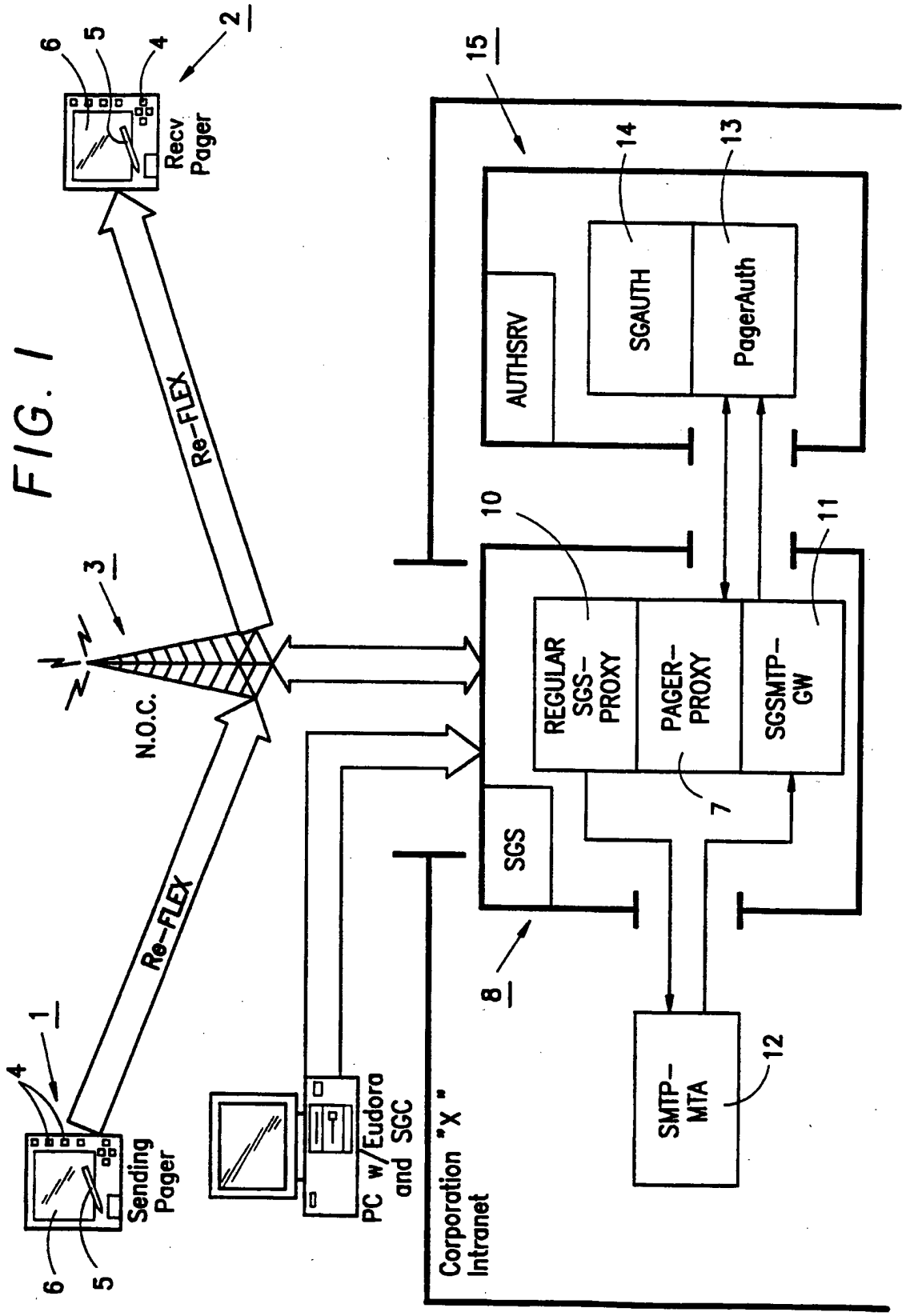
means for receiving an encrypted message transmitted via a wireless pager network from a pager proxy server;

means for decrypting an encrypted session key appended to the message;

means for decrypting the encrypted message transmitted from the pager proxy server using the decrypted session key; and

means for displaying the message.

FIG. 1



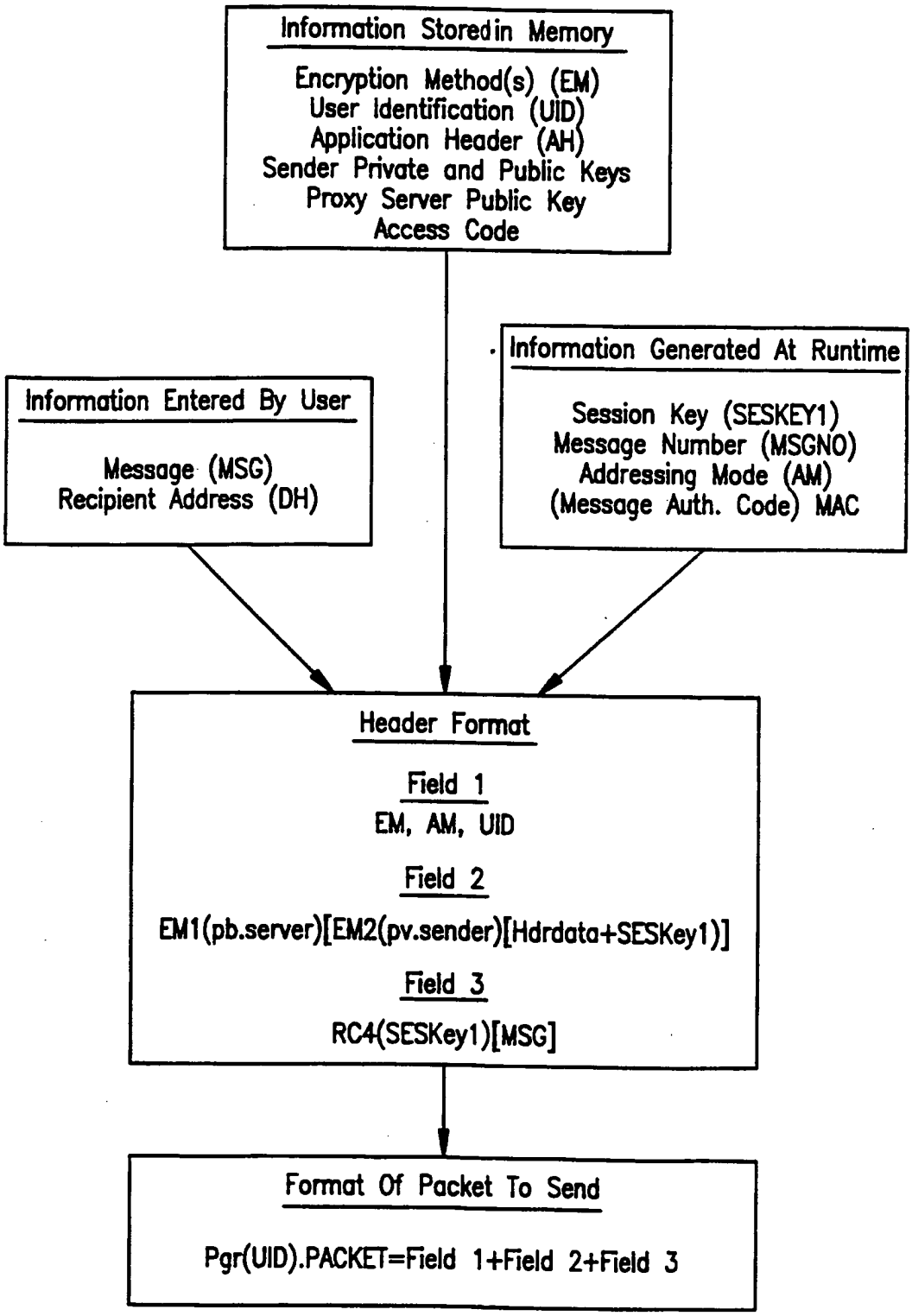
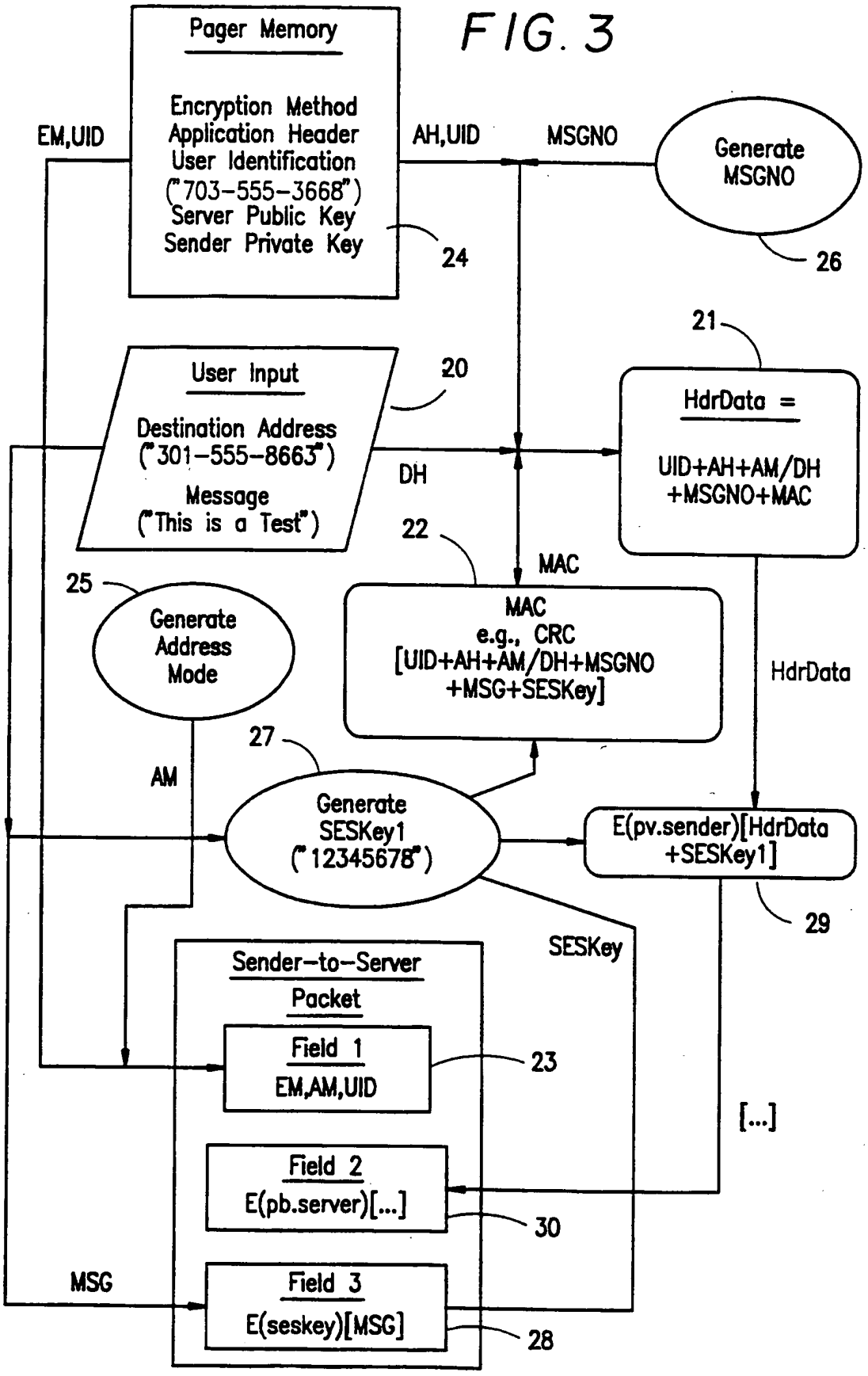


FIG. 2

FIG. 3



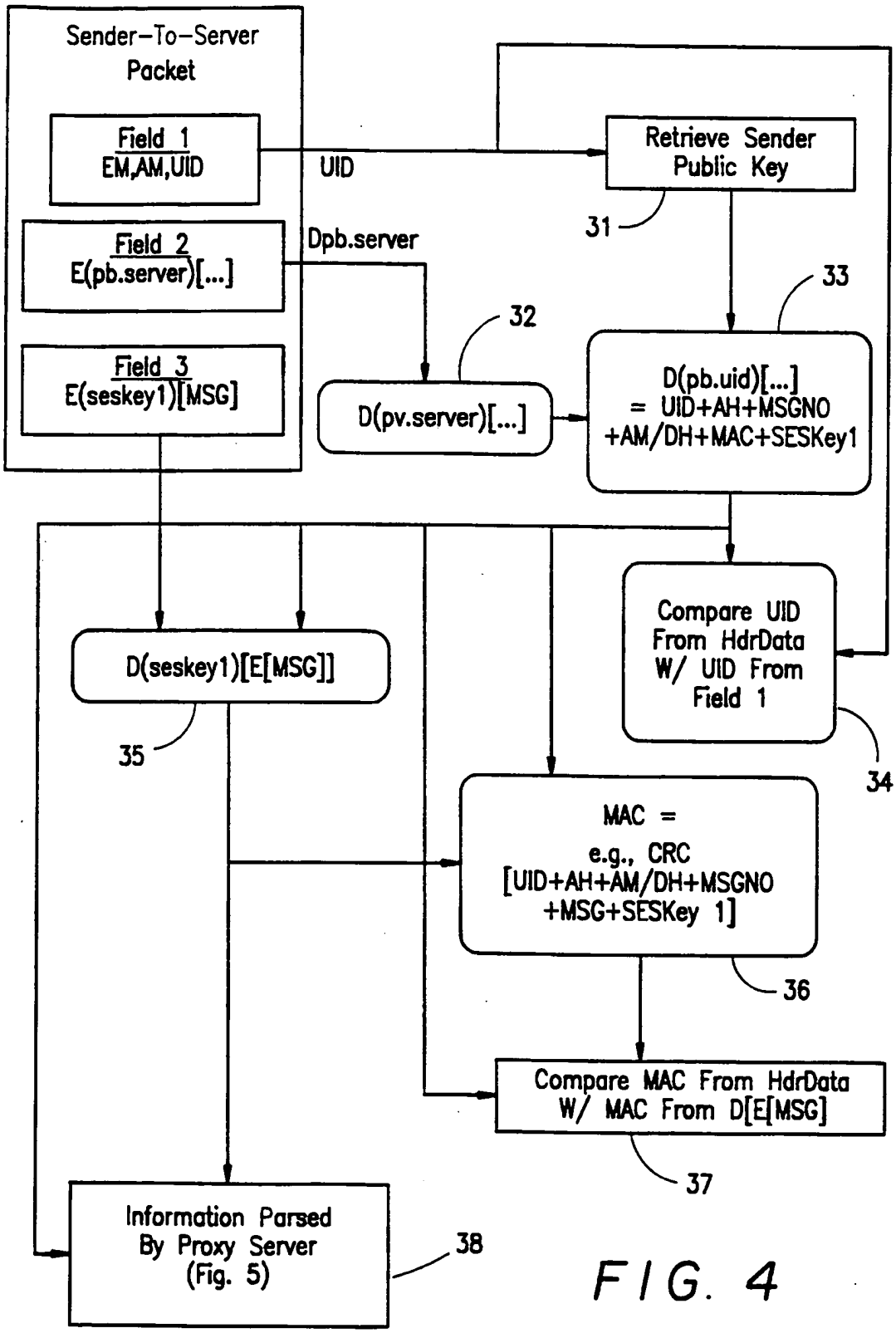


FIG. 4

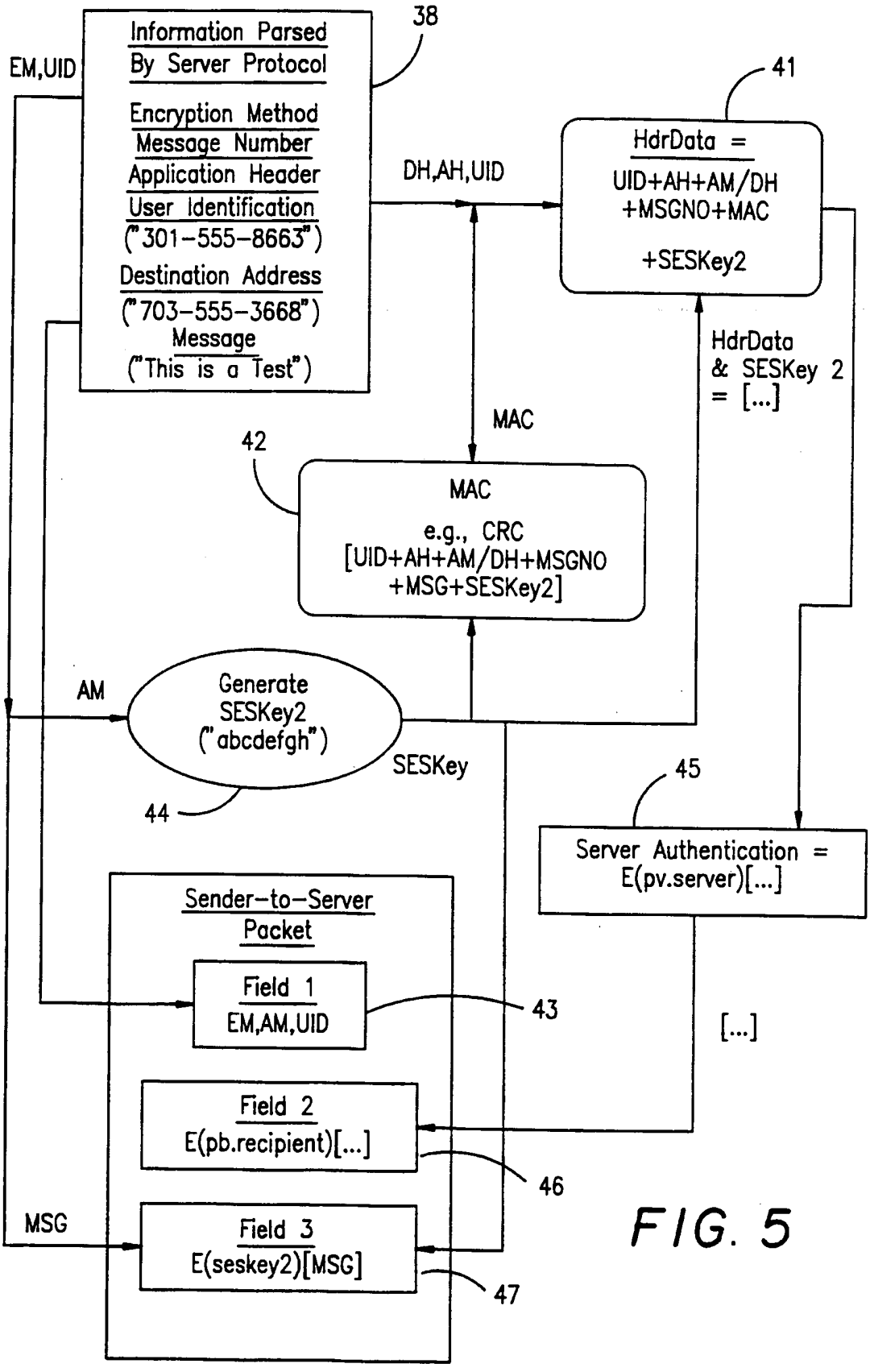


FIG. 5

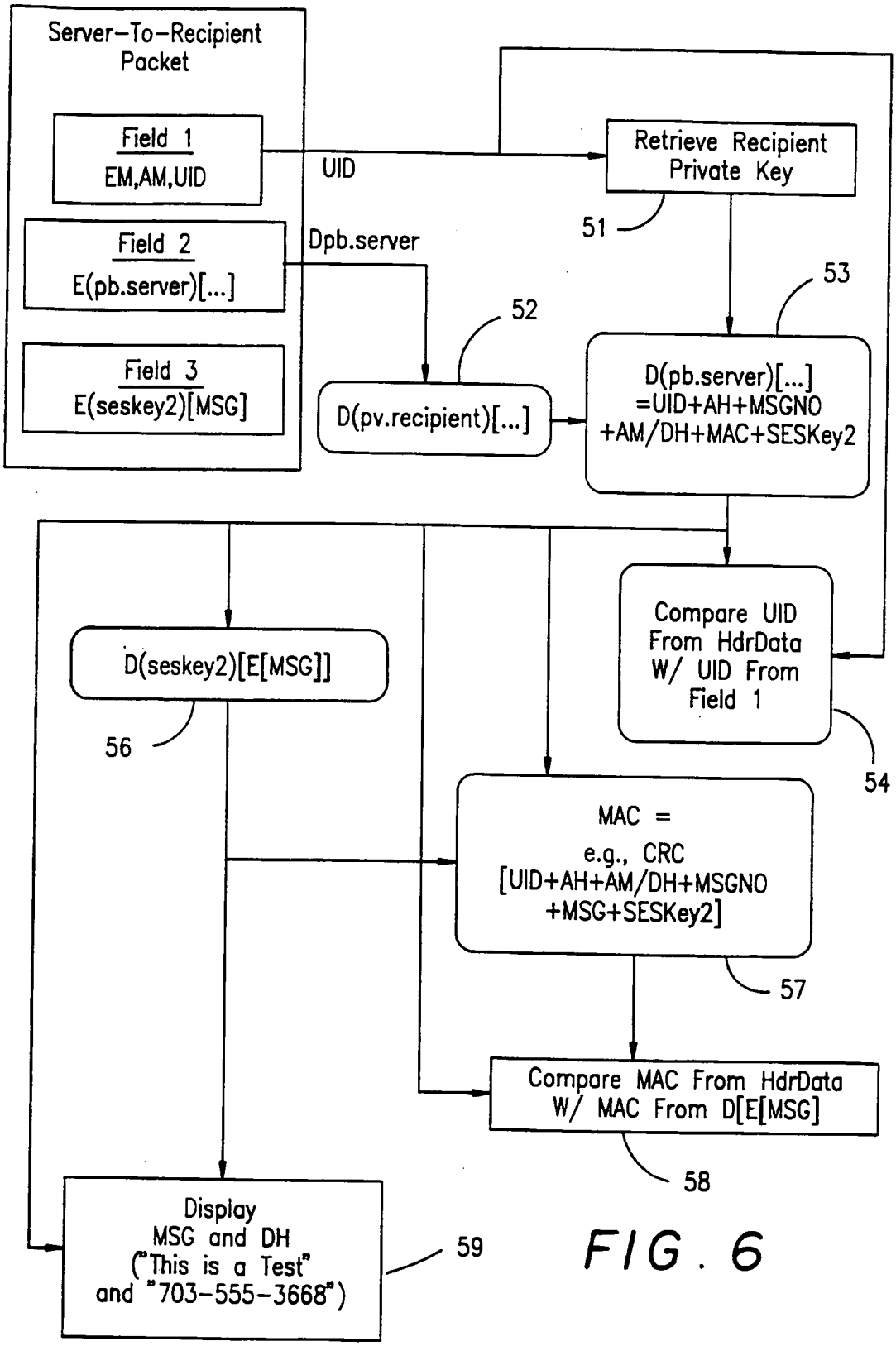


FIG. 6

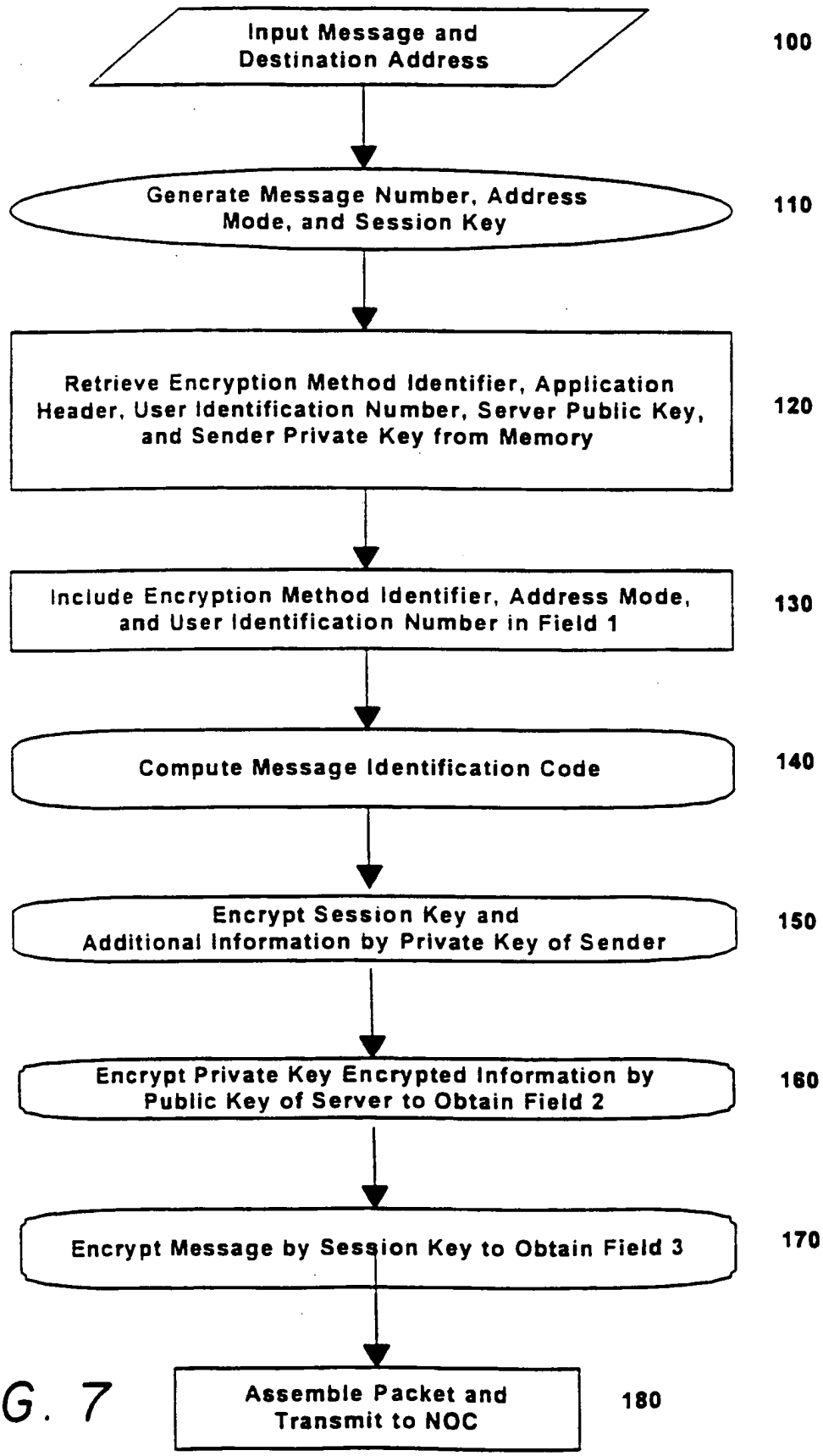
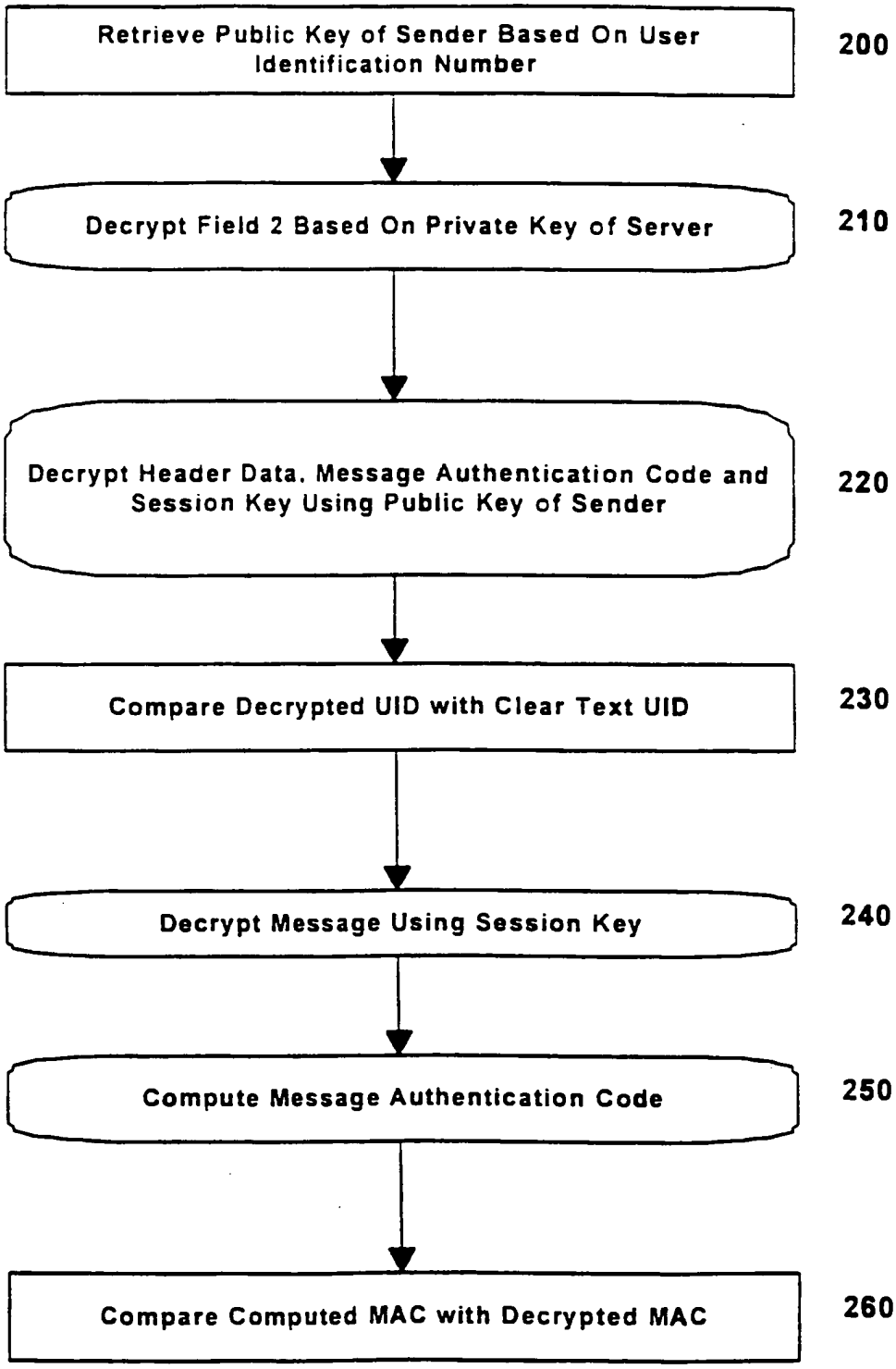


FIG. 7

FIG. 8



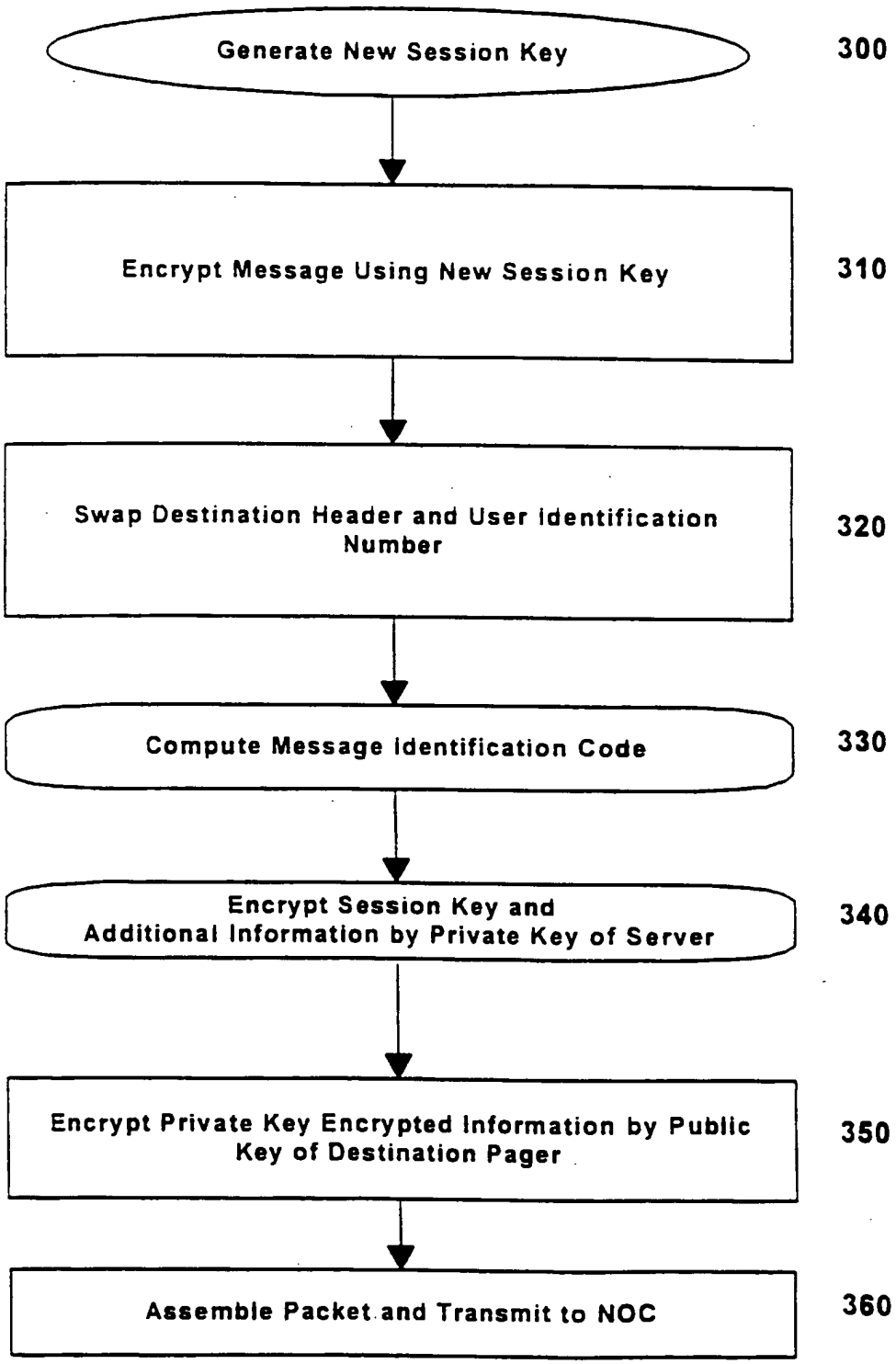
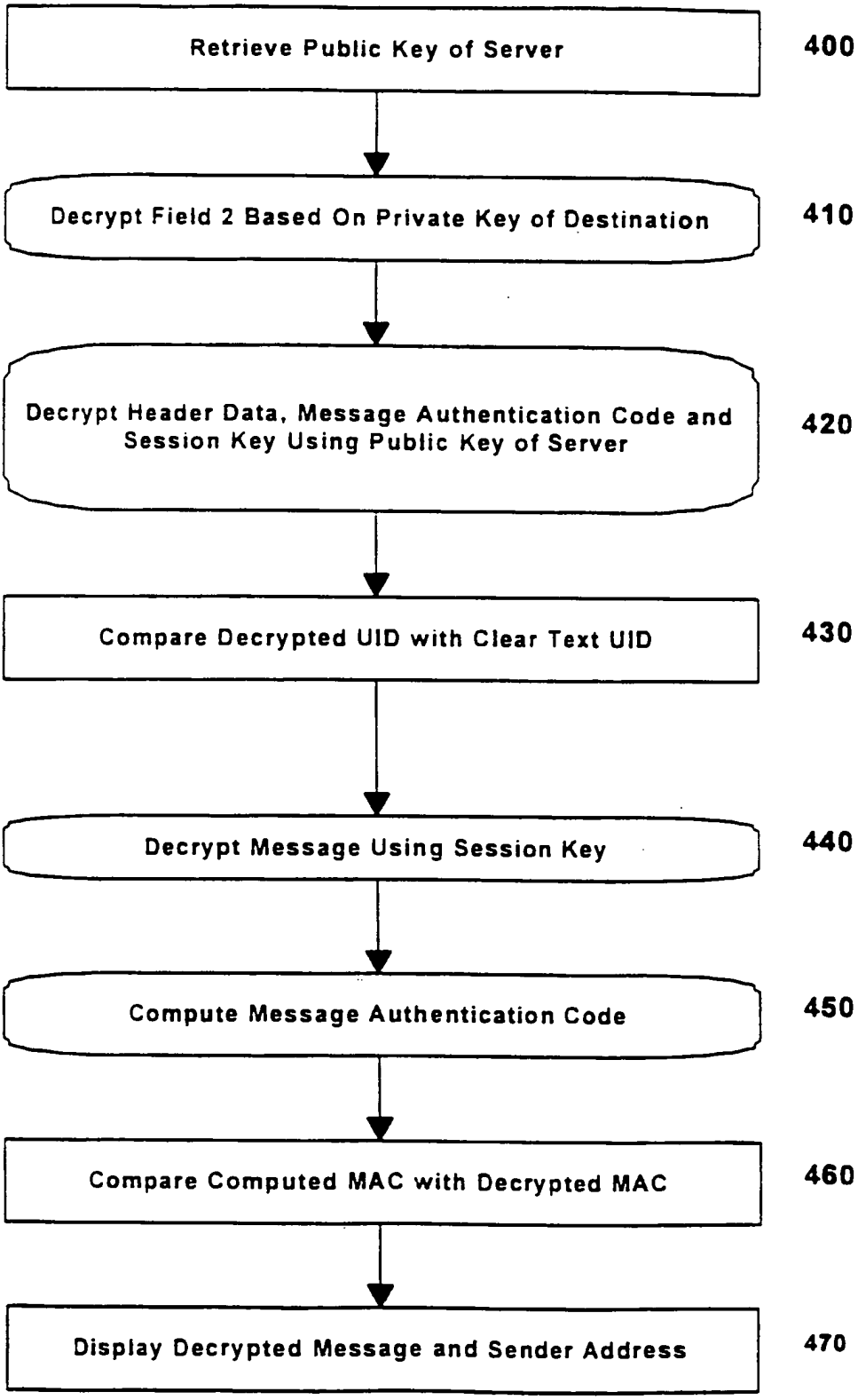


FIG. 9

FIG. 10



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/27531

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :H04L 9/08
US CL :380/21

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. : 380/21,44,45,49

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)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,285,496 A (FRANK et al) 08 February 1994 (08.02.94), column 2, lines 28-44, column 4, lines 12-68, column 6, lines 11-49.	1-31,36-56, 58,60-68,74-78
Y	US 5,604,801 A (DOLAN et al) 18 February 1997 (18.02.97), abstract, column 3, lines 2-38, 50-60, column 4, lines 19-24, 40-55.	1-31,36-56 58,60-68, 74-78
A	US 5,602,918 A (CHEN et al) 11 February 1997 (11.02.97), abstract, column 2, lines 36-41,57-60, column 4, lines 43-63.	3-4,6,17-18,27-28,31-32,40,42,53,65,68-70,77
A	US 5,452,356 A (ALBERT et al) 19 September 1995 (19.09.95), column 1, lines 60-68, column 2, lines 1-42, column 11, lines 15-55.	1-78

[X] 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 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
17 FEBRUARY 1999	06 MAY 1999

Name and mailing address of the ISA/US	Authorized officer
Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	GAIL HAYES
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9711

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/27531

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 5,495,533 A (LINEHAN et al) 27 February 1996 (27.02.96), column 9, lines 42-58, column 10, lines 22-32.	7,9,35,59,69-70,72-73

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US98/27531

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

APS
search terms: cypher, cipher, encode,encrypt, decrypt, key, keys, pager, wireless, proxy server, authenticate, authentication, transmission, transmitting, key management, public key, two-way communication, re-encrypt, messages, data, information

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04N 7/16, 7/167

A1

(11) International Publication Number: **WO 98/43426**

(43) International Publication Date: 1 October 1998 (01.10.98)

(21) International Application Number: PCT/EP97/02108

(22) International Filing Date: 25 April 1997 (25.04.97)

(30) Priority Data:
97400650.4 21 March 1997 (21.03.97) EP
(34) Countries for which the regional or international application was filed: FR et al.

(71) Applicant *(for all designated States except US)*: CANAL+ SOCIETE ANONYME [FR/FR]; 85/89, quai André Citroën, F–75711 Paris Cedex 15 (FR).

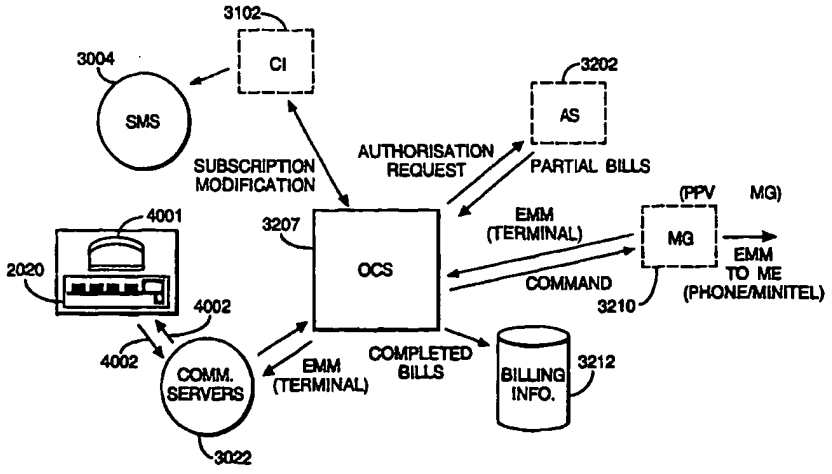
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: BAYASSI, Mulham [FR/FR]; 30, rue de Chambéry, F–75015 Paris (FR). DE LA TULLAYE, Pierre [FR/FR]; 7, allée Marcel Jouhandeau, F–92500 Rueil Malmaison (FR). JEZEQUEL, Jean–François [FR/FR]; 35, rue du Commandant Kieffer, F–95240 Cormeille en Parisis (FR).

(74) Agent: COZENS, Paul, Dennis; Mathys & Squire, 100 Grays Inn Road, London WC1X 8AL (GB).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, 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, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, 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.

(54) Title: BROADCAST AND RECEPTION SYSTEM, AND CONDITIONAL ACCESS SYSTEM THEREFOR



(57) Abstract

A digital satellite television system has a plurality of set–top–boxes associated with a plurality of end users' television receivers, a modem and a decoder housed in each STB, a Subscriber Authorization System (SAS) incorporating or having associated therewith a plurality of communication servers, means included in the SAS for generating Electronic Managements Messages (EMM), a back channel interconnecting each of the STBs individually with the SAS, means included in the SAS and each STB so that the necessary information required to inject a relevant EMM into the system is supplied directly to the relevant communication server included in or associated with the SAS to authorise the release of the said EMM and/or means to connect the modem to the back channel and means whereby an EMM is transmissible to the decoder directly from a relevant communication server included in or associated with the SAS. Further important features are also disclosed.

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		

BROADCAST AND RECEPTION SYSTEM, AND CONDITIONAL ACCESS SYSTEM THEREFOR

The present invention relates to a broadcast and reception system, in particular to a mass-market digital interactive satellite television system, and to a conditional access system therefor.

In particular, but not exclusively, the invention relates to a mass-market broadcast system having some or all of the following preferred features:-

- It is an information broadcast system, preferably a radio and/or television broadcast system
- It is a satellite system (although it could be applicable to cable or terrestrial transmission)
- It is a digital system, preferably using the MPEG, more preferably the MPEG-2, compression system for data/signal transmission
- It affords the possibility of interactivity.

More particularly the present invention relates to so-called pay television (or radio) where a user/viewer selects a programme/film/game to be viewed for which payment is to be made, this being referred to as a pay-per-view (PPV) or in the case of data to be downloaded a so-called pay-per-file (PPF).

With such known PPV or PPF systems a significant amount of time is required to be spent by the user/viewer in order to carry out the actions necessary to actually access the product being selected.

For example, in one known system the sequence of steps which have to be carried out are as follows:-

I) The user telephones a so-called Subscriber Management System (SMS) which in this known system includes a number of human operators which answer the subscriber's call and to whom the subscriber communicates the necessary information concerning the selected product and concerning the financial status of the subscriber

to a so-called Subscriber Authorization System (SAS) which has included in it or associated with it a plurality of communications servers.

ii) The operator at the SMS then has to check the financial status of the user before authorising the connection from the communications servers to the user's television set so that the product can be delivered and viewed by the user.

In another known system the human operator is replaced by an automatic voice server so that when the user telephones the SMS he/she hears a voice activated recording to which the user conveys the same information as I) above.

This second arrangement reduces the delay inherent in the first described arrangement which can be more easily overloaded when large numbers of users are wishing to order a product at the same time.

However, even with this second arrangement the user is involved in inputting significant information in the form of lengthy serial numbers which operation provides plenty of scope for error as well as being time consuming.

The third known arrangement involves the user making use of existing screen based systems such as MINITEL in France and PRESTEL in the United Kingdom, which systems replace the voice activated server referred to above in connection with the second arrangement. The MINITEL and PRESTEL systems themselves incorporate a modem at the consumer end.

In all these known arrangements the user is involved in the expenditure of significant time and effort in inputting all the information necessary to enable the system to in effect authorize the transmission of the chosen product to the user's television set.

In the case of a satellite television system there is a further delay involved in the user actually receiving the product selected.

In PPV and PPF systems the key element in controlling the user's access to products are so-called Entitlement Management Messages (EMM) which have to be injected into the system in order to give the user product access. More particularly the EMMs are the mechanism by which the encrypted data representative of a product is decrypted for a particular individual user.

In known satellite television systems the EMMs are transmitted to the user's televisions via the satellite link at regular intervals in the MPEG-2 data stream. Thus in the case of a particular user's EMM there can be a significant delay of perhaps several minutes before the user's next EMM transmission arrives at that user's television set.

This transmission delay is in addition to the delay referred to earlier which is inherent in the user having to manually input certain data into the system. The cumulative effect of these two delays is that it may take perhaps typically five minutes for a user to be able to gain access to the selected product.

The present invention is concerned with overcoming this problem.

In a first aspect, the present invention provides a conditional access system comprising:

means for generating a plurality of (preferably conditional access) messages; and

means for receiving the messages, said receiving means being adapted to communicate with said generating means via a communications server connected directly to said generating means.

Preferably, the message is an entitlement message for transmission (for example by broadcast) to the receiving means, said generating means being adapted to generate entitlement messages in response to data received from said receiving means.

The generating means may be arranged to transmit a message as a packet of digital

data to said receiving means either via said communications server or via a satellite transponder.

The receiving means may be connectable to said communications server via a modem and telephone link.

In a related aspect, the present invention provides a conditional access system for affording conditional access to subscribers, comprising:

a subscriber management system;

a subscriber authorization system coupled to the subscriber management system; and

a communications server; said server being connected directly to the subscriber authorization system.

The system may further comprise a receiver/decoder for the subscriber, the receiver/decoder being connectable to said communications server, and hence to said subscriber authorization system, via a modem and telephone link.

In a second aspect, the present invention provides a broadcast and reception system including a conditional access system as described above.

In a third aspect, the present invention provides a broadcast and reception system comprising:

means for generating a plurality of entitlement messages relating to broadcast programs;

means for receiving said messages from said generating means; and

means for connecting the receiving means to the generating means to receive said messages, said connecting means being capable of effecting a dedicated connection between the receiving means and the generating means.

The dedicated connection would typically be a hard-wired connection and/or a modemmed connection, with the possibility of the connection been made via a cellular

telephone system. In other words, the dedicated connection is capable of forming a channel of communication (from point to point). This is in contrast to broadcasting of information through the air or ambient medium. The connecting means would typically be a modem at the receiving means.

Hence, in a closely related aspect, the present invention provides a broadcast and reception system comprising:

means for generating a plurality of entitlement messages relating to broadcast programs;

means for receiving said messages from said generating means via a modem; and

means for connecting said modem to said generating means and said receiving means.

The above features can afford the advantage of providing the user the necessary viewing authorization (via the EMM) more quickly than has hitherto been possible, partly because, since the SAS typically uses a smaller amount of computer code than the SMS, the SAS can operate more efficiently (and in real time), partly because the SAS can itself, directly, generate the requisite EMM, and partly because the EMM can be passed to the user or subscriber via a dedicated (typically modemmed) link.

Preferably, the generating means is connected to said modem via a communications server which is preferably included in or associated with said generating means.

The receiving means may be further adapted to receive said entitlement messages via a satellite transponder.

The receiving means may be a receiver/decoder comprising means for receiving a compressed MPEG-type signal, means for decoding the received signal to provide a television signal and means for supplying the television signal to a television.

Preferably, the receiving means is adapted to communicate with said generating means

via said modem and connecting means. The receiving means may comprise means for reading a smartcard insertable thereinto by an end user, the smartcard having stored therein data to initiate automatically the transmission of a message from said receiving means to said generating means upon insertion of the smartcard by the end user.

In addition, the system may further comprise a voice link to enable the end user of the broadcast and reception system to communicate with the generating means.

It will be understood from the above that the present invention provides two arrangements by which the time it takes for an end user to access a desired product is reduced. Preferably both arrangements are employed to achieve the maximum time saving but either arrangement can be used individually.

According to a further aspect of the present invention, there is provided a broadcast and reception system, comprising, at the broadcast end:

a broadcast system including means for broadcasting a callback request;

and at the reception end:

a receiver including means for calling back the broadcast system in response to the callback request.

By providing that the broadcast system can request the receiver to call it back, the possibility is afforded of the broadcast system obtaining information from the receiver about the state of the receiver.

Preferably, the means for calling back the broadcast system includes a modem connectable to a telephone system. By using a modemmed back channel, a simple way of putting the invention into effect can be provided.

Preferably also, the means for calling back the broadcast system is arranged to transfer to the broadcast system information concerning the receiver. This information might include the number of remaining tokens, the number of pre-booked sessions, and so

on.

Preferably, the broadcast system includes means for storing the information, so that it can be processed at a later time, as desired.

Preferably, the broadcast means is arranged to broadcast a callback request which includes a command that the callback be made at a given time, and the means for calling back the broadcast system is arranged to respond to said command. By arranging for the callback to be later than the actual request, greater flexibility can be imparted to the system.

The broadcasting means may be arranged to broadcast as the callback request one or more Entitlement Messages for broadcast.

Preferably, the broadcast system includes means for generating a check message (such as a random number) and passing this to the receiver, the receiver includes means for encrypting the check message and passing this to the broadcast system, and the broadcast system further includes means for decrypting the check message received from the receiver and comparing this with the original check message. In this way it can be checked whether the receiver is genuine.

Any of the above features may be combined together in any appropriate combination. They may also be provided, as appropriate, in method aspects.

Preferred features of the present invention will now be described, purely by way of example, with reference to the accompanying drawings, in which:-

Figure 1 shows the overall architecture of a digital television system according to the preferred embodiment of the present invention;

Figure 2 shows the architecture of a conditional access system of the digital television system;

Figure 3 shows the structure of an Entitlement Management Message used in the conditional access system;

Figure 4 is a schematic diagram of the hardware of a Subscriber Authorisation System (SAS) according to a preferred embodiment of the present invention;

Figure 5 is a schematic diagram of the architecture of the SAS;

Figure 6 is a schematic diagram of a Subscriber Technical Management server forming part of the SAS;

Figure 7 is a flow diagram of the procedure for automatic renewal of subscriptions as implemented by the SAS;

Figure 8 is a schematic diagram of a group subscription bitmap used in the automatic renewal procedure;

Figure 9 shows the structure of an EMM used in the automatic renewal procedure;

Figure 10 shows in detail the structure of the EMM;

Figure 11 is a schematic diagram of an order centralized server when used to receive commands directly through communications servers;

Figure 12 illustrates diagrammatically a part of Figure 2 showing one embodiment of the present invention;

Figure 13 is a schematic diagram of the order centralized server when used to receive commands from the subscriber authorization system to request a callback;

Figure 14 is a schematic diagram of the communications servers;

Figure 15 shows the manner in which EMM emission cycle rate is varied according to the timing of a PPV event;

Figure 16 is a schematic diagram of a Message Emitter used to emit EMMs;

Figure 17 is a schematic diagram showing the manner of storage of EMMs within the Message Emitter;

Figure 18 is a schematic diagram of a smartcard;

Figure 19 is a schematic diagram of an arrangement of zones in the memory of the smartcard; and

Figure 20 is a schematic diagram of a PPV event description.

An overview of a digital television broadcast and reception system 1000 according to the present invention is shown in Figure 1. The invention includes a mostly conventional digital television system 2000 which uses the known MPEG-2 compression system to transmit compressed digital signals. In more detail, MPEG-2 compressor 2002 in a broadcast centre receives a digital signal stream (typically a stream of video signals). The compressor 2002 is connected to a multiplexer and scrambler 2004 by linkage 2006. The multiplexer 2004 receives a plurality of further input signals, assembles one or more transport streams and transmits compressed digital signals to a transmitter 2008 of the broadcast centre via linkage 2010, which can of course take a wide variety of forms including telecom links. The transmitter 2008 transmits electromagnetic signals via uplink 2012 towards a satellite transponder 2014, where they are electronically processed and broadcast via notional downlink 2016 to earth receiver 2018, conventionally in the form of a dish owned or rented by the end user. The signals received by receiver 2018 are transmitted to an integrated receiver/decoder 2020 owned or rented by the end user and connected to the end user's television set 2022. The receiver/decoder 2020 decodes the compressed MPEG-2 signal into a television signal for the television set 2022.

A conditional access system 3000 is connected to the multiplexer 2004 and the receiver/decoder 2020, and is located partly in the broadcast centre and partly in the decoder. It enables the end user to access digital television broadcasts from one or more broadcast suppliers. A smartcard, capable of decrypting messages relating to commercial offers (that is, one or several television programmes sold by the broadcast supplier), can be inserted into the receiver/decoder 2020. Using the decoder 2020 and smartcard, the end user may purchase events in either a subscription mode or a pay-per-view mode.

An interactive system 4000, also connected to the multiplexer 2004 and the receiver/decoder 2020 and again located partly in the broadcast centre and partly in the decoder, enables the end user to interact with various applications via a modemmed back channel 4002.

The conditional access system 3000 is now described in more detail.

With reference to Figure 2, in overview the conditional access system 3000 includes a Subscriber Authorization System (SAS) 3002. The SAS 3002 is connected to one or more Subscriber Management Systems (SMS) 3004, one SMS for each broadcast supplier, by a respective TCP-IP linkage 3006 (although other types of linkage could alternatively be used). Alternatively, one SMS could be shared between two broadcast suppliers, or one supplier could use two SMSs, and so on.

First encrypting units in the form of ciphering units 3008 utilising "mother" smartcards 3010 are connected to the SAS by linkage 3012. Second encrypting units again in the form of ciphering units 3014 utilising mother smartcards 3016 are connected to the multiplexer 2004 by linkage 3018. The receiver/decoder 2020 receives a "daughter" smartcard 3020. It is connected directly to the SAS 3002 by Communications Servers 3022 via the modemmed back channel 4002. The SAS sends amongst other things subscription rights to the daughter smartcard on request.

The smartcards contain the secrets of one or more commercial operators. The

"mother" smartcard encrypts different kinds of messages and the "daughter" smartcards decrypt the messages, if they have the rights to do so.

The first and second ciphering units 3008 and 3014 comprise a rack, an electronic VME card with software stored on an EEPROM, up to 20 electronic cards and one smartcard 3010 and 3016 respectively, for each electronic card, one (card 3016) for encrypting the ECMs and one (card 3010) for encrypting the EMMs.

The operation of the conditional access system 3000 of the digital television system will now be described in more detail with reference to the various components of the television system 2000 and the conditional access system 3000.

Multiplexer and Scrambler

With reference to Figures 1 and 2, in the broadcast centre, the digital video signal is first compressed (or bit rate reduced), using the MPEG-2 compressor 2002. This compressed signal is then transmitted to the multiplexer and scrambler 2004 via the linkage 2006 in order to be multiplexed with other data, such as other compressed data.

The scrambler generates a control word used in the scrambling process and included in the MPEG-2 stream in the multiplexer 2004. The control word is generated internally and enables the end user's integrated receiver/decoder 2020 to descramble the programme.

Access criteria, indicating how the programme is commercialised, are also added to the MPEG-2 stream. The programme may be commercialised in either one of a number of "subscription" modes and/or one of a number of "Pay Per View" (PPV) modes or events. In the subscription mode, the end user subscribes to one or more commercial offers, or "bouquets", thus getting the rights to watch every channel inside those bouquets. In the preferred embodiment, up to 960 commercial offers may be selected from a bouquet of channels. In the Pay Per View mode, the end user is provided with the capability to purchase events as he wishes. This can be achieved

by either pre-booking the event in advance ("pre-book mode"), or by purchasing the event as soon as it is broadcast ("impulse mode"). In the preferred embodiment, all users are subscribers, whether or not they watch in subscription or PPV mode, but of course PPV viewers need not necessarily be subscribers.

Both the control word and the access criteria are used to build an Entitlement Control Message (ECM); this is a message sent in relation with one scrambled program; the message contains a control word (which allows for the descrambling of the program) and the access criteria of the broadcast program. The access criteria and control word are transmitted to the second encrypting unit 3014 via the linkage 3018. In this unit, an ECM is generated, encrypted and transmitted on to the multiplexer and scrambler 2004.

Each service broadcast by a broadcast supplier in a data stream comprises a number of distinct components; for example a television programme includes a video component, an audio component, a sub-title component and so on. Each of these components of a service is individually scrambled and encrypted for subsequent broadcast to the transponder 2014. In respect of each scrambled component of the service, a separate ECM is required.

Programme Transmission

The multiplexer 2004 receives electrical signals comprising encrypted EMMs from the SAS 3002, encrypted ECMs from the second encrypting unit 3014 and compressed programmes from the compressor 2002. The multiplexer 2004 scrambles the programmes and transmits the scrambled programmes, the encrypted EMMs and the encrypted ECMs as electric signals to a transmitter 2008 of the broadcast centre via linkage 2010. The transmitter 2008 transmits electromagnetic signals towards the satellite transponder 2014 via uplink 2012.

Programme Reception

The satellite transponder 2014 receives and processes the electromagnetic signals transmitted by the transmitter 2008 and transmits the signals on to the earth receiver

2018, conventionally in the form of a dish owned or rented by the end user, via downlink 2016. The signals received by receiver 2018 are transmitted to the integrated receiver/decoder 2020 owned or rented by the end user and connected to the end user's television set 2022. The receiver/decoder 2020 demultiplexes the signals to obtain scrambled programmes with encrypted EMMs and encrypted ECMs.

If the programme is not scrambled, that is, no ECM has been transmitted with the MPEG-2 stream, the receiver/decoder 2020 decompresses the data and transforms the signal into a video signal for transmission to television set 2022.

If the programme is scrambled, the receiver/decoder 2020 extracts the corresponding ECM from the MPEG-2 stream and passes the ECM to the "daughter" smartcard 3020 of the end user. This slots into a housing in the receiver/decoder 2020. The daughter smartcard 3020 controls whether the end user has the right to decrypt the ECM and to access the programme. If not, a negative status is passed to the receiver/decoder 2020 to indicate that the programme cannot be descrambled. If the end user does have the rights, the ECM is decrypted and the control word extracted. The decoder 2020 can then descramble the programme using this control word. The MPEG-2 stream is decompressed and translated into a video signal for onward transmission to television set 2022.

Subscriber Management System (SMS)

A Subscriber Management System (SMS) 3004 includes a database 3024 which manages, amongst others, all of the end user files, commercial offers (such as tariffs and promotions), subscriptions, PPV details, and data regarding end user consumption and authorization. The SMS may be physically remote from the SAS.

Each SMS 3004 transmits messages to the SAS 3002 via respective linkage 3006 which imply modifications to or creations of Entitlement Management Messages (EMMs) to be transmitted to end users.

The SMS 3004 also transmits messages to the SAS 3002 which imply no

modifications or creations of EMMs but imply only a change in an end user's state (relating to the authorization granted to the end user when ordering products or to the amount that the end user will be charged).

As described later, the SAS 3002 sends messages (typically requesting information such as call-back information or billing information) to the SMS 3004, so that it will be apparent that communication between the two is two-way.

Entitlement Management Messages (EMMs)

The EMM is a message dedicated to an individual end user (subscriber), or a group of end users, only (in contrast with an ECM, which is dedicated to one scrambled programme only or a set of scrambled programmes if part of the same commercial offer). Each group may contain a given number of end users. This organisation as a group aims at optimising the bandwidth; that is, access to one group can permit the reaching of a great number of end users.

Various specific types of EMM are used in putting the present invention into practice. Individual EMMs are dedicated to individual subscribers, and are typically used in the provision of Pay Per View services; these contain the group identifier and the position of the subscriber in that group. So-called "Group" subscription EMMs are dedicated to groups of, say, 256 individual users, and are typically used in the administration of some subscription services. This EMM has a group identifier and a subscribers' group bitmap. Audience EMMs are dedicated to entire audiences, and might for example be used by a particular operator to provide certain free services. An "audience" is the totality of subscribers having smartcards which bear the same Operator Identifier (OPI). Finally, a "unique" EMM is addressed to the unique identifier of the smartcard.

The structure of a typical EMM is now described with reference to Figure 3. Basically, the EMM, which is implemented as a series of digital data bits, comprises a header 3060, the EMM proper 3062, and a signature 3064. The header 3060 in turn comprises a type identifier 3066 to identify whether the type is individual, group,

audience or some other type, a length identifier 3068 which gives the length of the EMM, an optional address 3070 for the EMM, an operator identifier 3072 and a key identifier 3074. The EMM proper 3062 of course varies greatly according to its type. Finally, the signature 3064, which is typically of 8 bytes long, provides a number of checks against corruption of the remaining data in the EMM.

Subscriber Authorization System (SAS)

The messages generated by the SMS 3004 are passed via linkage 3006 to the Subscriber Authorization System (SAS) 3002, which in turn generates messages acknowledging receipt of the messages generated by the SMS 3004 and passes these acknowledgements to the SMS 3004.

As shown in Figure 4, at the hardware level the SAS comprises in known fashion a mainframe computer 3050 (in the preferred embodiment a DEC machine) connected to one or more keyboards 3052 for data and command input, one or more Visual Display Units (VDUs) 3054 for display of output information and data storage means 3056. Some redundancy in hardware may be provided.

At the software level the SAS runs, in the preferred embodiment on a standard open VMS operating system, a suite of software whose architecture is now described in overview with reference to Figure 5; it will be understood that the software could alteratively be implemented in hardware.

In overview the SAS comprises a Subscription Chain area 3100 to give rights for subscription mode and to renew the rights automatically each month, a Pay Per View Chain area 3200 to give rights for PPV events, and an EMM Injector 3300 for passing EMMs created by the Subscription and PPV chain areas to the multiplexer and scrambler 2004, and hence to feed the MPEG stream with EMMs. If other rights are to be granted, such as Pay Per File (PPF) rights in the case of downloading computer software to a user's Personal Computer, other similar areas are also provided.

One function of the SAS 3002 is to manage the access rights to television

programmes, available as commercial offers in subscription mode or sold as PPV events according to different modes of commercialisation (pre-book mode, impulse mode). The SAS 3002, according to those rights and to information received from the SMS 3004, generates EMMs for the subscriber.

The Subscription Chain area 3100 comprises a Command Interface (CI) 3102, a Subscriber Technical Management (STM) server 3104, a Message Generator (MG) 3106, and the Ciphering Unit 3008.

The PPV Chain area 3200 comprises an Authorisation Server (AS) 3202, a relational database 3204 for storing relevant details of the end users, a local blacklist database 3205, Database Servers 3206 for the database, an Order Centralized Server (OCS) 3207, a Server for Programme Broadcaster (SPB) 3208, a Message Generator (MG) 3210 whose function is basically the same as that for the Subscription Chain area and is hence not described further in any detail, and the Ciphering Unit 3008.

The EMM Injector 3300 comprises a plurality of Message Emitters (MEs) 3302, 3304, 3306 and 3308 and Software Multiplexers (SMUXs) 3310 and 3312. In the preferred embodiment, there are two MEs, 3302 and 3304 for the Message Generator 3106, with the other two MEs 3306 and 3308 for the Message Generator 3210. MEs 3302 and 3306 are connected to the SMUX 3310 whilst MEs 3304 and 3308 are connected to the SMUX 3312.

Each of the three main components of the SAS (the Subscription Chain area, the PPV Chain area and the EMM Injector) are now considered in more detail.

Subscription Chain Area

Considering first the Subscription Chain area 3100, the Command Interface 3102 is primarily for despatching messages from the SMS 3004 to the STM server 3104, as well as to the OCS 3206, and from the OCS to the SMS. The Command Interface takes as input from the SMS either direct commands or batch files containing commands. It performs syntactic analysis on the messages coming from the STM

server, and is able to emit accurate messages when an error occurs in a message (parameter out of range, missing parameter, and so on). It traces incoming commands in textual form in a trace file 3110 and also in binary form in a replay file 3112 in order to be able to replay a series of commands. Traces can be disabled and the size of files limited.

Detailed discussion of the STM server 3104 is now provided with particular reference to Figure 6. The STM server is effectively the main engine of the Subscription Chain area, and has the purpose of managing free rights, the creation of new subscribers and the renewal of existing subscribers. As shown in the figure, commands are passed on to the Message Generator 3106, albeit in a different format from that in which the commands are passed to the STM server. For each command, the STM server is arranged to send an acknowledgement message to the CI only when the relevant command has been successfully processed and sent to the MG.

The STM server includes a subscriber database 3120, in which all the relevant parameters of the subscribers are stored (smartcard number, commercial offers, state, group and position in the group, and so on). The database performs semantic checks of the commands sent by the CI 3102 against the content of the database, and updates the database when the commands are valid.

The STM server further manages a First In First Out (FIFO) buffer 3122 between the STM server and the MG, as well as a backup disk FIFO 3124. The purpose of the FIFOs is to average the flow of commands from the CI if the MG is not able to respond for a while for any reason. They can also ensure that in the case of a crash of the STM server or MG no command will be lost, since the STM server is arranged to empty (that is, send to the MG) its FIFOs when restarted. The FIFOs are implemented as files.

The STM server includes at its core an automatic renewal server 3126 which automatically generates renewals, and, if required by the operators, free rights. In this context, the generation of renewals may be thought of as including the generation of

rights for the first time, although it will be understood that the generation of new rights is initiated at the SMS. As will become apparent, the two can be treated by roughly the same commands and EMMs.

Having the STM separate from the SAS, and the automatic renewal server within the SAS rather than (in known systems) in the SMS 3004, is a particularly important feature, since it can significantly reduce the number of commands which need to be passed from the SMS to the SAS (bearing in mind that the SMS and SAS may be in different locations and operated by different operators). In fact, the two main commands required from the SMS are merely commands that a new subscription should be started and that an existing subscription should be stopped (for example in the case of non-payment). By minimising command exchange between the SMS and SAS, the possibility of failure of command transfer in the linkage 3006 between the two is reduced; also, the design of the SMS does not need to take into account the features of the conditional access system 3000 generally.

Automatic renewal proceeds in the fashion indicated in the flow diagram of Figure 7. In order to reduce bandwidth, and given that a very high percentage of all renewals are standard, renewal proceeds in groups of subscribers; in the preferred embodiments there are 256 individual subscribers per group. The flow diagram begins with the start step 3130, and proceeds to step 3132 where a monthly activation of the renewal function is made (although of course it will be appreciated that other frequencies are also possible). With a monthly frequency, rights are given to the end user for the current month and all of the following month, at which point they expire if not renewed.

In step 3134 the subscriber database 3120 is accessed in respect of each group and each individual within that group to determine whether rights for the particular individual are to be renewed.

In step 3136, a group subscription bitmap is set up according to the contents of the subscriber database, as shown in Figure 8. The bitmap comprises a group identifier

(in this case Group 1 – "G1") 3138 and 256 individual subscriber zones 3140. The individual bits in the bitmap are set to 1 or zero according to whether or not the particular subscriber is to have his rights renewed. A typical set of binary data is shown in the figure.

In step 3142 the appropriate commands, including the group subscription bitmap, are passed to the Message Generator 3106. In step 3143 the Message Generator sets an obsolescence date to indicate to the smartcard the date beyond which the particular subscription EMM is not valid; typically this date is set as the end of the next month.

In step 3144 the Message Generator generates from the commands appropriate group subscription EMMs and asks the Ciphering Unit 3008 to cipher the EMMs, the ciphered EMMs being then passed to the EMM Injector 3300, which, in step 3146, injects the EMMs into the MPEG–2 data stream.

Step 3148 indicates that the above described procedure is repeated for each and every group. The process is finally brought to an end at stop step 3150.

The flow diagram described above with reference to Figure 7 relates in fact specifically to the renewal of subscriptions. The STM also manages in a similar way free audience rights and new subscribers.

In the case of free audience rights, available for specific television programmes or groups of such programmes, these are made available by the STM issuing a command to the Message Generator to generate appropriate audience EMMs (for a whole audience) with an obsolescence date a given number of days (or weeks) hence. The MG computes the precise obsolescence date based on the STM command.

In the case of new subscribers, these are dealt with in two stages. Firstly, on purchase the smartcard in the receiver/decoder 2020 (if desired by the operator) affords the subscriber free rights for a given period (typically a few days). This is achieved by generating a bitmap for the subscriber which includes the relevant obsolescence date.

The subscriber then passes his completed paperwork to the operator managing the subscriber (at the SMS). Once the paperwork has been processed, the SMS supplies to the SAS a start command for that particular subscriber. On receipt by the SAS of the start command, the STM commands the MG to assign a unique address to the new subscriber (with a particular group number and position within the group) and to generate a special, so-called "commercial offer" subscription EMM (as opposed to the more usual "group" subscription EMM used for renewals) to provide the particular subscriber with rights until the end of the next month. From this point renewal of the subscriber can occur automatically as described above. By this two stage process it is possible to grant new subscribers rights until the SMS issues a stop command.

It is to be noted that the commercial offer subscription EMM is used for new subscribers and for reactivation of existing subscribers. The group subscription EMM is used for renewal and suspension purposes.

With reference to Figure 9, a typical subscription EMM proper (that is, ignoring the header and signature) generated by the above procedure comprises the following main portions, namely typically a 256 bit subscription (or subscribers' group) bitmap 3152, 128 bits of management ciphering keys 3154 for the ciphering of the EMM, 64 bits of each exploitation ciphering key 3156 to enable the smartcard 3020 to decipher a control word to provide access to broadcast programmes, and 16 bits of obsolescence date 3158 to indicate the date beyond which the smartcard will ignore the EMM. In fact in the preferred embodiment three exploitation keys are provided, one set for the present month, one set for the next month, and one for resume purposes in the event of system failure.

In more detail, the group subscription EMM proper has all of the above components, except the management ciphering keys 3154. The commercial offer subscription EMM proper (which is for an individual subscriber) includes instead of the full subscribers' group bitmap 3152 the group ID followed by the position in the group, and then management ciphering keys 3154 and three exploitation keys 3156, followed by the relevant obsolescence date 3158.

The Message Generator 3106 serves to transform commands issued by the STM server 3104 into EMMs for passing to the Message Emitter 3302. With reference to Figure 5, firstly, the MG produces the EMMs proper and passes them to the Ciphering Unit 3008 for ciphering with respect to the management and exploitation keys. The CU completes the signature 3064 on the EMM (see Figure 3) and passes the EMM back to the MG, where the header 3060 is added. The EMMs which are passed to the Message Emitter are thus complete EMMs. The Message Generator also determines the broadcast start and stop time and the rate of emission of the EMMs, and passes these as appropriate directions along with the EMMs to the Message Emitter. The MG only generates a given EMM once; it is the ME which performs its cyclic transmission.

Again with reference to Figure 5, the Message Generator includes its own EMM database 3160 which, for the lifetime of the relevant EMM, stores it. It is erased once its emission duration has expired. The database is used to ensure consistency between the MG and ME, so that for example when an end user is suspended the ME will not continue to send renewals. In this regard the MG computes the relevant operations and sends them to the ME.

On generation of an EMM, the MG assigns a unique identifier to the EMM. When the MG passes the EMM to the ME, it also passes the EMM ID. This enables identification of a particular EMM at both the MG and the ME.

Also concerning the Subscription Chain area, the Message Generator includes two FIFOs 3162 and 3164, one for each of the relevant Message Emitters 3302 and 3304 in the EMM Injector 3300, for storing the ciphered EMMs. Since the Subscription Chain area and EMM Injector may be a significant distance apart, the use of FIFOs can allow full continuity in EMM transmission even if the links 3166 and 3168 between the two fail. Similar FIFO's are provided in the Pay Per View Chain area.

One particular feature of the Message Generator in particular and the conditional access system in general concerns the way that it reduces the length of the EMM

proper 3062 by mixing parameter length and identifier to save space. This is now described with reference to Figure 10 which illustrates an exemplary EMM (in fact a PPV EMM, which is the simplest EMM). The reduction in length occurs in the Pid (Packet or "Parameter" identifier) 3170. This comprises two portions, the actual ID 3172, and the length parameter for the packet 3174 (necessary in order that the start of the next packet can be identified). The whole Pid is expressed in just one byte of information, 4 bits being reserved for the ID, and four for the length. Because 4 bits is not sufficient to define the length in true binary fashion, a different correspondence between the bits and the actual length is used, this correspondence being represented in a look-up table, stored in storage area 3178 in the Message Generator (see Figure 5). The correspondence is typically as follows:-

0000	=	0
0001	=	1
0010	=	2
0011	=	3
0100	=	4
0101	=	5
0110	=	6
0111	=	7
1000	=	8
1001	=	9
1010	=	10
1011	=	11
1100	=	12
1101	=	16
1110	=	24
1111	=	32

It will be seen that the length parameter is not directly proportional to the actual length of the packet; the relationship is in part more quadratic rather than linear. This allows for a greater range of packet length.

Pay Per View Chain Area

Concerning the Pay Per View Chain area 3200, with reference to Figure 5 in more detail the Authorisation Server 3202 has as its client the Order Centralized Server 3207, which requests information about each subscriber which connects to the Communications Servers 3022 to purchase a PPV product.

If the subscriber is known from the AS 3202, a set of transactions takes place. If the subscriber is authorized for the order, the AS creates a bill and sends it to the OCS. Otherwise, it signals to the OCS that the order is not authorized.

It is only at the end of this set of transactions that the AS updates the end users database 3204 via the database servers (DBAS) 3206, if at least one transaction was authorized; this optimizes the number of database accesses.

The criteria according to which the AS authorizes purchase are stored in the database, accessed through DBAS processes. In one embodiment, the database is the same as the database accessed by the STM.

Depending on consumer profile, the authorization may be denied (PPV_Forbidden,Casino_Forbidden ...). These kind of criteria are updated by STM 3104, on behalf of the SMS 3004.

Other parameters are checked, such as limits allowed for purchase (either by credit card, automatic payment, or number of authorized token purchases per day).

In case of payment with a credit card, the number of the card is checked against a local blacklist stored in the local blacklist database 3205.

When all the verifications are successful, the AS:-

1. Generates a bill and sends it to the OCS, which completes this bill and stores it in a file, this file being later sent to the SMS for processing (customer actual billing); and

2. Updates the database, mainly to set new purchase limits.

This check-and-generate-bill-if-OK mechanism applies for each command a subscriber may request during a single connection (it is possible to order e.g. 5 movies in a single session).

It is to be noted that the AS has a reduced amount of information concerning the subscriber, by comparison with that held by the SMS. For example, the AS does not hold the name or address of the subscriber. On the other hand, the AS does hold the smartcard number of the subscriber, the subscriber's consumer category (so that different offers can be made to different subscribers), and various flags which state whether, for example, the subscriber may purchase on credit, or he is suspended or his smartcard has been stolen. Use of a reduced amount of information can help to reduce the amount of time taken to authorize a particular subscriber request.

The main purpose of the DBASs 3206 is to increase database performance seen from the AS, by paralleling the accesses (so actually it does not make much sense to define a configuration with only one DBAS). An AS parameter determines how many DBASes should connect. A given DBAS may be connected to only one AS.

The OCS 2307 mainly deals with PPV commands. It operates in several modes.

Firstly, it operates to process commands issued by the SMS, such as product refreshment (for instance, if the bill is already stored by the SMS, no bill is generated by the OCS), update of the wallet in the smartcard 3020, and session cancellation/update.

The various steps in the procedure are:-

1. Identifying the relevant subscriber (using the AS 3202);

2. If valid, generate adequate commands to the Message Generator, in order to send an appropriate EMM. Commands may be:

Product commands,

Update of the wallet,

Session erasure.

Note that these operations do not imply creation of billing information, since billing is already known from the SMS. These operations are assimilated to "free products" purchase.

Secondly, the OCS deals with commands received from the subscribers through the Communications Servers 3022. These may be received either via a modem connected to the receiver/decoder 2020, or by voice activation via the telephone 4001, or by key activation via a MINITEL, PRESTEL or like system where available.

Thirdly, the OCS deals with callback requests issued by the SMS. These last two modes of operation are now discussed in more detail.

In the second type of mode described above it was stated that the OCS deals with commands received directly from the end user (subscriber) through the Communications Servers 3022. These include product orders (such as for a particular PPV event), a subscription modification requested by the subscriber, and a reset of a parental code (a parental code being a code by which parents may restrict the right of access to certain programmes or classes of programmes).

The way in which these commands are dealt with is now described in more detail with reference to Figure 11.

Product orders by a subscriber involve the following steps:

1. Identifying through the AS the caller who is making a call through the CS 3022 ordering a particular product;
2. Checking the caller's request validity, again using the AS (where the order is placed using the receiver/decoder 2020, this is achieved by verifying the smartcard 3020 details);
3. Ascertain the price of the purchase;
4. Check that the price does not exceed the caller's credit limit etc;
5. Receiving a partial bill from the AS;

6. Filling additional fields in the bill to form a completed bill;
7. Adding the completed bill to a billing information storage file 3212 for later processing; and
8. Sending corresponding command(s) to the PPV Message Generator 3210 to generate the relevant EMM(s).

The EMM(s) is sent either on the modem line 4002 if the consumer placed the product order using the receiver/decoder 2020 (more details of this are described later), or else it is broadcast. The one exception to this is where there is some failure of the modem connection (in the case where the consumer places the order using the receiver/decoder); in this event the EMM is broadcast over the air.

A subscription modification requested by a subscriber involves:

1. Identifying the caller (using the AS);
2. Sending information to the Command Interface; the CI in turn forwards this information to the SMS; and
3. Via the CI, the OCS then receives an answer from the SMS (in terms of the cost of the modification, if the modification is possible).

If modification was requested using the receiver/decoder, the OCS generates a confirmation to the SMS. Otherwise, for example in the case of phone or Minitel, the subscriber is prompted for confirmation and this answer sent to the SMS via the OCS and the CI.

Reset of a parental code involves:

1. Identifying the caller (using AS); and
2. Sending a command to the MG to generate an appropriate EMM bearing an appropriate reset password.

In the case of reset of parental code, the command to reset the code is for security reasons not permitted to originate from the receiver/decoder. Only the SMS, telephone and MINITEL or like can originate such a command. Hence in this

particular case the EMM(s) are broadcast only on air, never on the telephone line.

It will be understood from the above examples of different modes of operation of the OCS that the user can have direct access to the SAS, and in particular the OCS and AS, in that the Communications Servers are directly connected to the SAS, and in particular the OCS. This important feature is concerned with reducing the time for the user to communicate his command to the SAS.

This feature is illustrated further with reference to Figure 12, from which it can be seen that the end user's Set-Top-Box, and in particular its receiver/decoder 2020, has the capability of communicating directly with the Communications Servers 3022 associated with the SAS 3002. Instead of the connection from the end user to the Communications Servers 3022 of the SAS 3002 being through the SMS 3004 the connection is directly to the SAS 3002.

In fact, as directly mentioned two direct connections are provided.

The first direct connection is by a voice link via a telephone 4001 and appropriate telephone line (and/or by MINITEL or like connection where available) where the end users still have to input a series of voice commands or code numbers but time is saved compared with the communication being via the SMS 3004.

The second direct connection is from the receiver/decoder 2020 and the input of data is achieved automatically by the end user inserting his own daughter smartcard 3020 thus relieving the end user of the job of having to input the relevant data which in turn reduces the time taken and the likelihood of errors in making that input.

A further important feature which arises out of the above discussion is concerned with reducing the time taken for the resulting EMM to be transmitted to the end user in order to initiate viewing by the end user of the selected product.

In broad terms, and with reference to Figure 12, the feature is again achieved by

providing the end user's receiver/decoder 2020 with the capability of communicating directly with the Communications Servers 3022 associated with the SAS 3002.

As described earlier the integrated receiver/decoder 2020 is connected directly to the Communications Servers 3022 by the modemmed back channel 4002 so that commands from the decoder 2020 are processed by the SAS 3002, messages generated (including EMMs) and then sent back directly to the decoder 2020 through the back channel 4002. A protocol is used in the communication between the CS 3022 and the receiver/decoder 2020 (as described later), so that the CS receive acknowledgement of receipt of the relevant EMM, thereby adding certainty to the procedure.

Thus, for example, in the case of a pre-book mode the SAS 3002 receives messages from the end user via the smartcard and decoder 2020 via its modem and via the telephone line 4002, requesting access to a specific event/product, and returns a suitable EMM via the telephone line 4002 and modem to the decoder 2020, the modem and decoder being preferably located together in a Set-Top-Box (STB). This is thus achieved without having to transmit the EMM in the MPEG-2 data stream 2002 via the multiplexer and scrambler 2004, the uplink 2012, satellite 2014 and datalink 2016 to enable the end user to view the event/product. This can save considerably on time and bandwidth. Virtual certainty is provided that as soon as the subscriber has paid for his purchase the EMM will arrive at the receiver/decoder 2020.

In the third type of mode of operation of the OCS 3207 described above, the OCS deals with callback requests issued by the SAS. This is illustrated with reference to Figure 13. Typical callback requests have the purpose of ensuring that the receiver/decoder 2020 calls back the SAS via the modemmed back channel 4002 with the information that the SAS requires of the receiver /decoder.

As instructed by the Command Interface 3102, the subscription chain Message Generator 3106 generates and sends to the receiver/decoder 202 a callback EMM. This EMM is ciphered by the Ciphering Unit 3008 for security reasons. The EMM may contain the time/date at which the receiver/decoder should wake up and perform

a callback on its own, without being explicitly solicited; the EMM may also typically contain the phone numbers which the terminal must dial, the number of further attempts after unsuccessful calls and the delay between two calls.

When receiving the EMM, or at the specified time-date, the receiver/decoder connects to the Communications Servers 3022. The OCS 3207 first identifies the caller, using the AS 3202, and verifies certain details, such as smartcard operator and subscriber details. The OCS then asks the smartcard 3020 to send various ciphered information (such as the relevant session numbers, when the session was watched, how many times the subscriber is allowed to view the session again, the way in which the session was viewed, the number of remaining tokens, the number of prebooked sessions, etc). This information is deciphered by the PPV chain Message Generator 3210, again using the Ciphering Unit 3008. The OCS adds this information to a callback information storage file 3214 for later processing and passing to the SMS 3004. The information is ciphered for security reasons. The whole procedure is repeated until there is nothing more to be read from the smartcard.

One particular preferred feature of the callback facility is that before reading the smartcard (so just after the identification of the caller using the AS 3202 as described above) a check is made by the SAS 3002 that the receiver/decoder is indeed a genuine one rather than a pirated version or computer simulation. Such a check is carried out in the following manner. The SAS generates a random number, which is received by the receiver/decoder, ciphered, and then returned to the SAS. The SAS deciphers this number. If the deciphering is successful and the original random number is retrieved, it is concluded that the receiver/decoder is genuine, and the procedure continues. Otherwise, the procedure is discontinued.

Other functions which may occur during the callback are erasure of obsolete sessions on the smartcard, or filling of the wallet (this latter also being described later under the section entitled "Smartcard").

Also as regards the Pay Per View Chain area 3200, description is now made of the

Communications Servers 3022. At the hardware level, these comprise in the preferred embodiment a DEC Four parallel processor machine. At the software architecture level, with reference to Figure 14, in many respects the Communications Servers are conventional. One particular divergence from conventional designs arises from the fact that the Servers must serve both receiver/decoders 2020 and voice communication with conventional telephones 4001, as well possibly as MINITEL or like systems.

It will be noted in passing that two Order Centralized Servers 3207 are shown in Figure 14 (as "OCS1" and "OCS2"). Naturally any desired number may be provided.

The Communication Servers include two main servers ("CS1" and "CS2") as well as a number of frontal servers ("Frontal 1" and "Frontal 2"); whilst two frontal servers are shown in the figure, typically 10 or 12 may be provided per main server. Indeed, although two main servers CS1 and CS2 and two frontal servers, Frontal 1 and Frontal 2, have been shown, any number could be used. Some redundancy is usually desirable.

CS1 and CS2 are coupled to OCS1 and OCS2 via high level TCP/IP links 3230, whilst CS1 and CS2 are coupled to Frontal 1 and Frontal 2 via further TCP/IP links 3232.

As illustrated, CS1 and CS2 comprise servers for "SENDR" (transmission), "RECVR" (reception), "VTX" (MINITEL, PRESTEL or the like), "VOX" (voice communication), and "TRM" (communication with the receiver/decoder). These are coupled to the "BUS" for communication of signals to the Frontal servers.

CS1 and CS2 communicate directly with the receiver/decoders 2020 via their modemmed back channels 4002 using the X25 public network common protocol. The relatively low-level protocol between the Communications Servers 3022 and the receiver/decoders 3020 is in one preferred embodiment based upon the V42 standard international CCITT protocol, which provides reliability by having error detection and data re-transmission facilities, and uses a checksum routine to check the integrity of

the re-transmission. An escape mechanism is also provided in order to prevent the transmission of disallowed characters.

On the other hand, voice telephone communication is carried out via the Frontal Communications Servers, each capable of picking up, say, 30 simultaneous voice connections from the connection 3234 to the local telephone network via the high speed "T2" (E1) standard telephony ISDN lines.

Three particular functions of the software portion of the Communications Servers (which could of course alternatively be implemented fully in hardware) are firstly to convert the relatively low level protocol information received from the receiver/decoder into the relatively high level protocol information output to the OCS, secondly to attenuate or control the number of simultaneous connections being made, and thirdly to provide several simultaneous channels without any mixing. In this last regard, the Communications Servers play the role of a form of multiplexer, with the interactions in a particular channel being defined by a given Session ID (identifier), which is in fact used throughout the communication chain.

Finally as regards the Pay Per View Chain area 3200, and with reference again to Figure 5, the Server for Programme Broadcast (SPB) 3208 is coupled to one or more Programme Broadcasters 3250 (which would typically be located remotely from the SAS) to receive programme information. The SPB filters out for further use information corresponding to PPV events (sessions).

A particularly important feature is that the filtered programme event information is passed by the SPB to the MG which in turn sends a directive (control command) to the ME to change the rate of cyclic emission of the EMMs in given circumstances; this is done by the ME finding all EMMs with the relevant session identifier and changing the cycle rate allocated to such EMMs. This feature might be thought of as a dynamic allocation of bandwidth for specific EMMs. Cyclic EMM emission is discussed in more detail in the section below concerned with the EMM Injector.

The circumstances in which the cycle rate is changed are now described with reference to Figure 15, which demonstrates how cycle rate 3252 is raised a short while (say 10 minutes) before a particular PPV programme event until the end of the event from a slow cycle rate of say once every 30 minutes to a fast cycle rate of say once every 30 seconds to 1 minute in order to meet the anticipated extra user demand for PPV events at those times. In this way bandwidth can be allocated dynamically according to the anticipated user demand. This can assist in reducing the overall bandwidth requirement.

The cycle rate of other EMMs may also be varied. For example the cycle rate of subscription EMMs may be varied by the Multiplexer and Scrambler 2004 sending the appropriate bitrate directive.

EMM Injector

Concerning the EMM Injector 3300, details of the Message Emitters 3302 to 3308, forming part of the EMM Injector and acting as output means for the Message Generator, are now described with reference to Figure 16. Their function is take the EMMs and to pass them cyclically (in the manner of a carousel) via respective links 3314 and 3316 to the Software Multiplexers 3310 and 3312 and thence to the hardware multiplexers and scramblers 2004. In return the software multiplexers and scramblers 2004 generate a global bitrate directive to control the overall cycling rate of the EMMs; to do so, the MEs take into account various parameters such as the cycle time, the size of EMM, and so on. In the figure, EMM_X and EMM _Y are group EMMs for operators X and Y, whilst EMM_Z are other EMMs for either operator X or operator Y.

Further description proceeds for an exemplary one of the Message Emitters; it will be appreciated that the remaining MEs operate in similar fashion. The ME operates under control of directives from the MG, most notably transmission start and stop time and emission rate, as well as session number if the EMM is a PPV EMM. In relation to the emission rate, in the preferred embodiment the relevant directive may take one of five values from Very fast to Very slow. The numeric values are not specified in

the directive, but rather the ME maps the directive to an actual numeric value which is supplied by the relevant part of the SAS. In the preferred embodiment, the 5 emission rates are as follows:-

1.	Very fast	- every 30 seconds
2.	Fast	- every minute
3.	Medium	- every 15 minutes
4.	Slow	- every 30 minutes
5.	Very slow	- every 30 minutes

The ME has first and second databases 3320 and 3322. The first database is for those EMMs which have not yet achieved their broadcast date; these are stored in a series of chronological files in the database. The second database is for EMMs for immediate broadcast. In the event of a system crash, the ME is arranged to have the ability to re-read the relevant stored file and perform correct broadcast. All the files stored in the databases are updated upon request from the MG, when the MG wishes to maintain consistency between incoming directives and EMMs already sent to the ME. The EMMs actually being broadcast are also stored in Random Access Memory 3324.

A combination of the FIFOs 3162 and 3164 in the Message Generator and the databases 3320 and 3322 in the Message Emitter means that the two can operate in standalone mode if the link 3166 between them is temporarily broken; the ME can still broadcast EMMs.

The Software Multiplexers (SMUX) 3310 and 3312 provide an interface between the MEs and the hardware multiplexers 2004. In the preferred embodiment, they each receive EMMs from two of the MEs, although in general there is no restriction on the number of MEs that can be connected with one SMUX. The SMUXs concentrate the EMMs and then pass them according to the type of EMM to the appropriate hardware multiplexer. This is necessary because the hardware multiplexers take the different types of EMMs and place them at different places in the MPEG-2 stream. The

SMUX's also forward global bitrate directives from the hardware multiplexers to the MEs.

One particularly important feature of the ME is that it emits EMMs in random order. The reason for this is as follows. The Message Emitter has no ability to sense or control what it emits to the multiplexer. Hence it is possible that it may transmit two EMMs which are to be received and decoded by the receiver/decoder 2020 back to back. In such circumstances, further, it is possible that if the EMMs are insufficiently separated the receiver/decoder and smartcard will be unable to sense and decode properly the second of the EMMs. Cyclically emitting the EMMs in random order can solve this problem.

The manner in which randomization is achieved is now described with reference to Figure 17; in the preferred embodiment the necessary software logic is implemented in the ADA computer language. A particularly important part of the randomization is the correct storage of the EMMs in the databases 3320 and 3322 (which are used for backup purposes) and in the RAM 3324. For a particular cycle rate and operator, the EMMs are stored in a two-dimensional array, by rank 3330 (going say from A to Z) and number in the rank 3332 (going from 0 to N). A third dimension is added by cycle rate 3334, so that there are as many two-dimensional arrays as there are cycle rates. In the preferred embodiment there are 256 ranks and typically 200 or 300 EMMs in each rank; there are 5 cycle rates. A final dimension to the array is added by the presence of different operators; there are as many three-dimensional arrays as there are operators. Storage of the data in this fashion can permit rapid retrieval in the event that the MG wants to delete a particular EMM.

Storage of the EMMs takes place according to the "hash" algorithm (otherwise known as the "one-way hash function". This operates on a modulo approach, so that successive ranks are filled before a higher number in the rank is used, and the number of EMMs in each rank remains roughly constant. The example is considered of there being 256 ranks. When the MG sends the ME an EMM with identifer (ID) 1, the rank "1" is assigned to this EMM, and it takes the first number 3332 in the rank 3330.

The EMM with ID 2 is assigned the rank "2", and so on, up to the rank 256. The EMM with ID 257 is assigned the rank "1" again (based on the modulo function), and takes the second number in the first rank, and so on.

Retrieval of a specific EMM, for example when deletion of a specific EMM is requested by the MG, is effected by means of the inverse of the above. The hash algorithm is applied to the EMM ID to obtain the rank, after which the number in the rank is found.

The actual randomization occurs when the EMMs are, on a cyclical basis, retrieved from RAM 3324 using the randomization means 3340 which is implemented in the hardware and/or software of the Message Emitter. The retrieval is random, and again based on the hash algorithm. Firstly, a random number (in the above example initially in the range 1 to 256) is chosen, to yield the particular rank of interest. Secondly, a further random number is chosen to yield the particular number in the rank. The further random number is selected according to the total number of EMMs in a given rank. Once a given EMM has been selected and broadcast, it is moved to a second identical storage area in the RAM 3324, again using the hash function. Hence the first area diminishes in size as the EMMs are broadcast, to the extent that, once a complete rank has been used, this is deleted. Once the first storage area is completely empty, it is replaced by the second storage area before a new round of EMM broadcast, and vice versa.

In the above fashion, after two or three cycles of the EMMs, statistically the chances of any two EMMs destined for the same end user being transmitted back to back is negligible.

At regular intervals whilst the EMMs are being stored the computer 3050 computes the number of bytes in storage and from this computes the bitrate of emission given the global bitrate directive from the multiplexer and software multiplexer.

Reference was made above to the backup databases 3320 and 3322. These are in fact

in the preferred embodiment sequential file stores, which hold a backup version of what is in the RAM 3324. In the event of failure of the Message Emitter and subsequent restart, or more generally when the ME is being restarted for whatever reason, a link is made between the RAM and the databases, over which the stored EMMs are uploaded to RAM. In this way, the risk of losing EMMs in the event of failure can be removed.

Similar storage of PPV EMMs occurs to that described above in relation to subscription EMMs, with the rank typically corresponding to a given operator and the number in the rank corresponding to the session number.

Smartcard

A daughter, or "subscriber", smartcard 3020 is schematically shown in Figure 18 and comprises an 8 bit microprocessor 110, such as a Motorola 6805 microprocessor, having an input/output bus coupled to a standard array of contacts 120 which in use are connected to a corresponding array of contacts in the card reader of the receiver/decoder 2020, the card reader being of conventional design. The microprocessor 110 is also provided with bus connections to preferably masked ROM 130, RAM 140 and EEPROM 150. The smartcard complies with the ISO 7816-1, 7816-2 and 7816-3 standard protocols which determine certain physical parameters of the smartcard, the positions of the contacts on the chip and certain communications between the external system (and particularly the receiver/decoder 2020) and the smartcard respectively and which will therefore not be further described here. One function of the microprocessor 110 is to manage the memory in the smartcard, as now described.

The EEPROM 150 contains certain dynamically-created operator zones 154, 155, 156 and dynamically-created data zones which will now be described with reference to Figure 19.

Referring to Figure 19, EEPROM 150 comprises a permanent "card ID" (or manufacturer) zone 151 of 8 bytes which contains a permanent subscriber smartcard

identifier set by the manufacturer of the smartcard 3020.

When the smartcard is reset, the microprocessor 110 issues a signal to receiver/decoder 2020, the signal comprising an identifier of the conditional access system used by the smartcard and data generated from data stored in the smartcard, including the card ID. This signal is stored by the receiver/decoder 2020, which subsequently utilises the stored signal to check whether the smartcard is compatible with the conditional access system used by the receiver/decoder 2020.

The EEPROM 150 also contains a permanent "random number generator" zone 152 which contains a program for generating pseudo-random numbers. Such random numbers are used for diversifying transaction output signals generated by the smartcard 3020 and sent back to the broadcaster.

Below the random number generator zone 152 a permanent "management" zone 153 of 144 bytes is provided. The permanent management zone 153 is a specific operator zone utilised by a program in the ROM 130 in the dynamic creation (and removal) of zones 154, 155, 156... as described below. The permanent management zone 153 contains data relating to the rights of the smartcard to create or remove zones.

The program for dynamically creating and removing zones is responsive to specific zone creation (or removal) EMMs which are transmitted by the SAS 3002 and received by the receiver/decoder 2020 and passed to the subscriber smartcard 3020. In order to create the EMMs the operator requires specific keys dedicated to the management zone. This prevents one operator from deleting zones relating to another operator.

Below the management zone 153 is a series of "operator ID" zones 154, 155, 156 for operators 1, 2 ..... N respectively. Normally at least one operator ID zone will be preloaded into the EEPROM of the subscriber smartcard 3020 so that the end user can decrypt programmes broadcast by that operator. However further operator ID zones can subsequently be dynamically created using the management zone 153 in response

to a transaction output signal generated via his smartcard 3020 by the end user (subscriber), as will subsequently be described.

Each operator zone 154, 155, 156 contains the identifier of the group to which the smartcard 3020 belongs, and the position of the smartcard within the group. This data enables the smartcard (along with the other smartcards in its group) to be responsive to a broadcast "group" subscription EMM having that group's address (but not the smartcard's position in the group) as well as to an "individual" (or commercial offers subscription) EMM addressed only to that smartcard within the group. There can be 256 member smartcards of each such group and this feature therefore reduces significantly the bandwidth required for broadcasting EMMs.

In order to reduce further the bandwidth required for broadcasting "group" subscription EMMs, the group data in each operator zone 154, 155, 156 and all similar zones in the EEPROM of smartcard 3020 and the other daughter smartcards is continually updated to enable a particular smartcard to change its position in each group to fill any holes created by e.g. deletion of a member of the group. The holes are filled by the SAS 3002 as in the STM server 3104 there is a list of such holes.

In this manner fragmentation is reduced and each group's membership is maintained at or near the maximum of 256 members.

Each operator zone 154, 155, 156 is associated with one or more "operator data objects" stored in the EEPROM 150. As shown in Figure 19, a series of dynamically created "operator data" objects 157–165 are located below the operator ID zones. Each of these objects is labelled with:

a) an "identifier" 1, 2, 3 .... N corresponding to its associated operator 1, 2, 3 ... N as shown in its left hand section in Figure 19;

b) an "ID" indicating the type of object; and

c) a "data" zone reserved for data, as shown in the right hand section of each relevant operator object in Figure 19. It should be understood that each operator is associated with a similar set of data objects so that the following description of the

types of data in the data objects of operator 1 is also applicable to the data objects of all the other operators. Also it will be noted that the data objects are located in contiguous physical regions of the EEPROM and that their order is immaterial.

Deletion of a data object creates a "hole" 166 in the smartcard, that is, the number of bytes that the deleted objects had previously occupied are not immediately occupied. The thus "freed" number of bytes, or "hole" are labelled with:

a) an "identifier" 0; and

b) an "ID" indicating that the bytes are free to receive an object.

The next data object created fills the hole, as identified by the identifier 0. In this manner the limited memory capacity (4 kilobytes) of the EEPROM 150 is efficiently utilised.

Turning now to the set of data objects associated with each operator, examples of the data objects are now described.

Data object 157 contains an EMM key used for decrypting encrypted EMM's received by the receiver/decoder 2020. This EMM key is permanently stored in the data object 157. This data object 157 may be created prior to distribution of the smartcard 3020, and/or may be created dynamically when creating a new operator zone (as described above).

Data object 159 contains ECM keys which are sent by the associated operator (in this case operator 1) to enable the end user to decrypt the particular "bouquet" of programs to which he has subscribed. New ECM keys are sent typically every month, along with a group subscription (renewal) EMM which renews the end user's overall right to view the broadcast from (in this case) operator 1. The use of separate EMM and ECM keys enables viewing rights to be purchased in different ways (in this embodiment by subscription and individually (Pay Per View)) and also increases security. The Pay Per View (PPV) mode will be described subsequently.

Since new ECM keys are sent periodically, it is essential to prevent a user from usin old ECM keys, for example by switching off the receiver/decoder or re-setting a clock to prevent expiry of an old ECM key so that a timer in the receiver/decoder 2020 could be overridden. Accordingly operator zone 154 comprises an area (typically having a size of 2 bytes) containing an obsolescence date of the ECM keys. The smartcard 3020 is arranged to compare this date with the current date which is contained in received ECMs and to prevent decryption if the current date is later than the obsolescence date. The obsolescence date is transmitted via EMMs, as described above.

Data object 161 contains a 64 bit subscription bitmap which is an exact representation of the broadcast operator's programs to which the subscriber has subscribed. Every bit represents a program and is set to "1" if it is subscribed to and "0" if it is not.

Data object 163 contains a quantity of tokens which can be used by the consumer in PPV mode to buy viewing rights to an imminent broadcast e.g. in response to a free preview or other advertisement. Data object 163 also contains a limit value, which may be set to e.g. a negative value to allow credit to the consumer. Tokens can be purchased e.g. by credit and via the modemmed back channel 4002, or by using a voice server in combination with a credit card, for example. A particular event can be charged as one token or a number of tokens.

Data object 165 contains a description of a PPV event, as shown with reference to table 167 of Figure 20.

The PPV event description 167 contains a "session ID" 168 identifying the viewing session (corresponding to the program and the time and date of broadcasting) a "session mode" 169 indicating how the viewing right is being purchased (e.g. in pre-book mode), a "session index" 170 and a "session view" 171.

In respect of receiving a programme in PPV mode, the receiver decoder 2020 determines whether the programme is one sold in PPV mode. If so, the decoder 2020

checks, using the items stored in the PPV event description 167 whether the session ID for the programme is stored therein. If the session ID is stored therein, the control word is extracted from the ECM.

If the session ID is not stored therein, by means of a specific application the receiver/decoder 2020 displays a message to the end user indicating that he has the right to view the session at a cost of, say, 25 tokens, as read from the ECM or to connect to the communications servers 3022 to purchase the event. Using the tokens, if the end user answers "yes" (by means of remote controller 2026 (see Figure 2)) the decoder 2020 sends the ECM to the smartcard, the smartcard decreases the wallet of the smartcard 3020 by 25 tokens, writes the session ID 168, the session mode 169, the session index 170 and the session view 171 in the PPV event description 167 and extracts and deciphers the control word from the ECM.

In the "pre-book" mode, an EMM will be passed to the smartcard 3020 so that the smartcard will write the session ID 168, the session mode 169, the session index 170 and the session view 171 in the PPV event description 167 using the EMM.

The session index 170 can be set to differentiate one broadcast from the other. This feature permits authorization to be given for a subset of broadcasts, for example, 3 times out of 5 broadcasts. As soon as an ECM with a session index different from the current session index 170 stored in the PPV event description 167 is passed to the smartcard, the number of the session view 171 is decreased by one. When the session view reaches zero, the smartcard will refuse to decipher an ECM with a different session index to the current session index.

The initial value of the session view depends only on the way in which the broadcast supplier wishes to define the event to which it relates; the session view for a respective event may take any value.

The microprocessor 110 in the smartcard implements a counting and a comparison program to detect when the limit to the number of viewings of a particular program

has been reached.

All of the session ID 168, the session mode 169, the session index 170 and the session view 171 in the PPV event description 167 may be extracted from the smartcard using the "call-back" procedure as described previously.

Each receiver/decoder 2020 contains an identifier which may either identify uniquely that receiver/decoder or identify its manufacturer or may classify it in some other way in order to enable it to work only with a particular individual smartcard, a particular class of smartcards made by the same or a corresponding manufacturer or any other class of smartcards which are intended for use with that class of receiver/decoders exclusively.

In this manner the receiver/decoders 2020 which have been supplied by one broadcast supplier to the consumer are protected against the use of non-authorised daughter smartcards 3020.

Additionally or alternatively to this first "handshake" between the smartcard and the receiver, the EEPROM of the smartcard 3020 could contain a field or bitmap describing the categories of receiver/decoders 2020 with which it can function. These could be specified either during the manufacture of the smartcard 3020 or by a specific EMM.

The bitmap stored in the smartcard 3020 typically comprises a list of up to 80 receiver/decoders, each identified with a corresponding receiver/decoder ID with which the smartcard may be used. Associated with each receiver/decoder is a level "1" or "0" indicating whether the smartcard may be used with the receiver/decoder or not, respectively. A program in the memory 2024 of the receiver/decoder searches for the identifier of the receiver/decoder in the bitmap stored in the smartcard. If the identifier is found, and the value associated with the identifier is "1", then the smartcard is "enabled"; if not, then the smartcard will not function with that receiver/decoder.

In addition, if, typically because of an agreement between operators, it is desired to authorize the use of other smartcards in a particular receiver/decoder, specific EMMs will be sent to those smartcards to change their bitmap via the transponder 2014.

Each broadcast supplier may differentiate his subscribers according to certain predetermined criteria. For example, a number of subscribers may be classed as "VIPs". Accordingly, each broadcast supplier may divide his subscribers into a plurality of subsets, each subset comprising any number of subscribers.

The subset to which a particular subscriber belongs is set in the SMS 3004. In turn, the SAS 3002 transmits an EMM to the subscriber which writes information (typically of length 1 byte) concerning the subset to which the subscriber belongs into the relevant operator data zone, say 154, of the EEPROM of the smartcard. In turn, as events are broadcast by the broadcast supplier, an ECM, typically of 256 bits, is transmitted with the event and indicating which of the subsets of subscribers may view the event. If, according to the information stored in the operator zone, the subscriber does not have the right to view the event, as determined by the ECM, programme viewing is denied.

This facility may be used, for example, to switch off all of a given operator's smartcards in a particular geographical region during the transmission of a particular program, in particular a program relating to a sports fixture taking place in that geographical region. In this manner football clubs and other sport bodies can sell broadcasting rights outside their locality whilst preventing local supporters from viewing the fixture on television. In this manner the local supporters are encouraged to buy tickets and attend the fixture.

Each of the features associated with zones 151 to 172 is considered to be a separate invention independent of the dynamic creation of zones.

It will be understood that the present invention has been described above purely by way of example, and modifications of detail can be made within the scope of the

invention.

Each feature disclosed in the description, and (where appropriate) the claims and drawings may be provided independently or in any appropriate combination.

In the aforementioned preferred embodiments, certain features of the present invention have been implemented using computer software. However, it will of course be clear to the skilled man that any of these features may be implemented using hardware. Furthermore, it will be readily understood that the functions performed by the hardware, the computer software, and such like are performed on or using electrical and like signals.

Cross reference is made to our co-pending applications, all bearing the same filing date, and entitled Signal Generation and Broadcasting (Attorney Reference no. PC/ASB/19707), Smartcard for use with a Receiver of Encrypted Broadcast Signals, and Receiver (Attorney Reference No. PC/ASB/19708), Broadcast and Reception System and Conditional Access System therefor (Attorney Reference No. PC/ASB/19710), Downloading a Computer File from a Transmitter via a Receiver/Decoder to a Computer (Attorney Reference No. PC/ASB/19711), Transmission and Reception of Television Programmes and Other Data (Attorney Reference No. PC/ASB/19712), Downloading Data (Attorney Reference No. PC/ASB/19713), Computer Memory Organisation (Attorney Reference No. PC/ASB/19714), Television or Radio Control System Development (Attorney Reference No. PC/ASB/19715), Extracting Data Sections from a Transmitted Data Stream (Attorney Reference No. PC/ASB/19716), Access Control System (Attorney Reference No. PC/ASB/19717), Data Processing System (Attorney Reference No. PC/ASB/19718), and Broadcast and Reception System, and Receiver/Decoder and Remote Controller therefor (Attorney Reference No. PC/ASB/19720). The disclosures of these documents are incorporated herein by reference. The list of applications includes the present application.

CLAIMS

1. A conditional access system comprising:

means for generating a plurality of messages; and

means for receiving the messages, said receiving means being adapted to communicate with said generating means via a communications server connected directly to said generating means.

2. A conditional access system according to Claim 1, wherein said message is an entitlement message for transmission to the receiving means, said generating means being adapted to generate entitlement messages in response to data received from said receiving means.

3. A conditional access system according to Claim 1 or 2, wherein said generating means is arranged to transmit a message as a packet of digital data to said receiving means either via said communications server or via a satellite transponder.

4. A conditional access system according to any preceding claim, wherein said receiving means is connectable to said communications server via a modem and telephone link.

5. A conditional access system for affording conditional access to subscribers, comprising:

a subscriber management system;

a subscriber authorization system coupled to the subscriber management system; and

a communications server; said server being connected directly to the subscriber authorization system.

6. A conditional access system according to Claim 5, further comprising a receiver/decoder for the subscriber, the receiver/decoder being connectable to said communications server, and hence to said subscriber authorization system, via a

modem and telephone link.

7. A broadcast and reception system including a conditional access system according to any preceding claim.

8. A broadcast and reception system comprising:

means for generating a plurality of entitlement messages relating to broadcast programs;

means for receiving said messages from said generating means; and

means for connecting the receiving means to the generating means to receive said messages, said connecting means being capable of effecting a dedicated connection between the receiving means and the generating means.

9. A broadcast and reception system comprising:

means for generating a plurality of entitlement messages relating to broadcast programs;

means for receiving said messages from said generating means via a modem; and

means for connecting said modem to said generating means and said receiving means.

10. A broadcast and reception system according to Claim 9, wherein said generating means is connected to said modem via a communications server.

11. A broadcast and reception system according to Claim 9 or 10, wherein said receiving means is adapted to communicate with said generating means via said modem and connecting means.

12. A broadcast and reception system according to Claim 11, wherein said receiving means comprises means for reading a smartcard insertable thereinto by an end user, the smartcard having stored therein data to initiate automatically the transmission of a message from said receiving means to said generating means upon

insertion of the smartcard by the end user.

13. A broadcast and reception system according to Claim 11or 12, further comprising a voice link to enable the end user of the broadcast and reception system to communicate with the generating means.

14. A broadcast and reception system according to any of Claims 8 to 13, wherein said receiving means comprises a receiver/decoder comprising means for receiving a compressed MPEG-type signal, means for decoding the received signal to provide a television signal and means for supplying the television signal to a television.

15. A broadcast and reception system, comprising, at the broadcast end:

a broadcast system including means for broadcasting a callback request;

and at the reception end:

a receiver including means for calling back the broadcast system in response to the callback request.

16. A system according to Claim 15, wherein the means for calling back the broadcast system includes a modem connectable to a telephone system.

17. A system according to Claim 15 or 16, wherein the means for calling back the broadcast system is arranged to transfer to the broadcast system information concerning the receiver.

18. A system according to Claim 17, wherein the broadcast system includes means for storing the information.

19. A system according to any of Claims 15 to 18, wherein the broadcast means is arranged to broadcast a callback request which includes a command that the callback be made at a given time, and the means for calling back the broadcast system is arranged to respond to said command.

20. A system according to any of Claims 15 to 19, wherein the broadcasting means is arranged to broadcast as the callback request one or more entitlement messages for broadcast.

21. A system according to any of Claims 15 to 20, wherein the broadcast system includes means for generating a check message and passing this to the receiver, the receiver includes means for encrypting the check message and passing this to the broadcast system, and the broadcast system further includes means for decrypting the check message received from the receiver and comparing this with the original check message.

22. A conditional access system or a broadcast and reception system substantially as herein described with reference to and as illustrated in the accompanying drawings, and especially Figures 12, 13 or 14 thereof.

Fig.1.

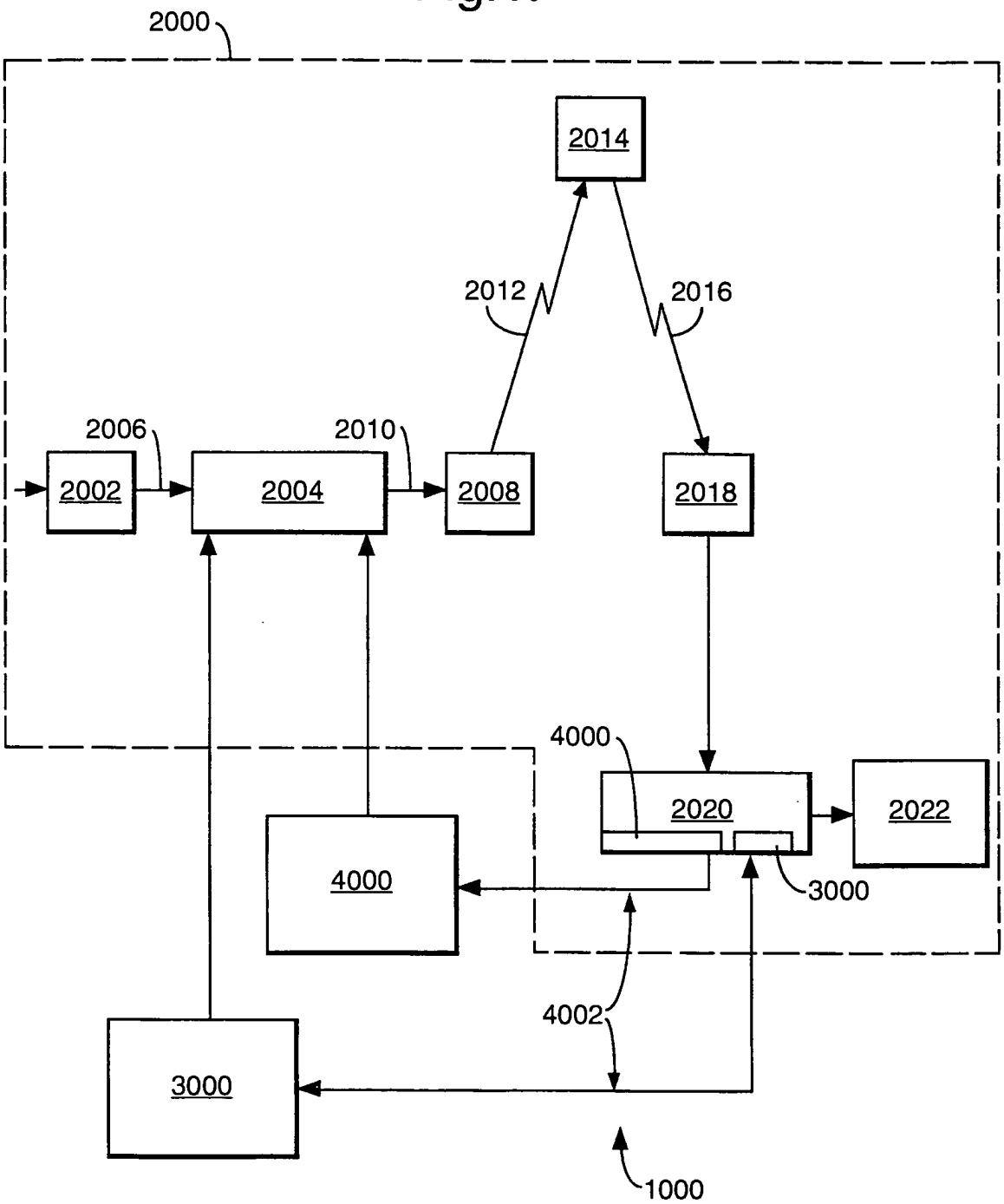


Fig.2.

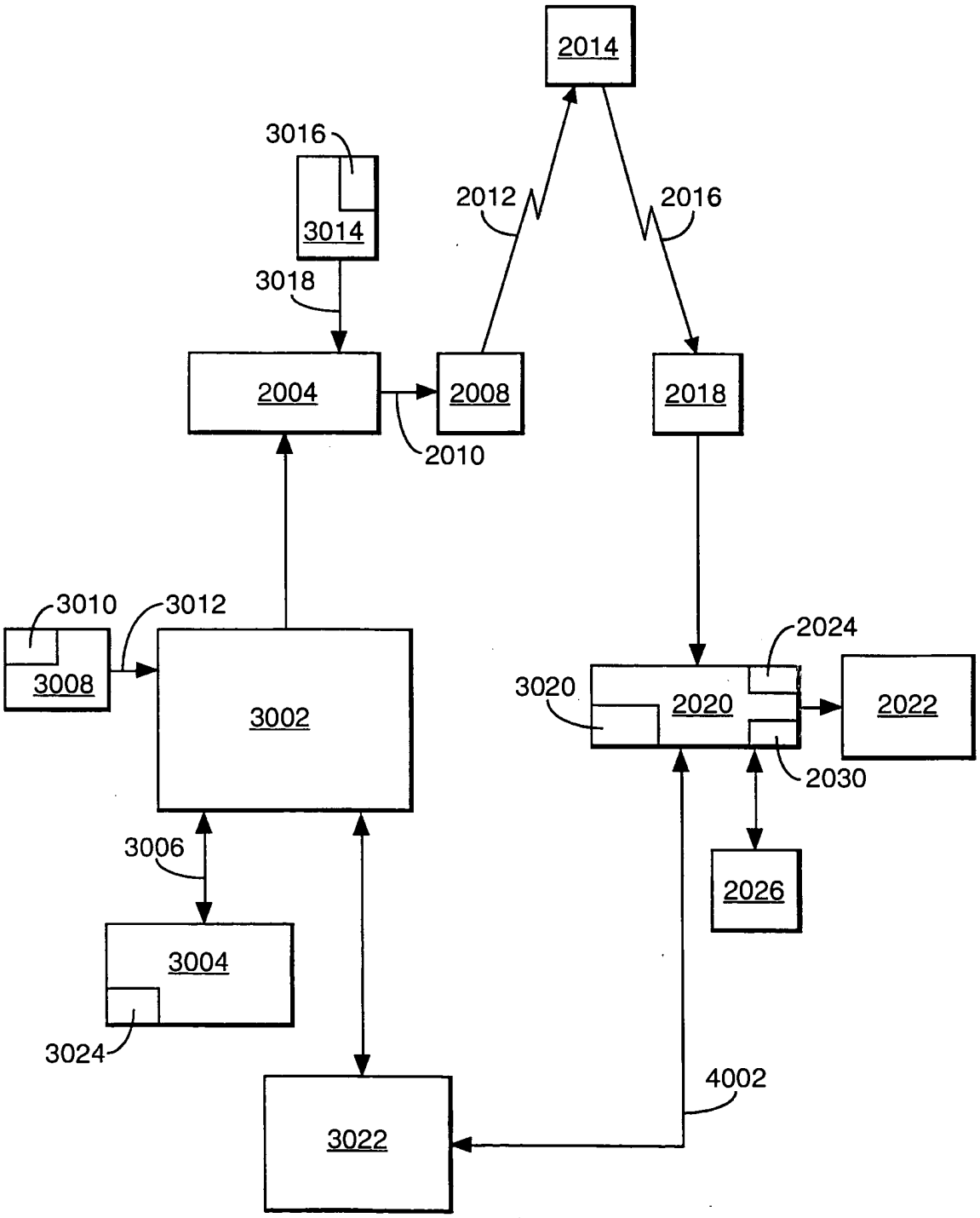
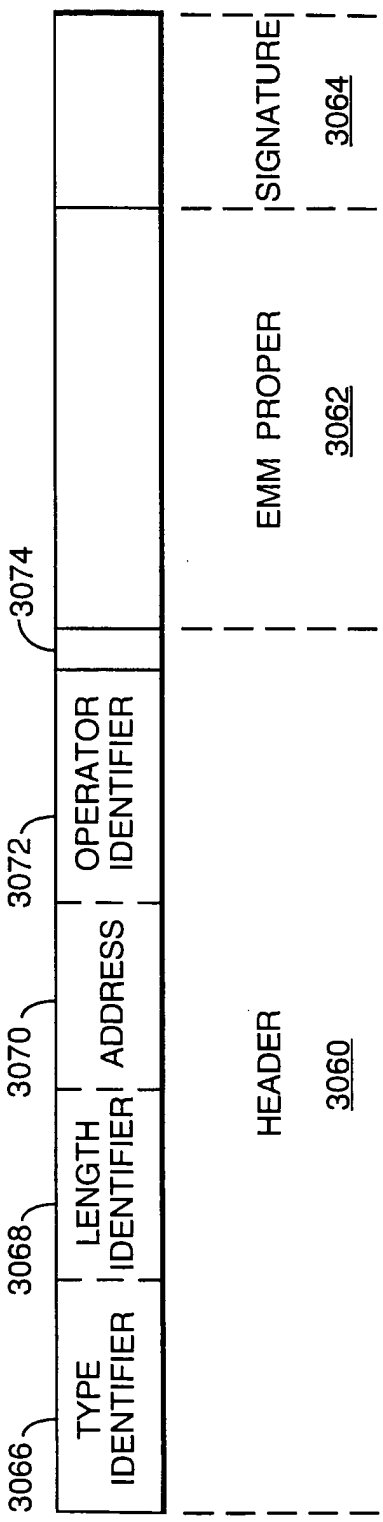


Fig.3.



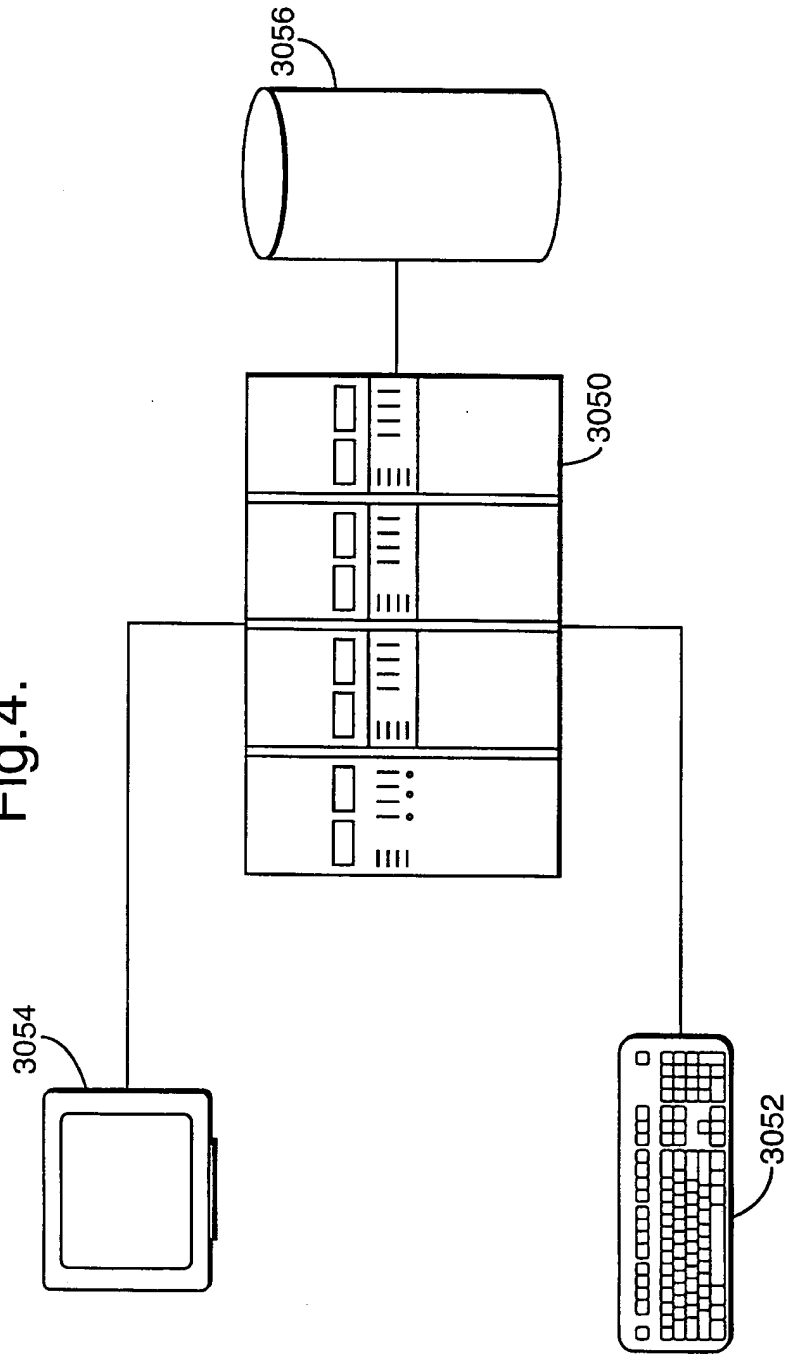
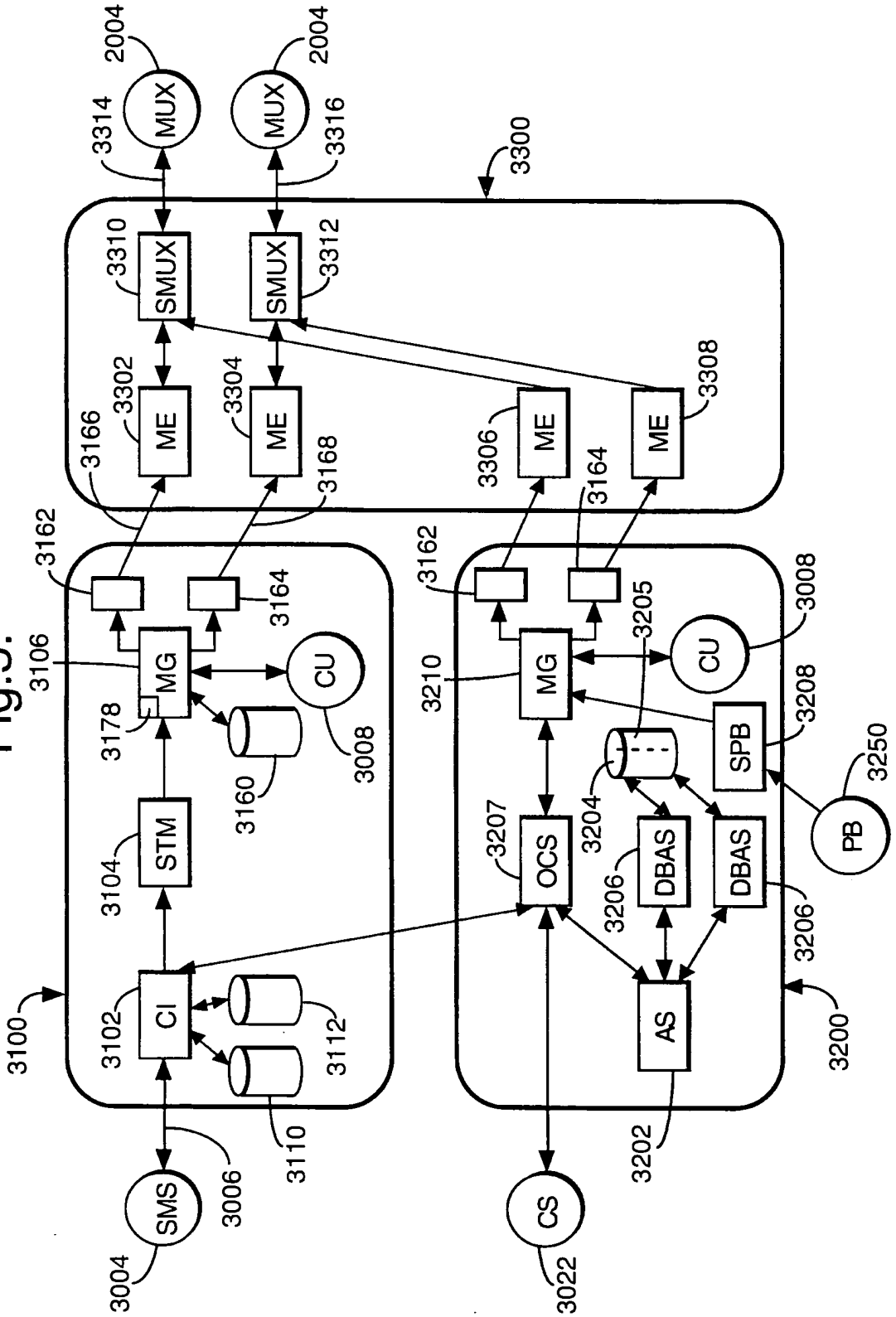


Fig.4.

Fig.5.



SUBSTITUTE SHEET (RULE 26)

Fig.6.

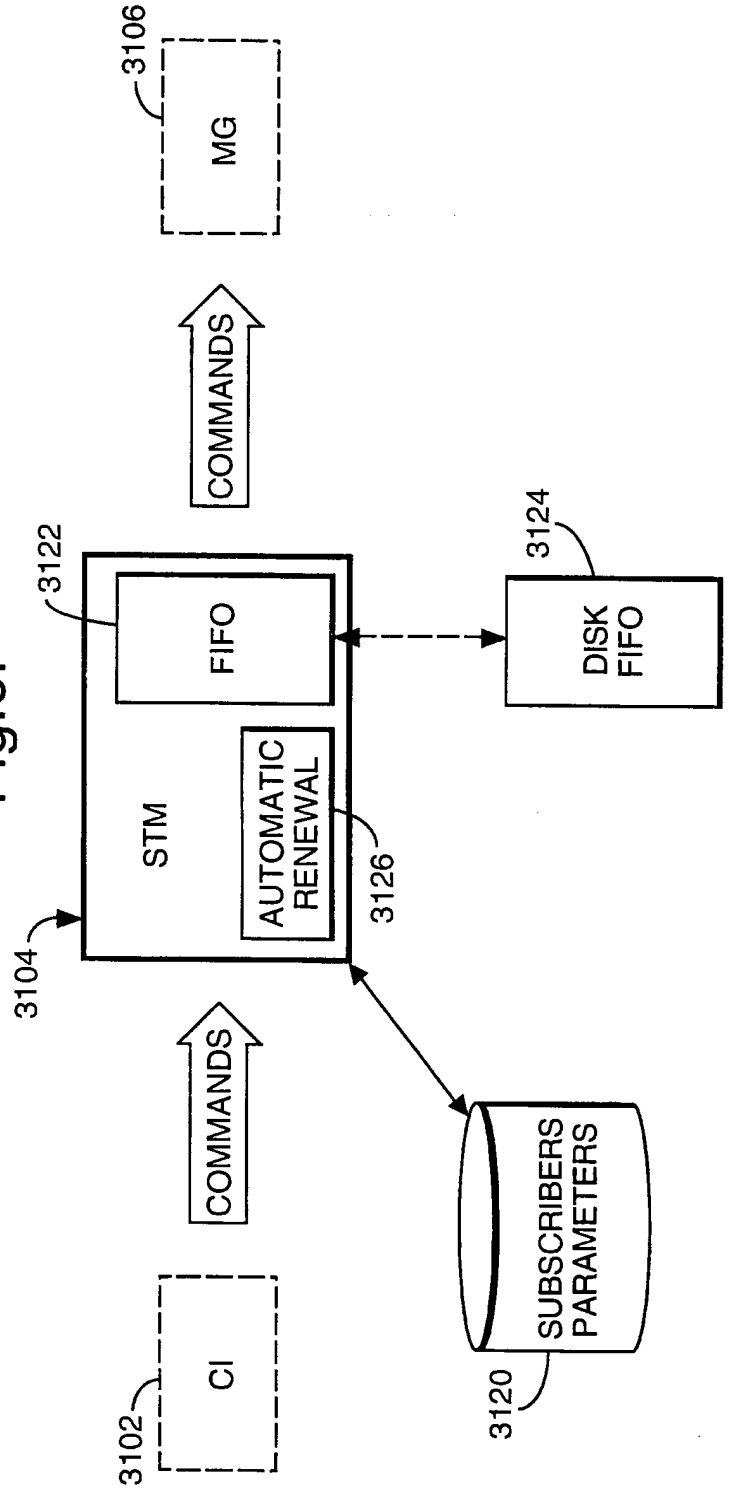


Fig.7.

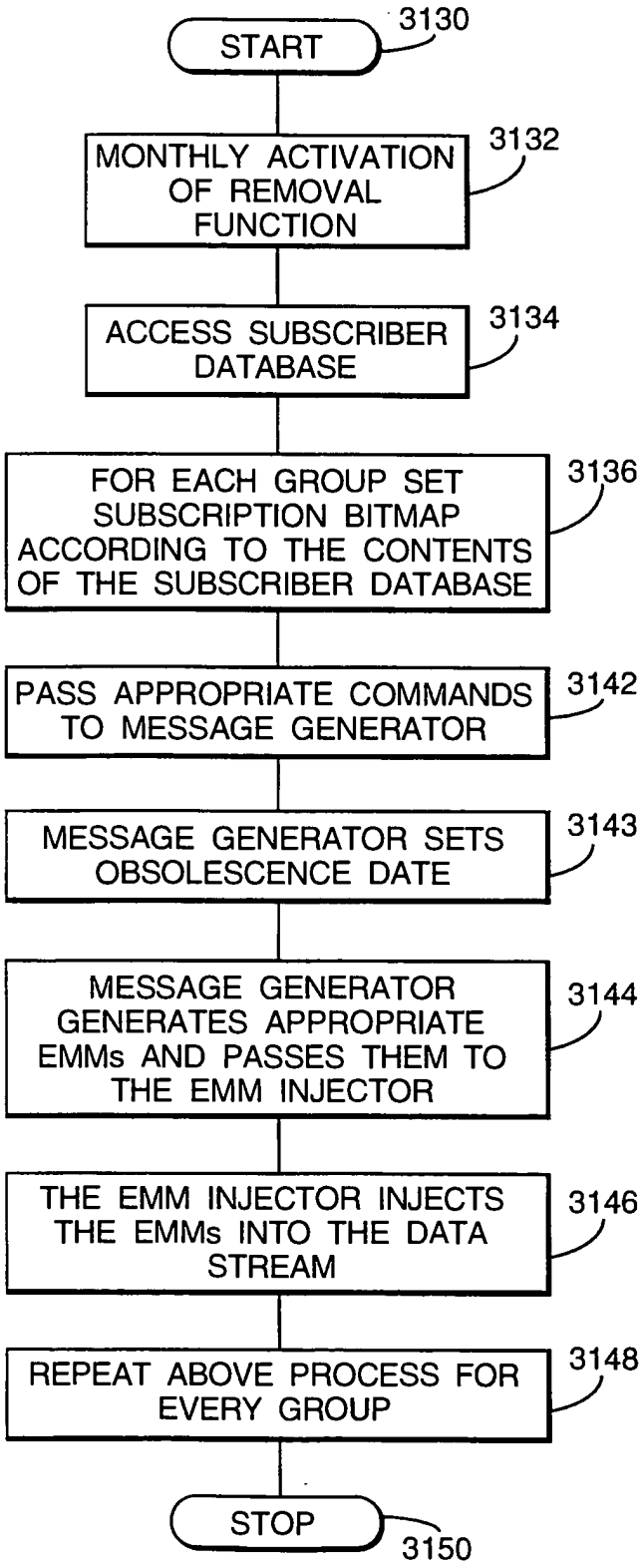


Fig.8.

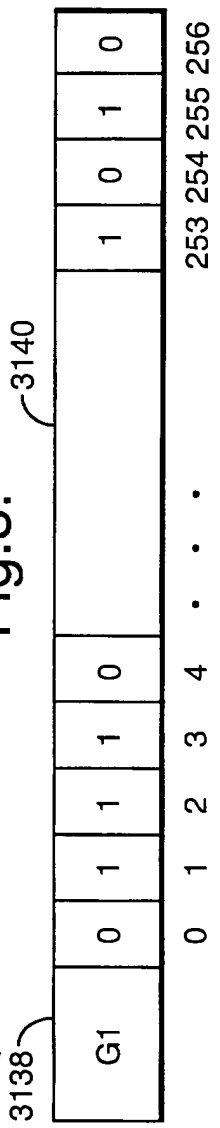


Fig.9.

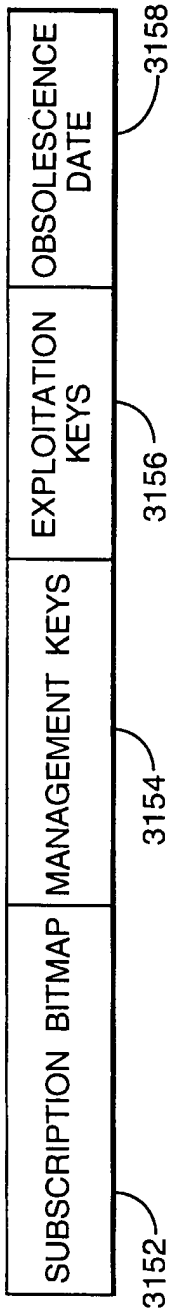


Fig.10.

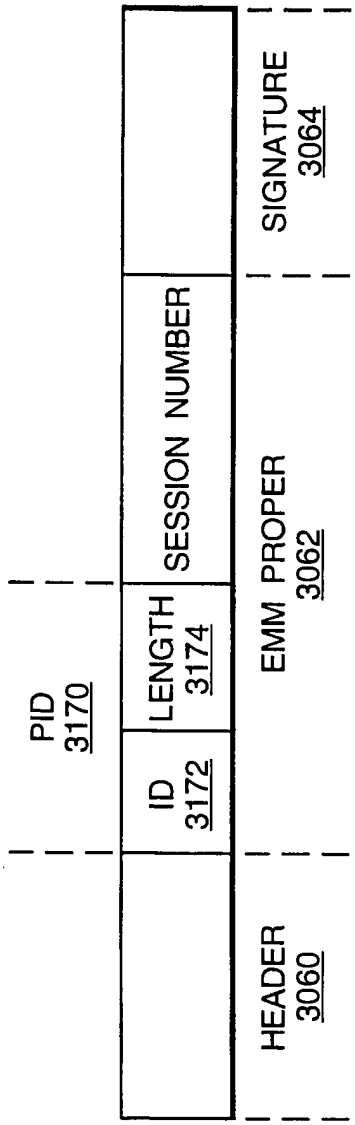


Fig.11.

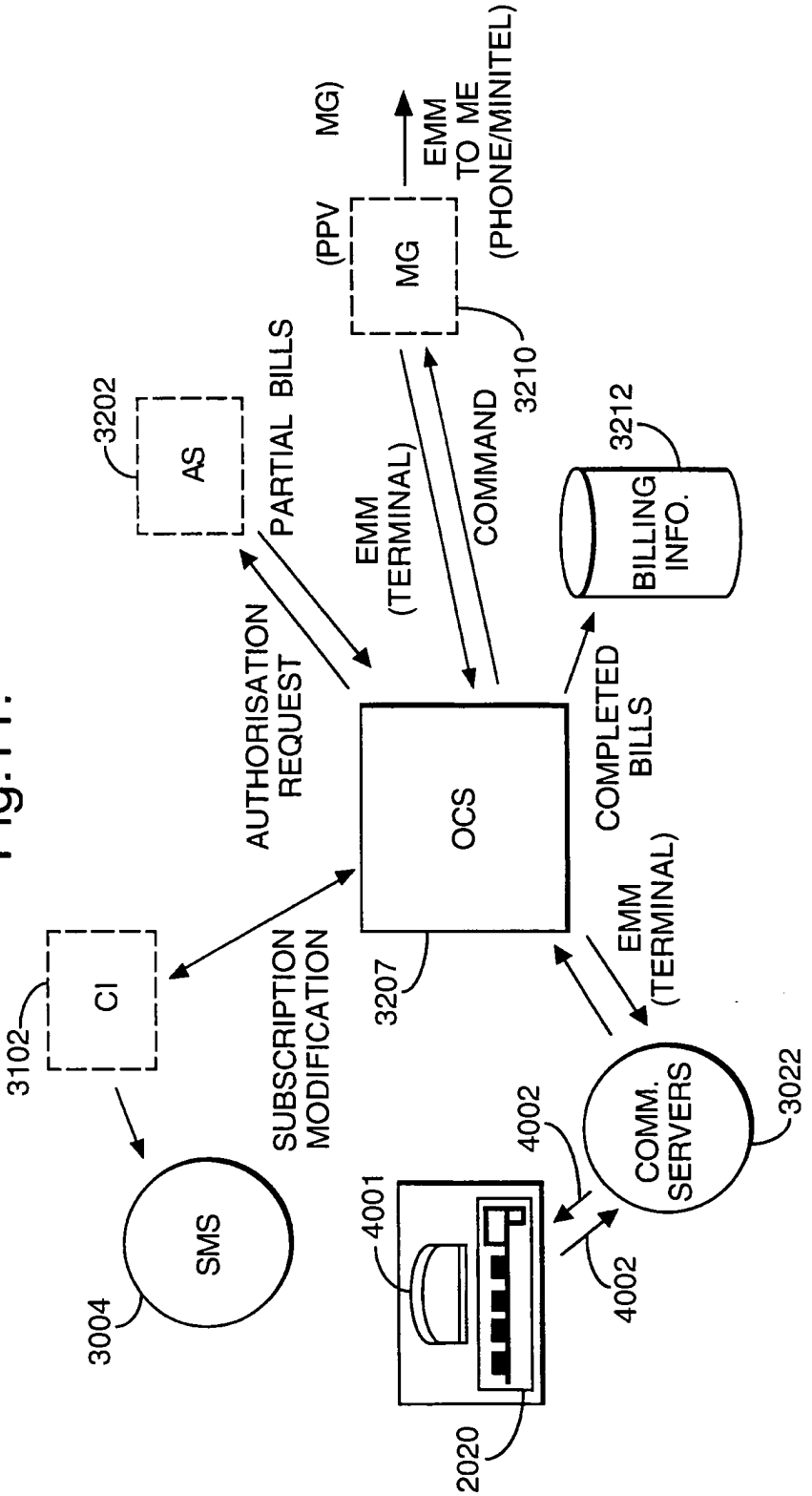


Fig.12.

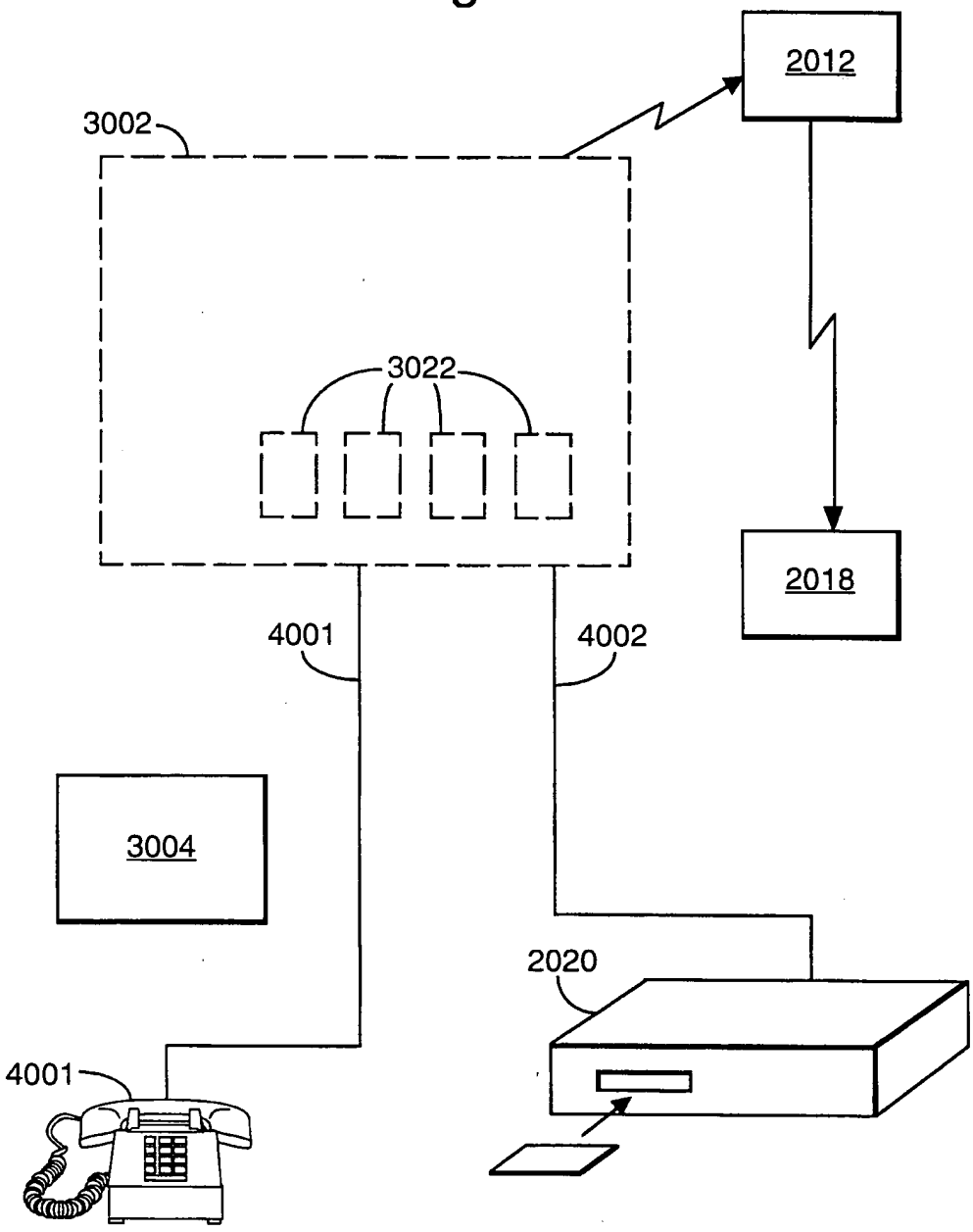


Fig.13.

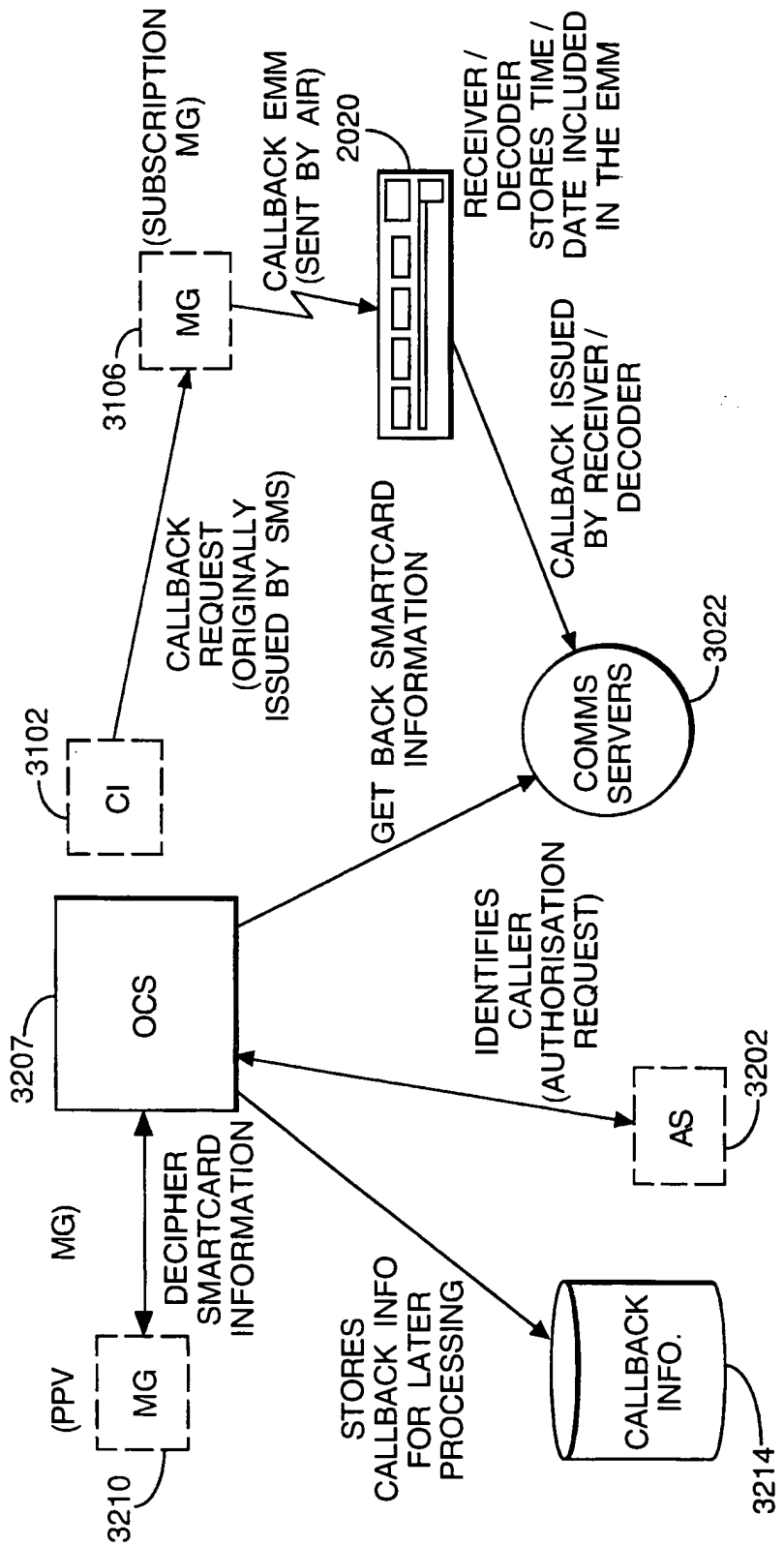


Fig.14.

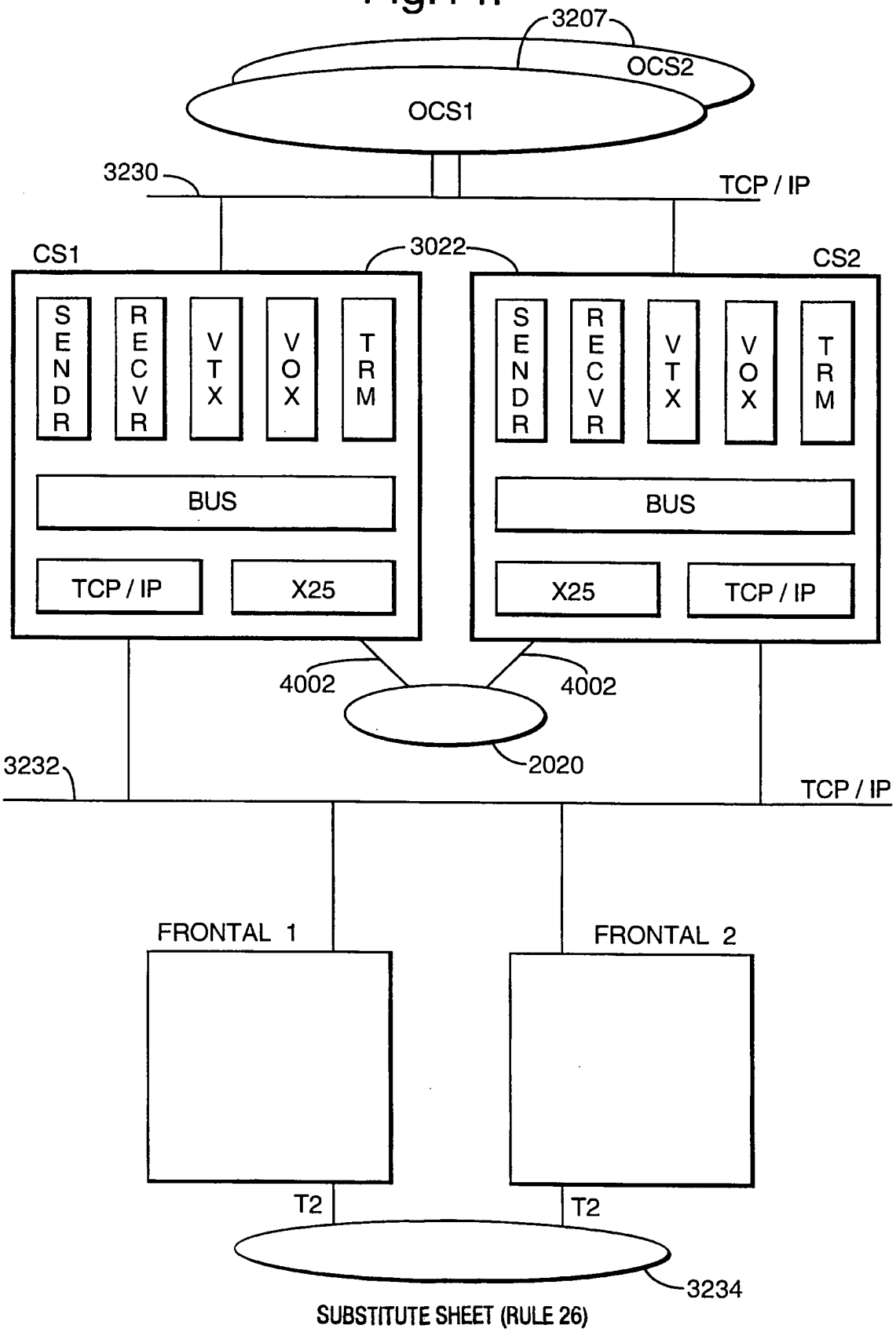
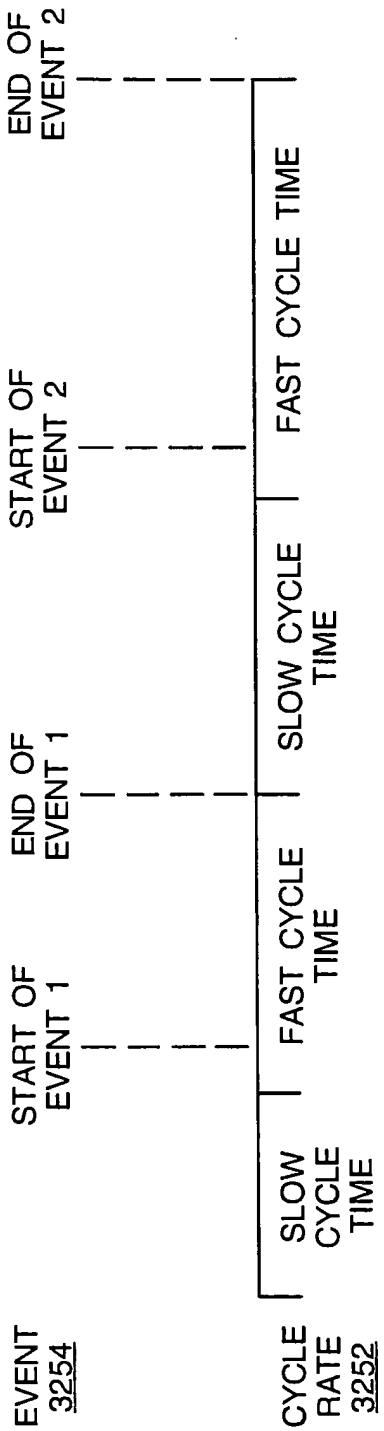


Fig.15.



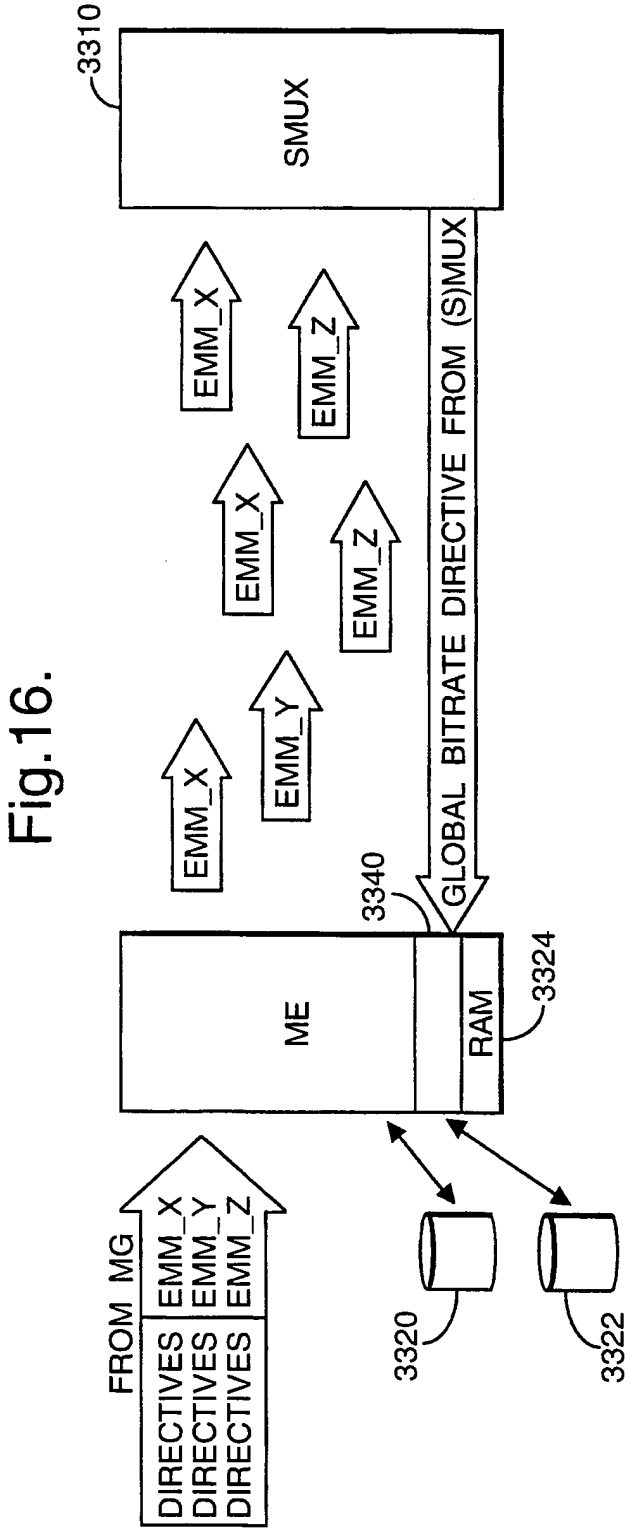


Fig.16.

Fig.17.

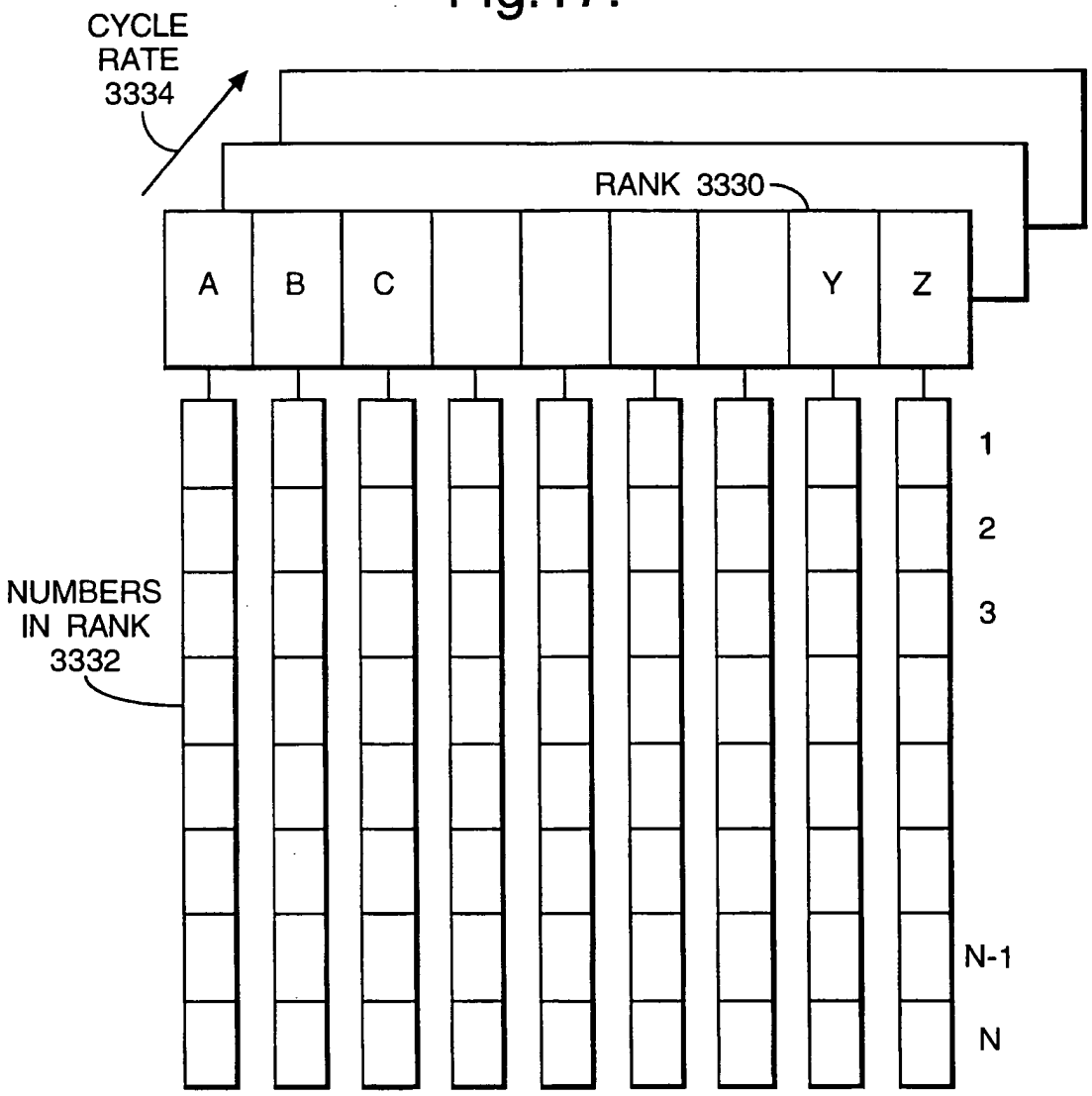


Fig.18.

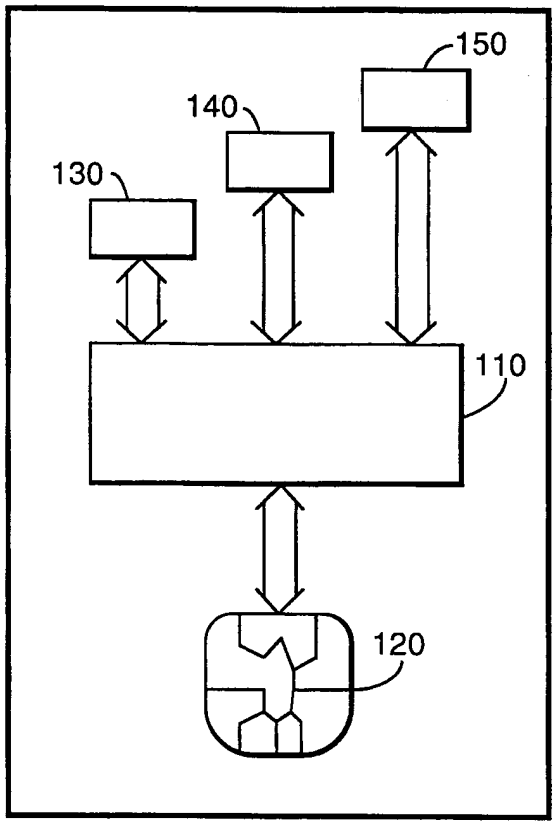


Fig.20.

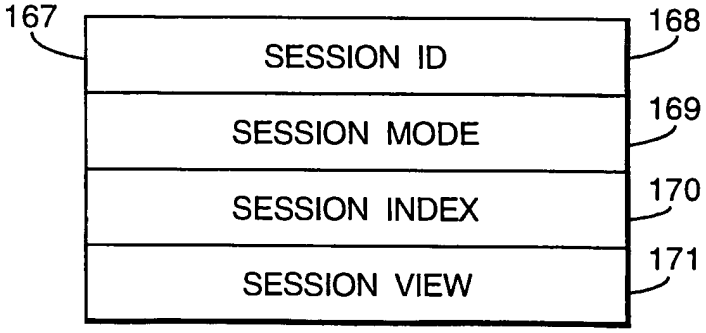
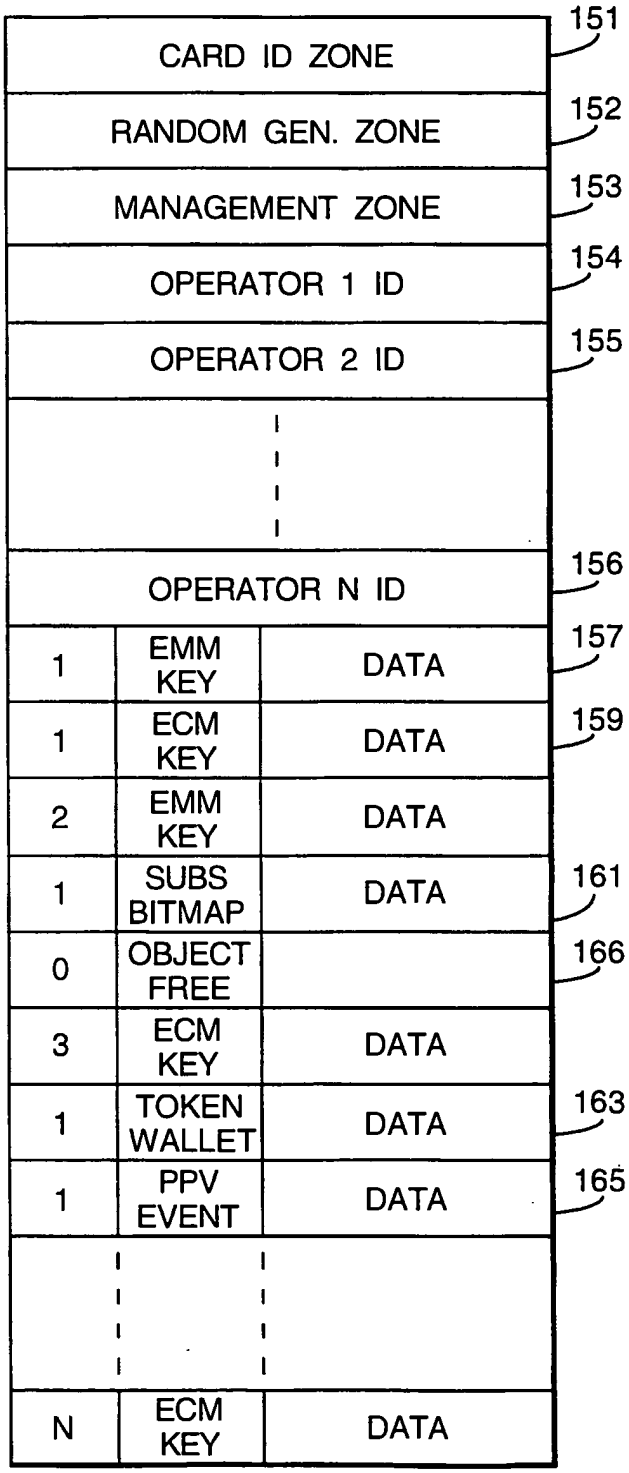


Fig.19.



INTERNATIONAL SEARCH REPORT

Intern: al Application No
PCT/EP 97/02108

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04N7/16 H04N7/167

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 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)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	"FUNCTIONAL MODEL OF A CONDITIONAL ACCESS SYSTEM" EBU REVIEW- TECHNICAL, no. 266, 21 December 1995, pages 64-77, XP000559450 see the whole document ---	1-12, 15-19, 21,22
X	WO 94 14284 A (DISCOVERY COMMUNICAT INC) 23 June 1994 see page 8, line 8 - page 14, line 23 see page 18, line 28 - page 21, line 19 see page 24, line 25 - page 29, line 31 see page 33, line 8 - line 17 see figures 1-11 --- -/--	1-12, 14-17

[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

1

Date of the actual completion of the international search	Date of mailing of the international search report
11 November 1997	18. 11. 97

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	Van der Zaal, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Interna al Application No
PCT/EP 97/02108

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 5 144 663 A (KUDELSKI ANDRE ET AL) 1 September 1992 see column 2, line 5 - line 23 see column 3, line 6 - column 4, line 65 see column 5, line 62 - column 8, line 58 see figures 1-11 -----	1-12

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Interna l Application No
PCT/EP 97/02108

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9414284 A	23-06-94	AU 5732994 A	04-07-94
		AU 5733094 A	04-07-94
		AU 5733194 A	04-07-94
		AU 5733294 A	04-07-94
		AU 5736394 A	04-07-94
		AU 5845894 A	22-06-94
		AU 5869894 A	04-07-94
		CA 2151458 A	23-06-94
		CN 1093211 A	05-10-94
		CN 1090451 A	03-08-94
		CN 1090452 A	03-08-94
		CN 1096151 A	07-12-94
		CN 1090453 A	03-08-94
		CN 1090454 A	03-08-94
		EP 0673578 A	27-09-95
		EP 0673579 A	27-09-95
		EP 0673580 A	27-09-95
		EP 0673581 A	27-09-95
		EP 0673582 A	27-09-95
		EP 0673583 A	27-09-95
		EP 0674824 A	04-10-95
		IL 107908 A	10-01-97
		IL 107909 A	15-04-97
		IL 107910 A	10-06-97
		IL 107912 A	18-02-97
		IL 107913 A	15-04-97
		JP 8510869 T	12-11-96
		JP 8506938 T	23-07-96
		JP 8506939 T	23-07-96
		JP 8506940 T	23-07-96
		JP 8506941 T	23-07-96
		JP 8506942 T	23-07-96
		NZ 259146 A	26-05-97
		NZ 259147 A	26-05-97
		NZ 259148 A	26-11-96
		WO 9413107 A	09-06-94
		WO 9414279 A	23-06-94
		WO 9414280 A	23-06-94
		WO 9414281 A	23-06-94
		WO 9414282 A	23-06-94

INTERNATIONAL SEARCH REPORT

Information on patent family members

Interna al Application No
PCT/EP 97/02108

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9414284 A		WO 9414283 A	23-06-94
		US 5559549 A	24-09-96
		US 5600364 A	04-02-97
		US 5659350 A	19-08-97
US 5144663 A	01-09-92	AU 599646 B	26-07-90
		AU 7157887 A	22-10-87
		DE 3751410 D	24-08-95
		DE 3751410 T	11-04-96
		EP 0243312 A	28-10-87
		EP 0626793 A	30-11-94
		ES 2076931 T	16-11-95
		JP 2610260 B	14-05-97
		JP 63023488 A	30-01-88
		JP 2520217 B	31-07-96
		JP 5244591 A	21-09-93

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 98/45768**

(43) International Publication Date: 15 October 1998 (15.10.98)

(21) International Application Number: PCT/CA98/00241

(22) International Filing Date: 18 March 1998 (18.03.98)

(30) Priority Data:
08/831,696 10 April 1997 (10.04.97) US

(71) Applicant: NORTHERN TELECOM LIMITED [CA/CA]; Station A, P.O. Box 6123, Montreal, Quebec H3C 3J5 (CA).

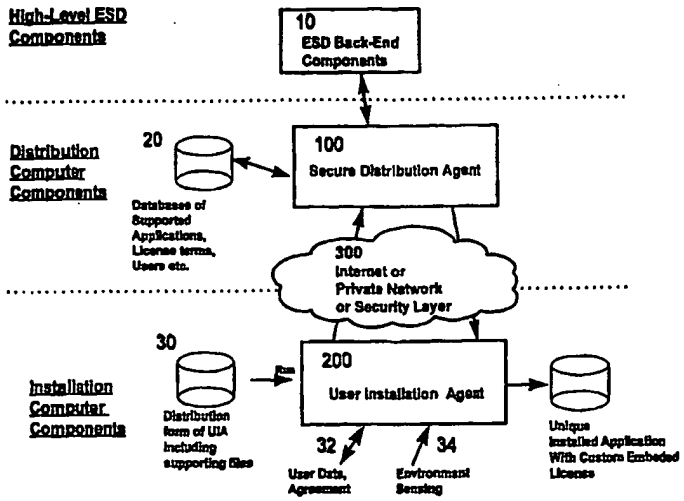
(72) Inventors: LAROSE, Gordon, Edward; 2417 Baseline Road, Ottawa, Ontario K2C 0E3 (CA). ALLAN, David, Ian; 852 Forest Street, Ottawa, Ontario K2B 5P9 (CA).

(74) Agents: MCGRAW, James et al.; Smart & Biggar, 900–55 Metcalfe Street, P.O. Box 2999, Station D, Ottawa, Ontario K1P 5Y6 (CA).

(81) Designated States: AU, CA, CN, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHOD AND SYSTEM FOR NETWORKED INSTALLATION OF UNIQUELY CUSTOMIZED, AUTHENTICABLE, AND TRACEABLE SOFTWARE APPLICATIONS



(57) Abstract

A method to create, distribute and install on an installation computer a uniquely customised instance of a software application that is authenticable and traceable to a particular user. A secure distribution agent resident on a distribution computer collects indentifying information, and calculates a cryptographic signature of the software application and identifying information. The identifying information and cryptographic signature are embedded in the software application by the secure distribution agent. The software application with embedded data is transmitted via a distribution channel to the installation computer. A user installation agent resident on the installation computer manages the installation of the software application with embedded data on the installation computer. Prior to installation, the user installation agent may use the cryptographic signature to verify that the software application, and the indentifying information, are authentic and have not been tampered with.

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		

METHOD AND SYSTEM FOR NETWORKED INSTALLATION OF UNIQUELY CUSTOMIZED, AUTHENTICABLE, AND TRACEABLE SOFTWARE APPLICATIONS

FIELD OF THE INVENTION

This invention relates to a method and system for the electronic distribution and installation to users via a network of software applications that are uniquely customized, authenticable and traceable to the individual user.

BACKGROUND OF THE INVENTION

With the increasing importance and reliance on networked computer environments such as the Internet, Electronic Software Distribution (ESD) is assuming an increased importance as a means of distributing software applications to users. The on-line infrastructures currently in place enable users to purchase and install software applications without the need for physical delivery of shrink-wrapped software. Typically, a software publisher will prepare a master of the software application for electronic distribution. A customer will then go on-line and submit an order to purchase the software application, which will be received and fulfilled by the publisher. The customer will then download the software application and install it to his/her own computer.

A disadvantage of the current on-line infrastructure is that it delivers software applications to users in a form that is identical with those found in retail stores and catalogues. Absent cryptographic protection, users can freely share the distribution form of the software amongst themselves.

Even where cryptographic protection are present, the potential for unauthorized copying is still significant because all the users possess identical copies (necessarily having identical encryption schemes) of a software application. There is in all such cases a single underlying decryption key, and in most cases this key, or an equivalent variant of it, is entered by the user, who can then share it with other users who can use it to obtain unlicensed usage of the program. There exist today bulletin boards and Internet sites devoted to the sharing of such keys, which are visited by persons interested in

obtaining unpaid for usage of programs by applying such keys to copies of the applications they have obtained.

Further, even where more subtle anti-piracy schemes are in place in a software application, it is not uncommon for software "hackers" to produce "crack" programs which can be used to process a freely-distributed, limited functionality version of a program to produce a revised, fully-functional version of the same program which can be used without purchasing a license. Even the most ingenious forms of single-key mass distribution, which might involve input of one-time-only responses to a dynamic challenge to infer the key, are vulnerable to a "crack" which simply causes the application of the "true" universal decryption key. Although such "crack" involves more technical sophistication than sharing of keys as above, the distribution channels and potential effect on the product's revenues are very similar.

In addition, software applications distributed by conventional ESD techniques provide no means to police their own integrity to prevent unauthorized tampering.

Portland Software has produced an electronic software distribution system sold under the trade-mark ZipLock™ that packages software for electronic distribution over the Internet. The ZipLock™ system discloses a system that distributes, from a secure server to a client resident on the user's computer, a standard executable software application that is protected by means of a cryptographic key. Data input by the users is transmitted to the secure server and is used to construct a customized digital licence certificate that is transmitted to the user in a separate computer file. The Ziplock™ system does not provide a mechanism to detect tampering done to the executable software application itself, nor does it provide traceability if the digital licence certificate is not included with an unauthorized redistribution of the software application.

The prior art discloses a number of other systems and methods to protect unauthorized use of software electronically distributed to users. In Choudhury U.S. Pat. No. 5,509,074,

there is disclosed a method of protecting electronically published materials using cryptographic protocols. A first described embodiment requires special purpose hardware to decrypt the document that is transmitted to the user. This eliminates the method from general use with personal computers used by the general public. In a second method, there is no requirement for special purpose hardware. In this method, the publisher modifies the inter-line or inter-word spaces of the document to make each document unique for each user. The unique document is then encrypted and transmitted to the user's computer. Upon receipt of the encrypted document, the user's computer will prompt the user to enter his/her secret key which is used to decrypt the document for viewing. The method disclosed by this reference does not prevent piracy, it only discourages piracy by making the pirated document traceable to the user. In addition, this reference pertains only to data files, not to the protection of executable files of any type.

In Cane U.S. Pat No. 5,416,840, there is disclosed a method and system for protecting computer program distribution in a broadcast medium, such as radio frequency public broadcast or computer network. In this reference, the method involves encrypting at least a portion of a computer program, the user being supplied with a password for use in decrypting the computer program so that the computer program can be installed and used. A unique password is generated and transmitted to the user for subsequent use in decrypting the selected software program contained on the medium. While there is disclosed a method and system for the generation, transmission and use of unique passwords that cannot be shared among different users of the software application, this reference requires the user to own proprietary hardware that eliminates it from general use with personal computer owned by the general public.

In Yuval U.S. Pat No. 5,586,186, there is disclosed a method and system for controlling unauthorized access to software distributed to users. The main components of the system are an encryptor, a user key generator, and a decryptor. The encryptor generates encryption and decryption keys,

encrypts the software using the encryption keys, and stores the encrypted forms of the software of the broadcast medium, such as CD ROM. The user key generator generates a unique key using numeric representations of identifying information supplied by users and the decryption keys. The decryptor is responsible for decrypting the encrypted forms of the software using the identifying information supplied by the user, and the unique user keys. The decryption method disclosed by this reference enables a large number of different but logically similar keys to be used as decryption keys, each of which is unique to a particular user. However, this reference does not disclose a means to customize a software application with user-specific data such that the software application itself can be authenticated. Furthermore, this reference does not prevent piracy by sharing of keys; it only discourages it through traceability of keys.

SUMMARY OF THE INVENTION

The present invention pertains to a method for the electronic distribution of a software application from a distribution computer to an installation computer comprising the steps of receiving at said distribution computer identifying information, embedding said identifying information in said software application at said distribution computer to form an identifiable software application, generating a cryptographic signature for said identifiable software application, embedding said cryptographic signature in said identifiable software application to form an identifiable and authenticable software application, and transferring said identifiable and authenticable software application from said distribution computer to said installation computer.

The method and system of the present invention discloses an on-line software customization, delivery and installation scheme. Instead of distributing a software application to a user that results in the installation of a totally generic, untraceable executable file on the installation computer, the method and system disclosed herein discloses a means to create, distribute and install on an

installation computer a uniquely customised instance of a software application that is authenticable and traceable to a particular user.

The method and system disclosed herein provides for a user installation agent (UIA) resident on an installation computer to establish a connection through a distribution channel to a secure distribution agent (SDA) resident on a distribution computer. The UIA and/or SDA prompt the user to input identifying information that, together with business related information such as licensing terms, etc., is used to create a unique data set that is embedded in the desired software application by the SDA. By the use of a cryptographic hash algorithm, and private/public key cryptography wherein a private key is only known to the SDA, a cryptographic signature of the desired software application and embedded data set is calculated and also embedded into the software application. The software application with embedded data and cryptographic signature is transmitted via a distribution channel to the installation computer where it is installed on the installation computer. Optionally, the installation computer may use the cryptographic signature to verify that neither the software application, nor the embedded data have been tampered with. Public key(s) used to decrypt the cryptographic signatures may be transmitted to the installation computer with the software application, or by any other means, such as e-mail, Internet bulletin boards, etc. Following installation, the embedded data and cryptographic signature are used in a variety of ways, such as to provide a means to trace the software application to the user, to police the continued integrity of the software application, to ensure that license conditions continue to be met, to perform virus checking, or automatic upgrading of the software application itself.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of a system overview showing the various inputs and components of the system and method of the present invention;

Figure 2 is a data flow diagram of the structure and operation of the Secure Distribution Agent employed by the present invention;

Figure 3A is a block diagram showing details of the construction of the aggregate distribution file using a one-step cryptographic process;

Figure 3B is a block diagram showing details of the construction of an aggregate distribution file using a two-step cryptographic process;

Figure 3C is a block diagram showing details of the construction of an aggregate distribution file using a cryptographic process that is a variant of the two-step cryptographic process shown in Figure 3B;

Figure 4 is a block diagram of the structure and operation of the User Installation Agent employed by the present invention;

Figure 5 is a block diagram showing the means of extracting and authenticating embedded data from an installed distribution file;

Figure 6 is a flow chart of a first embodiment of the present invention that authenticates embedded data by means of a common encryption key;

Figure 7 is a flow chart of a second embodiment of the present invention that authenticates embedded data by means of a unique per-user encryption key; and,

Figure 8 is a block diagram showing the various uses of the installed software application delivered to the user by means of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 shows the various inputs and components of the system and method of the present invention. At the top level is shown a representation of the Electronic Software Distribution (ESD) back-end components 10, which include clearinghouses of software, software manufacturers, publishers, credit card servers, etc., all of which interact with a Secure Distribution Agent (SDA) 100 resident on a distribution computer that forms an essential part of the present invention.

The SDA 100 interfaces with these ESD back-end components 10 via the Internet or private computer network to provide payment methods support, loading of software applications from publishers, etc. The exact nature of the ESD back-end components 10 may vary without affecting the method and system of the present invention.

The SDA 100 is comprised of a system of co-operating software programs, which run in a secure environment. The nature of the secure environment is immaterial to the invention as long as it ensures the ability to protect the privacy of user data, authentication of users and possibly other third-parties, and suitable limitations on the operations which can be accessed externally. This environment might or might not be physically separated from an installation computer. The structure and operation of the SDA 100 is more fully described in Figure 2.

One of the inputs to the SDA 100 is a set of databases 20 of supported software applications, license terms, licensed users, etc. The SDA 100 transmits and receives relevant data to/from the databases 20 prior to, and during the operation of the present invention. The exact nature and content of the databases 20 is not an essential feature of the invention.

A distribution channel 300 is illustrated in Figure 1 that can comprise computer networks such as the Internet, or private network, or a security layer as required to maintain security if the SDA 100 were located in close proximity with a user installation agent (UIA) 200. Alternatively, it may contain some combination of these elements. The distribution channel 300 is used to connect the UIA 200 to the SDA 100 (and thus connect the distribution computer to the installation computer) so that information may be exchanged between these two agents, and so that an aggregate distribution file 170 (shown in Figure 2) can be distributed from the SDA 100 to the UIA 200. Though the distribution channel 300 is illustrated between the SDA 100 and the UIA 200, the system of the present

invention does not require that the SDA 100 be physically distant from the UIA 200.

At a user's end is the UIA 200 which is an installation/automatic upgrade software program resident on the installation computer. This program is used to communicate via the distribution channel 300 to the SDA 100, and to perform the required operations, more fully described below, on the installation computer. Though normally one UIA 200 would be required for each supported software application, persons skilled in the art would be familiar with the capability to develop a UIA 200 that would support multiple software applications. Also shown in Figure 1 is a distribution form 30 of the UIA 200, including support files. The nature of the distribution form 30 of the UIA 200 is immaterial to the operation of the present invention. Any of CD ROM, World Wide Web (WWW) download, floppy diskette, etc. could be used.

The UIA 200 accepts data 32 input from the user, such as name, address, payment options, etc., as well as data pertaining to the acceptance of an end-user license. Environment sensing data 34 such as speed of CPU, size of hard disk, speed of modem, etc. may also be input to the UIA 200 for processing. The identifying information processed by the UIA 200 may include any information pertaining to the purchaser, the seller, the installation agent, date, serial number, license specifics, etc. This data may be used for the automatic registration of the desired software application with a publisher or its commercial proxies.

As noted above, the identifying data 32, 34 constitutes identifying information concerning the user, its computer, etc. The identifying data 32, 34 is processed by the UIA 200 and transmitted to the SDA 100 via the distribution channel 300. Of course, it is understood that the identifying information does not necessarily have to be transmitted to the SDA 100 by means of the distribution channel 300. For example, the identifying information may be locally entered into the SDA 100 by an agent using information received verbally, in writing, or in some other non-electronic manner. The SDA 100

combines the identifying data 32, 34 with the data stored in the databases 20 to produce an aggregated distribution file 170 that is uniquely customized, authenticable, and traceable to the user. The aggregate distribution file 170 is transmitted via the distribution channel 300 to the UIA 200. The output from the UIA 200 is a uniquely customized software application 15 (to be referred to below as an "installed aggregate distribution file") installed on the installation computer, with identifying information embedded therein.

Though the description of the present invention implies that the "user" is an individual user of the software application 15 to be installed on a personal computer, persons skilled in the art will appreciate that the present invention would also operate in the context of a networked end-user environment, where the "user" was a network administrator responsible for installing software on a central server for use by a number of end users.

Figure 2 is a data flow diagram of the structure and operation of the SDA 100 employed by the present invention. An original distribution file 130 is shown as an input to a conversion program 110. In the envisioned implementation, the original distribution file 130 is input to the SDA 100 by the databases 20 shown in Figure 1. It is understood that the original distribution file 130 does not necessarily have to be input to the SDA 100 by the databases 20, since the original distribution file 130 may already be resident on the distribution computer containing the SDA 100. The conversion program 110 has, as additional inputs, the data 140 to be embedded in the distribution file 130, and required public/private cryptographic key pairs 150. The embedded data 140 is produced by a user interaction program 120 which interacts with the user through the UIA 200 to receive identifying data 32, 34 (shown in Figure 1) as well as data from the databases 20 of supported software applications, license terms, licensed users, etc.

While the embedded data 140 can be of any form and content, it is anticipated that the embedded data 140 will

contain information enabling the software application 15 to be traceable to an individual user and license transaction. For example, the embedded data 140 can include a unique serial number used to identify the aggregate distribution file 170 to be distributed to the user. The would eliminate serial number fraud that is common in the software industry, whereby current software applications can only perform simple validity checks, which can be fooled by widespread fraudulent re-use of a single valid serial number. The embedded data 140 may take the form of a complete license agreement customized to the individual user, including user name, address, software serial no., license terms, etc. Records of the user information collected by the user interaction program 120 may be kept by the databases 20.

The output of the conversion program 110 is an aggregate distribution file 170 which contains both the contents of the original distribution file 130, the embedded data 140, as well as a cryptographic signature of the embedded data 140 and the original distribution file 130. The aggregate distribution file 170 is then transmitted via the distribution channel 300 to the UIA 200. The UIA 200 then installs the aggregate distribution file 170 on the installation computer. Once the aggregate distribution file 170 is installed, it takes the form of an installed aggregate distribution file 15.

By means of its connection with the UIA 200, the SDA 100 can negotiate arbitrary license terms with the user, display an End-User License Agreement (EULA), confirm acceptance of that agreement, and automatically perform on-line registration of the software based on the already-established identity of the user and the specific license terms. Subject to commercial and legal considerations, an SDA 100 could offer different pricing and license terms, and possibly different executable versions, to users in different countries, for example. In addition, differential pricing based on attributes of the installation computer such as CPU power could be provided.

The SDA 100 does not require intelligence within itself for functions such as establishing that a user's stated address and credit-card number are valid, consistent, and within a given geographical area. Such functions may be undertaken by the high-level ESD components 10 illustrated in Figure 1.

Figure 3A shows the procedure for constructing the aggregate distribution file 170 in greater detail. For the sake of illustration, the original distribution file 130 is assumed to have a structure including header information and different types of internal sections for code, static data and so on, such as a Windows™ 'Portable Executable' (PE) program file. One of ordinary skill in the art can appreciate that the method and system of the present invention can be applied to a number of different file formats. Similarly, the inputs 140, 151 to the conversion program 110, and output 170 from the conversion program 110 are illustrated to be computer files, but they could be in-memory images, streams from other processors, etc.

A typical sequence of steps run by the SDA 100 to construct the aggregate distribution file 170 is described below.

1. The conversion program 110 is run, as a result of the user interaction program 120 having determined that a conversion is required i.e. that a delivery of a particular aggregate distribution file 170 according to the method of the present invention is authorized, and that the required embedded data block 140 has been constructed. All subsequent steps are executed by the conversion program 110 unless otherwise indicated.

The object is to obtain what is often referred to as a "digital signature", or "cryptographic signature" which inherently has two aspects:

(i) By the use of a cryptographic hash algorithm, the production of a cryptographic fingerprint that uniquely corresponds to the data "ed" 130, 140; and

(ii) Protection of that cryptographic fingerprint by encrypting it with a private key, such that the recipient of the cryptographic fingerprint may, by using a public key and the cryptographic algorithm, verify that the data "ed" 130, 140 is intact, without having the ability to generate a new cryptographic fingerprint, and plausibly change the data.

These two steps are essential to realize the advantage of the present invention, since without both steps a third party may intervene and alter data without the recipient being able to detect it. This procedure is to be distinguished from simply encrypting the data "ed" 130, 140, which is a step that is not necessary to the operation of the present invention since there is no way to plausibly alter the data 130, 140 without such alterations being detectable.

2. The input/output logic 111 of the conversion program 110 reads in the desired original distribution file 130, its corresponding cryptographic private key 151, and the data to be embedded 140. Though not required by the conversion program 110, a public key 152 may be passed through in order that it may be added to the aggregate distribution file 130. Utilizing cryptographic hash algorithms 112 and Public-Private key (PPK) encryption algorithms 113, a cryptographic signature 174 is produced. The basic steps of this process are:

2.1 Apply a one-way hash function "hf" to the data "ed" 130, 140 producing a cryptographic fingerprint "edh", that is edh = hf(ed). The requirements for this cryptographic fingerprint are as follows: (i) that it produce a reasonably compact result i.e. size(edh) << size(ed), and preferably a fixed-length result; (ii) that the fingerprint alone cannot be used to ascertain the original data block back i.e. there is no back-hash function "bhf" such that bhf(edh) = ed; (iii) that it be extremely sensitive to changes in "ed"; say, that a single-bit change in "ed" changes on average 50% of the bits in "edh", and (iv) that it is extremely difficult to

construct a false embedded data block "fed" which produces the same fingerprint as "ed", that is hf(ed) = hf(fed). There are a number of algorithms which satisfy these requirements, such as MD5 (Message Digest 5) and SHA (Secure Hash Algorithm). Other algorithms that also meet the above criteria that may be employed by the present invention.

2.2 Encrypt the cryptographic fingerprint "edh" using the private key 151 "prk" and a public/private encryption function "ppef" to produce a cryptographic signature "edf" 174, that is: edf = ppef(prk, edh). The requirements for the encryption function "ppef" are as follows: (i) that it produce a result not substantially larger than its input; (ii) that it effectively protect relatively short data sets, since "edh" will be bytes long rather than kilobytes long; (iii) that is computationally infeasible to use the public key 151 ("puk") and the cryptographic signature "edf" 174, or multiple instances of "edf" 174 (which will be visible on the installation computer) to infer the private key "prk", that is, there is no cracking function "cf" such that: puk = cf(edf,puk); (iv) that there is no conceivable means of replicating the behaviour of "ppef" using "prk" without in fact possessing both "ppef" and "prk". In principle, "ppef" can be inferred from its corresponding decryption function, so "prk" is the important secret in practical terms; (v) that the corresponding public-key decryption function "ppdf" have acceptable performance on a typical installation computer for the pertinent file sizes. Note that if a specific ppef/ppdf is chosen for security reasons and does not yield acceptable performance, the encryption could be applied to only a portion of the selected files and still offer the same benefits; (vi) that it be suitable (preferably, via established cryptanalysis) for specific application in this domain i.e. digital signatures. There are a number of algorithms which might satisfy these requirements, such RSA and those of Rabin and ElGamal. The

careful selection of implementation parameters can help attain required security and performance.

3. The cryptographic signature 174 from step 2.1, and the data to be embedded 140 are inserted into the original distribution file 130 to produce the aggregate distribution file 170. This insertion is not a simple copying of bits into the middle of a file, since it must be compliant to the format requirements of the particular file types. For example, headers may have to be updated to identify the new data etc.

The system and method of the present invention does not require that the embedded data 171 or the cryptographic signature 174 be positioned in any particular manner in the aggregate distribution file 170. What is necessary is that: (i) the software on the installation computer, and the UIA 200 in particular, be able to locate the embedded data 171 and cryptographic signature 174, and (ii) that the aggregate distribution file 170, after it is installed on the installation computer, be able to perform its intended function; for example, if it is an executable file, that it still conform to structural and other platform requirements so it can load and run on the installation computer as it might have before the conversion process. For example, if the file were in a format common to current computers containing an Intel™ microprocessor and running a Microsoft™ Windows™ operating system, the conversion program 110 could inspect the "header" section of the original distribution file 130 to determine where there were sections containing static data so as to avoid sections containing executable code. A static data section would be selected and a suitable location for the embedded data block 171 and cryptographic signature 174 would be found or created. This would be done by, for example, (i) determining that an existing static data block had unused capacity sufficient to add the data, (ii) allocating a new static data block, or (iii) expanding an existing static data block.

The method illustrated in Figure 3A discloses a one-step process wherein cryptographic signature 174 is ascertained

for the original distribution file 130 and the embedded data 140. An optional method, such as that illustrated in Figure 3B, would be to employ a two-step process wherein a cryptographic signature 172 of the embedded data 171 is first produced using the same algorithm described in step 2 above. This embedded data cryptographic signature 172 is then itself embedded into the original distribution file 130. The original distribution file 130, embedded data 171, and embedded data cryptographic signature 172 are then input to the second cryptographic step, wherein an overall cryptographic signature 176 is ascertained using the same algorithm described in step 2 above. The benefit of the two step process is that it augments the capabilities of the system and method of the present invention to authenticate and detect tampering in the software application installed on the installation computer. For example, separate cryptographic public/private key pairs could be provided for the two cryptographic signatures 172, 176. Furthermore, the two-step process allows the embedded data 171 to be extracted and authenticated, even if the original file contents 173a, 173b have been corrupted.

Another alternative is to construct the aggregate distribution file 170 using a variation of the two-step cryptographic process wherein a first cryptographic signature 175 is made of only the original file contents 173a, 173b, and a second cryptographic signature 172 is made of the embedded data 171. This is illustrated in Figure 3C. This scheme has all the advantages of the two-step process illustrated in Figure 3B, and also allows for separate authentication of the embedded data 171 and the original file contents 173a, 173b. This would allow a user to verify that original distribution file 130 provided by the publisher had not been altered by the on-line installation process disclosed by the present invention.

One of ordinary skill in the art will appreciate that any of the cryptographic signatures 172, 174, 175, 176 shown in Figures 3A, 3B and 3C do not have to be produced using the same set of cryptographic public/private key pairs, or even the same

cryptographic algorithms. As well, it is not necessary that the cryptographic signatures 172, 174, 175, 176 be calculated each time an aggregate distribution file 170 is distributed to a user. The SDA 100 could maintain a database of partially-precomputed signatures to speed up the related calculations. The availability of cryptographic hardware support such as RSA co-processors in the installation computer, could be used to attain good responsiveness with maximal security. As well, it is not essential that the aggregate distribution file 170 be constructed in its entirety by the SDA 100. What is necessary is that the aggregate distribution file 170 be derivable in its entirety by the UIA 200.

Figure 4 illustrates the structure and operation of the UIA 200 which consists of a transient installation index 204, a transient installation input fileset 205, and a UIA proper executable software program 203. One skilled in the art will appreciate that there are many ways in which the UIA program 203 could be implemented. Since a significant part of the UIA's 200 functionality involves user interaction and dialog with the SDA 100, options for the implementation of the UIA 200 include either making it an adjunct to a World Wide Web (WWW) browser, or implementing it as a stand-alone program which itself embeds or invokes already-present browser capability on the installation computer.

A typical execution sequence of the UIA 200 is described below:

1. After the UIA program 203 and its support data 204, 205 have been copied onto the installation computer, the user runs the UIA program 203. Note that the UIA program 203 could also have been initiated remotely e.g. sent as an active program within a browser framework by a WWW server. Unless otherwise stated, all subsequent steps are executed by the UIA program 203.

2. The installation index 204 and installation input fileset 205 are read by the installation computer to determine the particular default SDA 100 appropriate for the installation

of the desired software application (known as the "installed aggregate distribution file") 15.

3. The installation computer is examined to determine the probable means of establishing communications with the SDA 100, for example, the presence of TCP/IP network interfaces, modems etc. If no such means are found, the program optionally assists the user to find parameters which will work properly, then fails with a warning. This is because access to the SDA 100 is essential for operation of the invention.

4. The user 1 is prompted with the default data from steps (2) and (3) above, i.e. informed where the UIA program 203 will look for the desired SDA 100, and over what sort of distribution channel 300. The user 1 is then given an opportunity to change this information, either for commercial reasons (e.g. maybe an SDA has changed names or locations), or for technical reasons (e.g. the user does not have working TCP/IP connectivity and wants to use a straight modem link, perhaps via an 800 number.)

5. Via the distribution channel 300, the UIA program 203 establishes contact with SDA 100. If this cannot be done, the UIA program 203, after optionally helping the user determine parameters which will work properly, fails with a warning. While the security of the distribution channel 300 is optional to the operation of this invention, it is expected that the distribution channel 300 will support appropriate protocols to protect the SDA 100 from fraud. A common protocol supporting authentication and privacy, such as Secure Sockets Layer (SSL) is appropriate.

6. The UIA program 203 acts as an intermediary between the user and the SDA 100, enabling the user 1 to establish any legitimate agreement which the SDA 100 supports with respect to the desired installed aggregate distribution file 15. The UIA program 203 also has the ability to determine whether the available system resources of the installation computer meet the requirements of the desired installed aggregate distribution file 15.

There are no technical limits to the variety of options that can be displayed to the user, the questions the user might be asked, the data that might be gathered about the installation computer, etc. Since the SDA 100 is being run throughout the data gathering, data embedding, software distribution and software installation process, the system and method of the present invention can employ various levels of cryptography without the user ever being informed of the cryptographic keys, or any information from which they could be derived. This is unlike other electronic delivery systems which typically require subsequent off-line entry of 'secret keys' or derivates thereof which have been explicitly divulged to the user. Of course, public keys used for the authentication of cryptographic signatures are an exception in that the user may be able to determine them easily, however this is not a security issue since they have no fraudulent application.

7. Assuming that the user 1 meets all the criteria set out by the SDA 100, the SDA 100 will determine a specific set of files that must be transmitted to the UIA 200 to complete installation on the installation computer, notably including at least one aggregate distribution file 170 (shown in Figures 3A-3C). It is immaterial to the system and method of the present invention what is the nature of the agreement entered into between the user 1 and SDA 100, or how it is validated. That is the responsibility of the SDA 100 and its subtending commercial systems 10, if any. Most importantly, the UIA 200 does not and cannot itself decide whether an agreement has been reached between the user and the SDA 100. The UIA 200 does not have, and should not have, access to all the information required to complete the installation, except through interaction with the SDA 100.

8. The SDA 100 transmits an index of the required distribution files to the UIA 200 via the distribution channel 300. The UIA 200 uses this index to augment its own local index 204 forming a complete index for the upcoming installation.

9. The SDA 100 constructs one or more aggregate distribution files 170 and any other files required for the installation, and transmits these files to the UIA 200 via the distribution channel 300.

10. Using its local index and support files 204, 205 the UIA program 203 completes the installation of the installed aggregate distribution file 15 in a manner compliant with the platform of the installation computer. In particular, the UIA 200 installs the aggregate distribution file 170 such that the cryptographic signature 174 and the embedded data 171 are unaffected. Once the aggregate distribution file 170 is installed on the installation computer, it is referred to as an installed aggregate distribution file 15. The UIA program 203 will also perform other system updates 212 as necessary, such as updating the operating system registry (in the case of Windows 95™), and installing any additional application files. Other optional operations, such as leaving an appropriate 'uninstall' utility, may also be involved.

12. If an error should occur, the UIA program 203 may signal the SDA 100 to re-initiate the installation. If no error has occurred, the UIA program 203 signals the SDA 100 that all required data has been received. This could, for example, be used as the trigger signal for the SDA 100 to commit to a financial transaction. Leaving the financial commit to this late part of the process minimizes the probability of the user being charged for a software application which has not been successfully installed, thus reducing one cause of customer frustration.

13. The UIA program 203 deletes any transient files, indices etc. that it might have placed on the installation computer.

14. The UIA program 203 disconnects from the SDA 100 and the distribution channel 300 and exits.

Upon successful completion of the optional authentication procedures described in further detail below, the user can then run the installed aggregate distribution file 15 on the installation computer. It should be understood that

the authentication procedures described below can be done either before or after the installation is completed.

The method and system of the present invention would diminish disputes arising from software which is purchased but not successfully installed. The UIA 200 can detect and warn the user if the installation computer had inadequate resources to run the desired software application, before any financial transaction has been made. Further, the final financial commitment to purchase the software application by the user can be done late in the installation process so that the probability of the financial transaction being successful, but the installation itself failing, would be low.

One of ordinary skill in the art will appreciate that the UIA 200 may be distributed to users in a mass-produced media form containing the original distribution file 130, or a derivative thereof not subject to successful fraudulent re-use through simple copying. In this scenario, the SDA 100 would transmit to the UIA 200 only the incremental information which the UIA 200 would require to construct the aggregate distribution file 170 and complete the installation. Any attempts to pirate the software application can be defeated by ensuring that the distribution form of the UIA 200 contains an incomplete set of executable files, thereby requiring essential data from the SDA 100 to be capable of executing on the installation computer.

Figure 5 illustrates the means of authenticating and extracting user data from an installed aggregate distribution file 15 to verify that neither the original file contents 173a, 173b, nor the embedded data 171 have been tampered with. This step is optional to the operation of the present invention because the installed aggregate distribution file 15 may be run by the user without authentication. It should be understood that the authentication procedures described below can be done either before or after the installation is completed. If authentication is done prior to installation on the installation computer, then the following procedures are

directed by the UIA 203 to the aggregate distribution file 170, instead of the installed aggregate distribution file 15.

The process illustrated in Figure 5 is in relation to an installed aggregate distribution file 15 constructed using the two-step process illustrated in Figure 3B. The principles of authenticating and extracting user data from an installed aggregate distribution file 15 constructed using the one-step cryptographic process illustrated in Figure 3A, or the variant of the two-step process illustrated in Figure 3C, are the same as those described below, with appropriate modifications, depending on the nature of the cryptographic signatures to be compared.

Though a separate authentication and reading program 400 is shown performing the functions of authentication and reading of embedded data 171, a person skilled in the art will appreciate that these functions need not be embodied in such a stand-alone program, and could be incorporated as functions of other programs, such as the UIA 200, a license-checker, a virus-checker, a program loader, a copy program, etc. A typical execution sequence of the authentication and reading program 400 is described below:

1. The authentication and reading program 400 is run, either by a user or by automatic invocation from another program such as the UIA 200. Unless otherwise indicated, the following steps are all executed by authentication and reading program 400.

2. Determine which installed aggregate distribution file 15 to process, either by prompting the user or having this passed as a parameter by the UIA 200. Also determine (if derivable therefrom in the particular implementation, as opposed to being contained in the file itself), which particular public key 152 is applicable to this installed aggregate distribution file 15.

3. Open the installed aggregate distribution file 15 in question and check that it meets the applicable format requirements. For example, a given implementation might support executable (EXE) and dynamic link library (DLL) files

in the 'PE' format for Intel™ processors. If the installed aggregate distribution file 15 fails these basic checks, or is not found, the authentication and reading program 400 fails with an appropriate warning.

4. Examine the file to determine the location of the overall cryptographic signature 176, the embedded data cryptographic signature 172, and the embedded data 171. The installed aggregate distribution file 15 can be formatted in various ways to support this, such as including pointers to these sections in the file header. If applicable in the particular implementation, (i.e. the public key 152 is included in the file as opposed to being otherwise determined the authentication and reading program 400), find and extract the required public key 152.

If any of the above steps fail, the authentication and reading program 400 fails with an appropriate warning.

5. Use the public key 152 to decrypt the overall cryptographic signature 176 into its unencrypted form 176a (the decrypted remote overall fingerprint).

6. Using the same known cryptographic signature algorithm as was employed by the SDA 100, calculate a local version 176b (the locally calculated overall fingerprint) of the overall cryptographic signature. This calculation will necessarily exclude the overall cryptographic signature 176 itself i.e. cover all parts of the installed aggregate distribution file 15 except 176, in order that the locally calculated overall fingerprint 176b will not depend on itself.

7. Compare the locally calculated overall fingerprint 176b to the decrypted remote overall fingerprint 176a. If they differ, the authentication and reading program 400 will fail with a warning that the installed aggregate distribution file 15 has been corrupted. At this point, the UIA 200 may be invoked to contact the SDA 100 to re-acquire the installed aggregate distribution file 15.

8. Extract the embedded data 171 and present it graphically to the user, if the program has been user-invoked,

or pass it in message form to the invoker routine, if software-invoked.

9. Use the public key 152 to decrypt the embedded data cryptographic signature 172 into its unencrypted form 172a (the decrypted remote embedded data fingerprint).

10. Calculate a local version 172b (the locally calculated embedded data fingerprint) of the embedded data cryptographic signature 172 using the same known cryptographic signature algorithm as the SDA 100 used.

11. Compare the locally calculated embedded data fingerprint 172b to the decrypted remote embedded data fingerprint 172a. If they differ, the authentication and reading program 400 will fail with a warning that the embedded data 171 has been corrupted.

A similar procedure of comparison would be followed in respect of the cryptographic signature 174 if the one step process illustrated in Figure 3A had been followed. As well, a similar procedure of comparison would be followed in respect of the original file contents cryptographic signature 175 if the variant of the two-step cryptographic process illustrated in Figure 3C had been undertaken.

Figure 6 is a flow-chart of a summary of the procedures described in relation to Figures 2, 3A, 3B, 3C, 4 and 5. It should be noted that the public key 152 used to authenticate the integrity of the installed aggregate distribution file 15 could be delivered to the UIA 200 by any means since it is not a secret and might be useful for more than one purpose. For example, the public key may be embedded in the aggregate distribution file 170, it may be explicitly sent to the user as a separate file or message, or it may be obtained automatically by the installation computer from a network trusted authority (e.g. Verisign™ Inc.)

Figure 7 is a flow-chart of another set of procedures that may be employed in accordance with the present invention, whereby the original file contents 173a, 173b are encrypted using a unique private key calculated by the SDA 100 for this particular transaction. A record of this unique private key is

kept by the SDA 100, and the corresponding unique public key is transmitted with the aggregate distribution file 170 via the distribution channel 300 to the UIA 200. The UIA 200 will decrypt the aggregate distribution file 170 using the public key. For security reasons, it is preferred that this public key not be permanently stored on the installation computer. Instead, the unique public key would exist only in the computer's Random Access Memory (RAM) for the duration of the installation. This makes usable redistribution of the aggregate distribution file 170 practically impossible.

Although the present invention has been described with reference to the preferred embodiments, one of ordinary skill in the art will recognize that a number of variations, alterations and modifications are possible. In Figure 8 there is an illustration showing the various uses of the installed aggregate distribution file 15. After installation and authentication by the UIA 200, the installed aggregate distribution file 15 may run normally without making use of the embedded data 171 in any way. To ensure licence compliance, the installed aggregate distribution file 15 may also be run in association with a license-enforcement program that verifies that any license terms comprising part of the embedded data 171 are being complied with. The embedded data 171 and cryptographic signatures 172, 174, 175, 176 (depending on the manner in which the aggregate distribution file 170 was constructed) may also be used as an input to a virus checker that may perform an integrity check on the installed aggregate distribution file 15 by using the public key 152 and the same known cryptographic signature algorithm as was employed by the SDA 100. Each time the installed aggregate distribution file 15 is run, the authentication and reading program 400 shown in Figure 5 may also be run, either by itself, or in association with an authenticating loader that would reject tampered files, and would not permit a tampered installed aggregate distribution file 15 to be run. The embedded data 171 may also be used simply for display to the user.

The method and system disclosed herein can also be used to upgrade an installed aggregate distribution file 15 present on an installation computer. In this case, the UIA 200 and the SDA 100 would verify of the license status of the installed aggregate distribution file 15 present on the installation computer, and then invoke the method and system disclosed herein to construct, deliver and install an upgraded version of the installed aggregate distribution file 15 to the installation computer. The capability to invoke the upgrading feature of the present invention could be done at the request of the user, or it could be invoked automatically upon detection by the UIA 200 of the availability of a new version of the original distribution file 130.

The uniqueness of the installed aggregate distribution file 15 can be used to restrict its operation to a specific central processing unit (CPU) on the installation computer. The identification of the CPU for these purposes would be done by the UIA 200 during the stage of gathering data 32, 34 for transmission to the SDA 100.

The SDA 100 and UIA 200 disclosed herein are not restricted to being invoked at the time of installation or upgrading of the installed distribution file 170. For example, in a computer game environment, the SDA 100 and UIA 200 could be invoked when the user reaches a certain point in the game, giving the user the option to purchase additional functionalities or levels for the game.

This disclosure does not presuppose that the UIA 200 does not possess added intelligence to increase the functionality of the present invention. For example, the UIA 200 may possess the intelligence to find and recognize separate Personal Digital Certificates on the installation computer which establish his identity for purposes sufficient to authorize all, or part of, the transaction in question. Such Personal Digital Certificates and their method of application would conform to established standards such as those used by commercial certificate provider Verisign™ Inc. In addition, the UIA 200 could possess the intelligence to find and

recognize digital "coupon" certificates which establish that the user has some specific privilege, such as an entitlement to a specific price for a piece of software, or one which establishes his membership in a specific group, such as a company. In addition, the UIA 200 could locate pre-existing files installed according to the method of the present invention, and examine the embedded data 171. If the UIA 200 determines that there is license information present which may affect the terms of the transaction, or which may indicate a user's likely interest in, for example, an upgrade, the UIA 200 can transmit this information to the SDA 100 so that it can suitably mediate the transaction, advertise an upgrade, etc. A typical example of this would be examining a word-processing application installed in accordance with the present invention to determine that the user is entitled to a free upgrade, which the present invention can then proceed to install.

In another set of variations of the invention, the installed aggregate distribution file 15 is one which uses the principles of Nortel Algorithmic Authorization (NAA), as disclosed in U.S. Patent Application No. 08/674,037 to add robust self-policing of its own integrity. In a first variation, the run-time NAA algorithms, which already have the capability of using the installed aggregate distribution file's 15 own code as an input required for proper operation, and thus of forcing catastrophic failure in the event of tampering, have the scope of this input expanded to include an in-memory copy of one or more of the data items added by the SDA 100, such as the overall cryptographic signature 176.

In a second variation, the "launch stub" component could go further, extracting and decoding the embedded data 171 in the installed aggregate distribution file 15, and comparing the license terms therein (e.g. a specific CPU identified by, say, a certain physical Media Access Control address on a network card) to those it found by examining its current environment. In accordance with the principles of Nortel Algorithmic Authorization, the "launch stub" would not have to "decide" whether to proceed, since such decision-

points are obvious attack points for 'hackers' wishing to defeat security mechanisms. Rather, it could modify data upon which proper program operation depended, in such a way that the program would continue to run properly only if said data corresponded to the proper environment per the license. As for the first variation, the application would have been pre-constructed for the specific instance, as per the patent-pending technology, in such a way that its proper flow of control used input data that was initially 'incorrect' in just such a way as to be 'corrected' only by application of the correct license data, or a simple derivative thereof.

The invention disclosed herein does not necessarily alter the functionality of the installed form of the installed aggregate distribution file 15, it only adds information and authenticability to it. However, there are a number of means by which the behaviour of the installed aggregate distribution file 15 can be mediated in new ways enabled by this invention. In one variation, the SDA 100 would have access to a variety of executable forms for a given program, or to software routines which would dynamically construct variant forms, in order to produce a program which meets particular customer function/cost requirements, and/or which actively binds itself to very specific license terms. For example, in the Microsoft Windows™ environment, different behaviour could be embodied by different Dynamic Link Libraries (DLLS) which could be selectively included.

In another variation, the initial executable form of the program file would have specific functional and license-binding choices built-in, and the SDA 100 would inject (possibly authenticable) data into the executable file which caused it to exhibit the desired behaviour. In yet another variation, the SDA 100 could make use of routines with detailed knowledge of specific program structures in order to add variant code to a pre-existing executable program which was not explicitly designed to accommodate such variation.

The described embodiments of the present invention focus on a single "core" file of a specific file type as the

cornerstone of a software application's installation and security. However the method of the present invention may certainly be applied to more than one file or file type in a particular case. For example, all of the static files associated with an installed software application could receive embedded information such that they were all authenticable and associable with the particular application and installation instance.

We Claim:

1. A method for the electronic distribution of a software application from a distribution computer to an installation computer comprising the steps of:

a. receiving at said distribution computer identifying information;

b. embedding said identifying information in said software application at said distribution computer to form an identifiable software application;

c. generating a cryptographic signature for said identifiable software application;

d. embedding said cryptographic signature in said identifiable software application to form an identifiable and authenticable software application; and

e. transferring said identifiable and authenticable software application from said distribution computer to said installation computer.

2. The method of claim 1, wherein the step of generating a cryptographic signature for said identifiable software application includes the steps of

a. applying a one-way hash function "hf" to the identifiable software application "ed" producing a hash result "edh", where edh = hf(ed); and

b. encrypting the hash result "edh" using a cryptographic key to obtain a cryptographic signature.

3. The method of claim 2, wherein the one-way hash function is generated using any one of a Message Digest 5 (MD5) algorithm and a Secure Hash Algorithm (SHA) algorithm.

4. The method of claim 2 or 3, wherein the step of encrypting the hash result "edh" includes the step of using a public/private encryption function "ppef" and a private encryption key "prk" to encrypt the hash result "edh" to produce a cryptographic signature "edf" where edf = ppef(prk, edh).

5. The method of claim 4, wherein the public/private encryption function "ppef" is generated using any one of an RSA algorithm, a Rabin algorithm and an ElGamal algorithm.

6. The method of claim 1, 2, 3, 4 or 5, wherein the distribution computer and the installation computer are connected by the Internet.

7. The method of claim 1, 2, 3, 4, 5, or 6, wherein the identifying information received at said distribution computer is transmitted from said installation computer.

8. A method of receiving electronically at an installation computer a software application distributed from a distribution computer comprising the steps of:

a. receiving an identifiable and authenticable software application from the distribution computer, the identifiable and authenticable software application having embedded therein the identifying information and a cryptographic signature of the identifiable and authenticable software application; and

b. installing the identifiable and authenticable software application at the installation computer.

9. The method of claim 8, wherein prior to the step of receiving an identifiable and authenticable software application from the distribution computer, the installation computer transmits identifying information to the distribution computer.

10. The method of claim 8 or 9, wherein prior to the step of installing the identifiable and authenticable software application, the installation computer authenticates the integrity of the software application.

11. The method of claim 10, wherein the installation computer uses the cryptographic signature to authenticate the integrity of the software application.

12. A method for the electronic distribution of a software application from a distribution computer to an installation computer comprising the steps of:

a. receiving identifying information at said distribution computer;

b. embedding said identifying information in said software application at said distribution computer to form an identifiable software application;

c. generating a cryptographic signature for said identifiable software application;

d. embedding said cryptographic signature in said identifiable software application to form an identifiable and authenticable software application;

e. transferring said identifiable and authenticable software application from said distribution computer to said installation computer; and

f. installing said identifiable and authenticable software application at said installation computer.

13. The method of claim 12, wherein the distribution computer and the installation computer are connected by the Internet.

14. The method of claim 12 or 13, wherein the identifying information received at said distribution computer is transmitted from said installation computer.

15. A software distribution computer for distributing an identifiable and authenticable software application to a user comprising:

a. a communications link between said software distribution computer and said user;

b. a storage device for storing a software application for distribution;

c. a communications interface in communication with said link, for receiving identification data from said user, and for transferring said identifiable and authenticable software application to said user;

d. means for embedding identification data received from said installation computer in said software application to form an identifiable software application;

e. means for generating a cryptographic signature for said identifiable software application; and

f. means for embedding said cryptographic signature in said identifiable software application to form said identifiable and authenticable software application.

16. A software installation computer for receiving an identifiable and authenticable software application distributed by a distribution computer:

a. a communications link between said software installation computer and said software distribution computer;

b. a storage device for storing identification data, and for storing an installed software application;

c. a computer communications interface in communication with said link, for transferring said identification data, and for receiving said identifiable and authenticable software application, the identifiable and authenticable software application having embedded therein the identification data, and a cryptographic signature of the identifiable and authenticable software application;

d. means for installing said identifiable and authenticable software application on said computer storage device.

17. A software distribution computer for distributing an identifiable and authenticable software application from a distribution computer to an installation computer comprising:

a distribution computer;

an installation computer;

a communications link between said installation computer and distribution computer;

said distribution computer comprising:

a. distribution computer storage device for storing a software application for distribution;

b. a distribution computer communications interface in communication with said link, for transferring an identifiable and authenticable software application to said installation computer and for receiving identification data from said installation computer;

c. means for embedding identification data received from said installation computer in said software application to form an identifiable software application;

d. means for generating a cryptographic signature for said identifiable software application; and

e. means for embedding said cryptographic signature in said identifiable software application to form an identifiable and authenticable software application;

said installation computer comprising:

a. an installation computer storage device for storing said identification data, and for storing an installed software application;

b. an installation computer communications interface in communication with said link, for transferring said identification data to said distribution computer and for receiving said identifiable and authenticable software application from said distribution computer; and,

d. means for installing said software application on said installation computer storage device.

FIG. 1

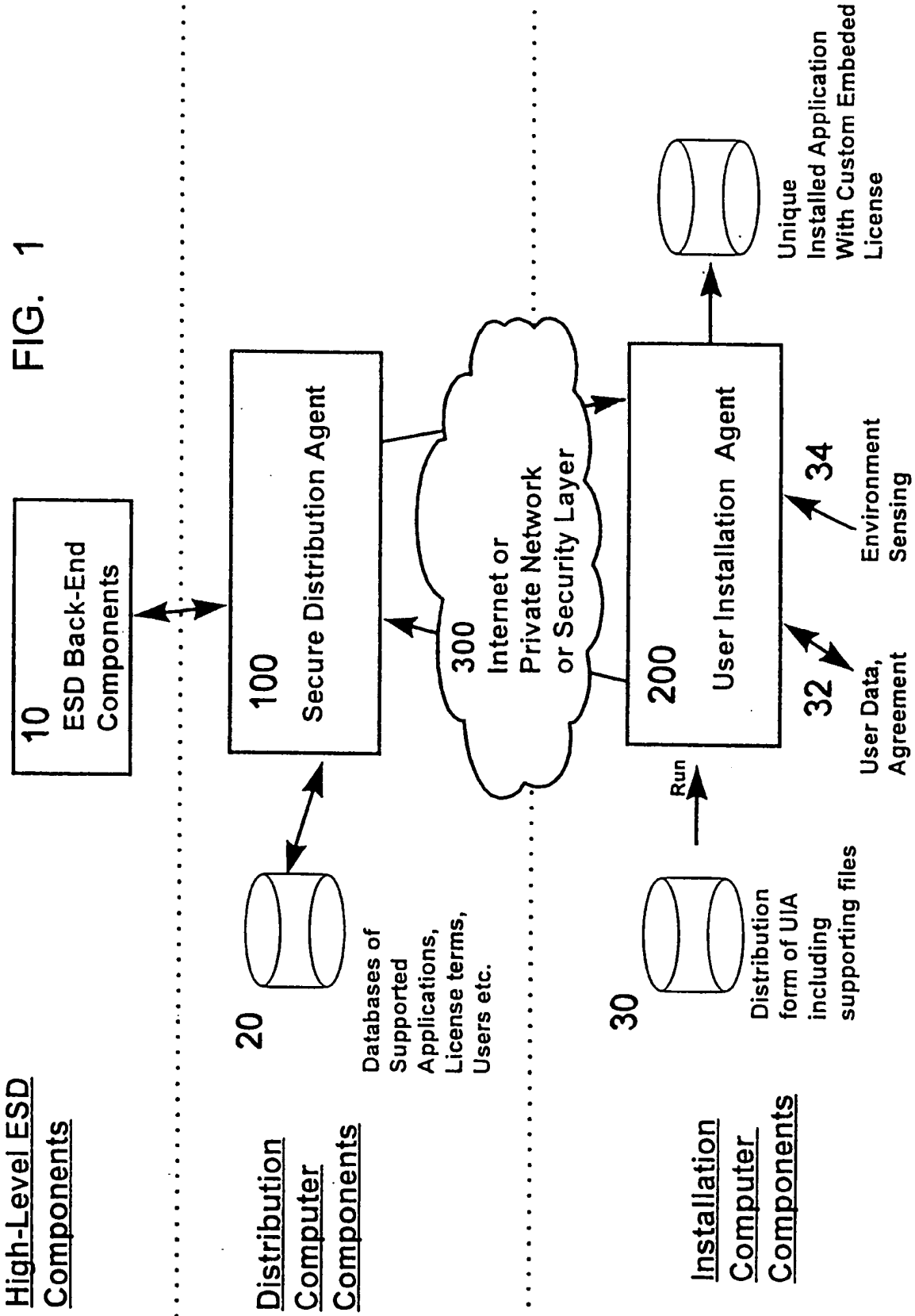
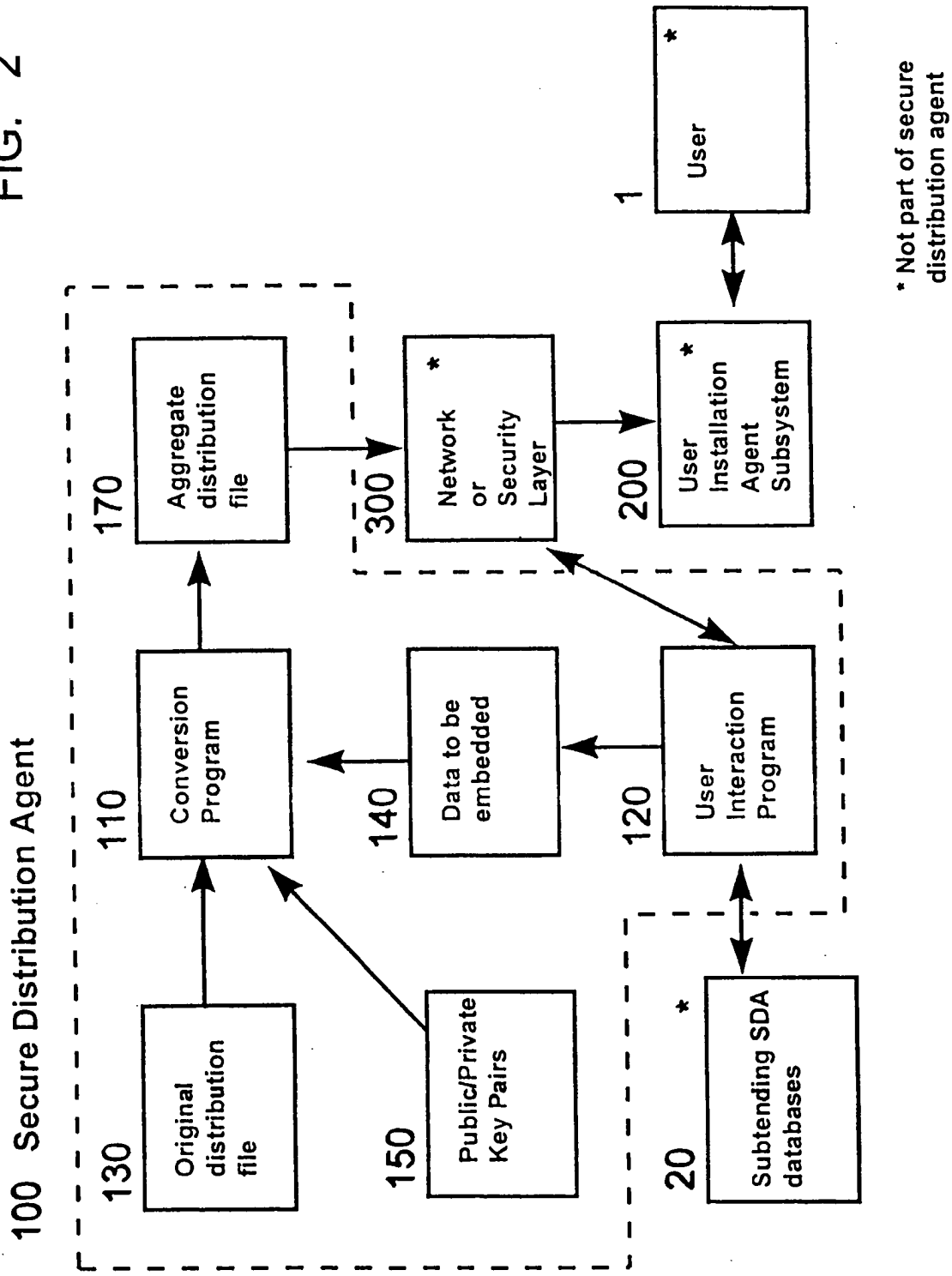

High-Level ESD Components
Distribution Computer Components
Installation Computer Components
10 ESD Back-End Components
100 Secure Distribution Agent
300 Internet or Private Network or Security Layer
200 User Installation Agent
20
Databases of Supported Applications, License terms, Users etc.
30
Distribution form of UIA including supporting files
Run
32
User Data, Agreement
34
Environment Sensing
Unique Installed Application With Custom Embeded License

FIG. 2


100 Secure Distribution Agent
1
User
*
* Not part of secure distribution agent
170
Aggregate distribution file
300
Network or Security Layer
200
User Installation Agent Subsystem
110
Conversion Program
140
Data to be embedded
120
User Interaction Program
130
Original distribution file
150
Public/Private Key Pairs
20
Subtending SDA databases

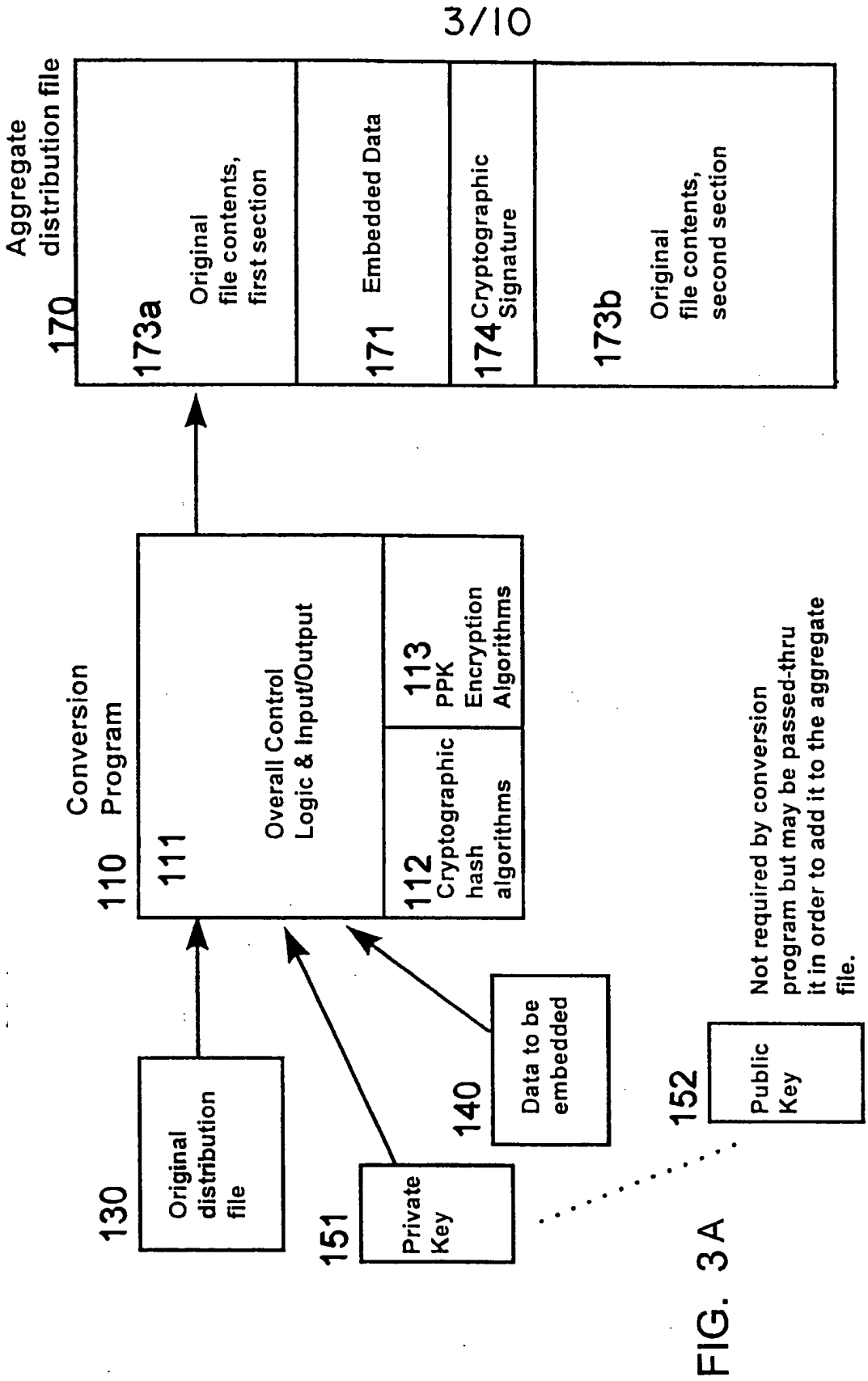

Aggregate distribution file
170
173a
Original file contents, first section
171
Embedded Data
174
Cryptographic Signature
173b
Original file contents, second section
Conversion Program
110
111
Overall Control Logic & Input/Output
112
Cryptographic hash algorithms
113
PPK Encryption Algorithms
130
Original distribution file
151
Private Key
140
Data to be embedded
152
Public Key
Not required by conversion program but may be passed-thru it in order to add it to the aggregate file.

FIG. 3A

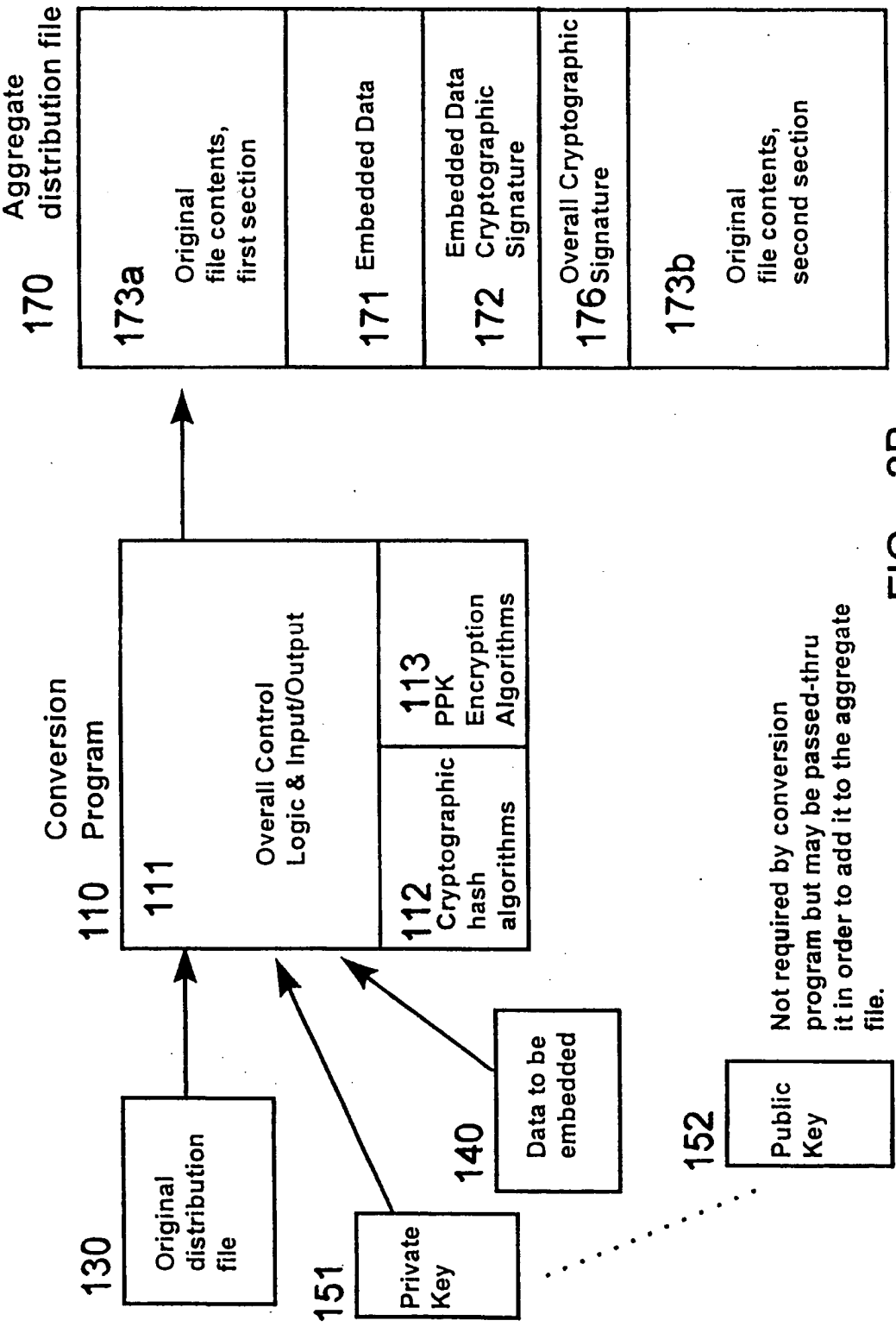

Aggregate distribution file
170
173a
Original file contents, first section
171 Embedded Data
172 Embedded Data Cryptographic Signature
176 Overall Cryptographic Signature
173b
Original file contents, second section
Conversion Program
110
111
Overall Control Logic & Input/Output
112 Cryptographic hash algorithms
113 PPK Encryption Algorithms
130
Original distribution file
151
Private Key
140
Data to be embedded
152
Public Key
Not required by conversion program but may be passed-thru it in order to add it to the aggregate file.

FIG. 3B

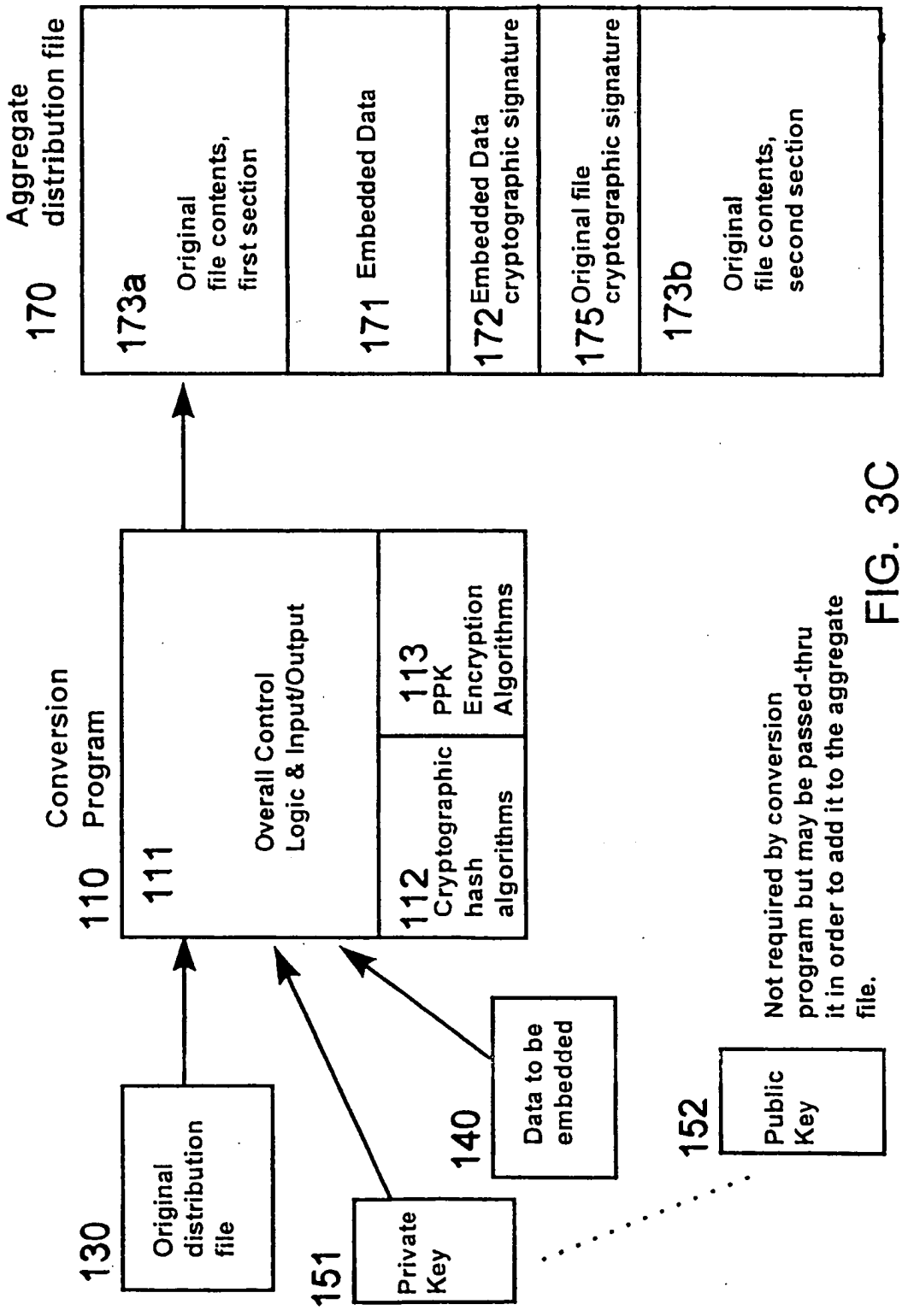

Aggregate distribution file
170
173a
Original file contents, first section
171 Embedded Data
172 Embedded Data cryptographic signature
175 Original file cryptographic signature
173b
Original file contents, second section
Conversion Program
110
111
Overall Control Logic & Input/Output
112 Cryptographic hash algorithms
113 PPK Encryption Algorithms
130
Original distribution file
151
Private Key
140
Data to be embedded
152
Public Key
Not required by conversion program but may be passed-thru it in order to add it to the aggregate file.

FIG. 3C

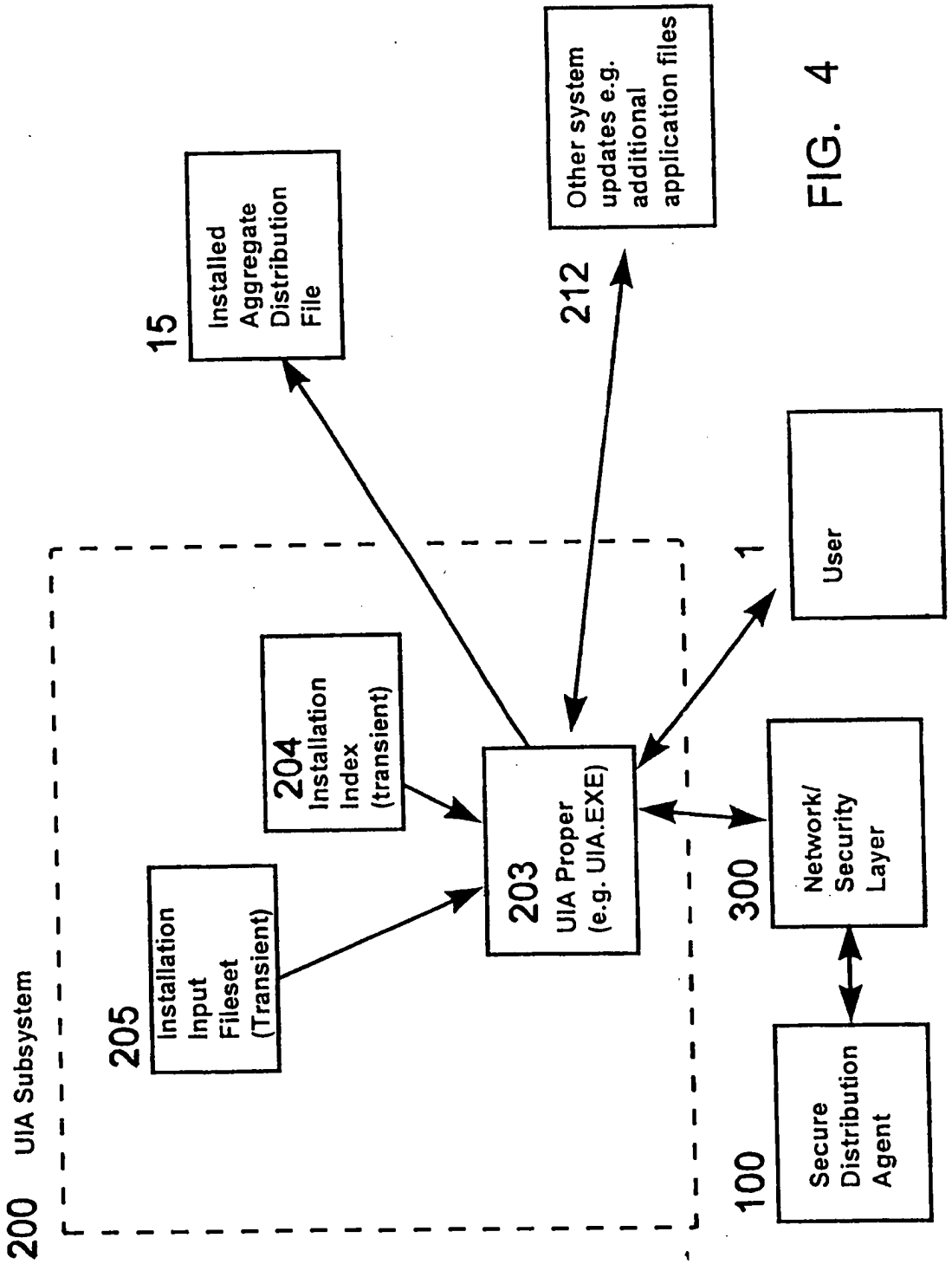

200 UIA Subsystem
205 Installation Input Fileset (Transient)
204 Installation Index (transient)
203 UIA Proper (e.g. UIA.EXE)
15 Installed Aggregate Distribution File
212
Other system updates e.g. additional application files
1
User
300 Network/ Security Layer
100 Secure Distribution Agent

FIG. 4

FIG. 5

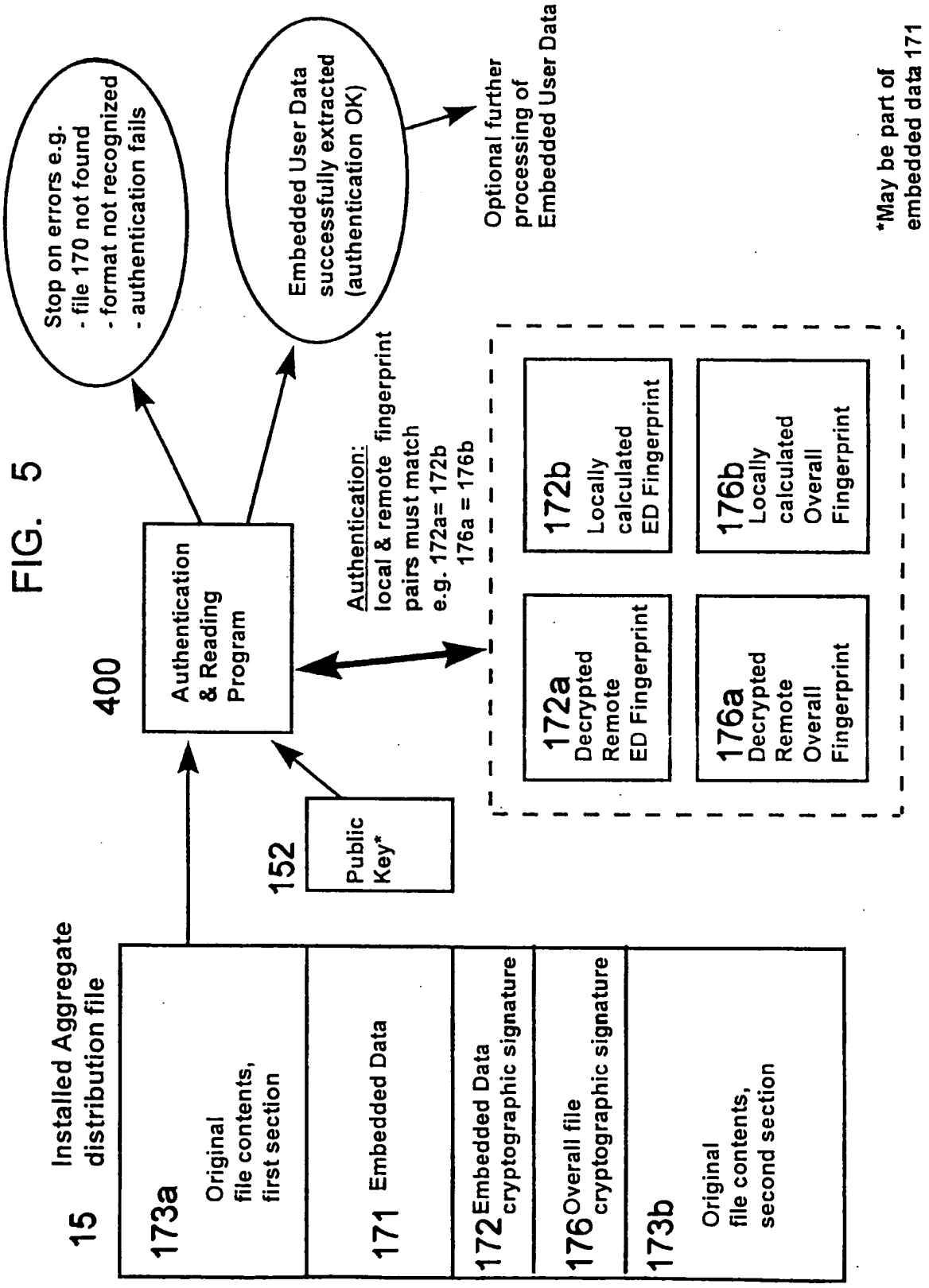

15 Installed Aggregate distribution file
173a Original file contents, first section
171 Embedded Data
172 Embedded Data cryptographic signature
176 Overall file cryptographic signature
173b Original file contents, second section
152 Public Key*
400 Authentication & Reading Program
Authentication: local & remote fingerprint pairs must match e.g. 172a= 172b 176a = 176b
172a Decrypted Remote ED Fingerprint
172b Locally calculated ED Fingerprint
176a Decrypted Remote Overall Fingerprint
176b Locally calculated Overall Fingerprint
Stop on errors e.g. - file 170 not found - format not recognized - authentication fails
Embedded User Data successfully extracted (authentication OK)
Optional further processing of Embedded User Data
*May be part of embedded data 171

FIG. 6

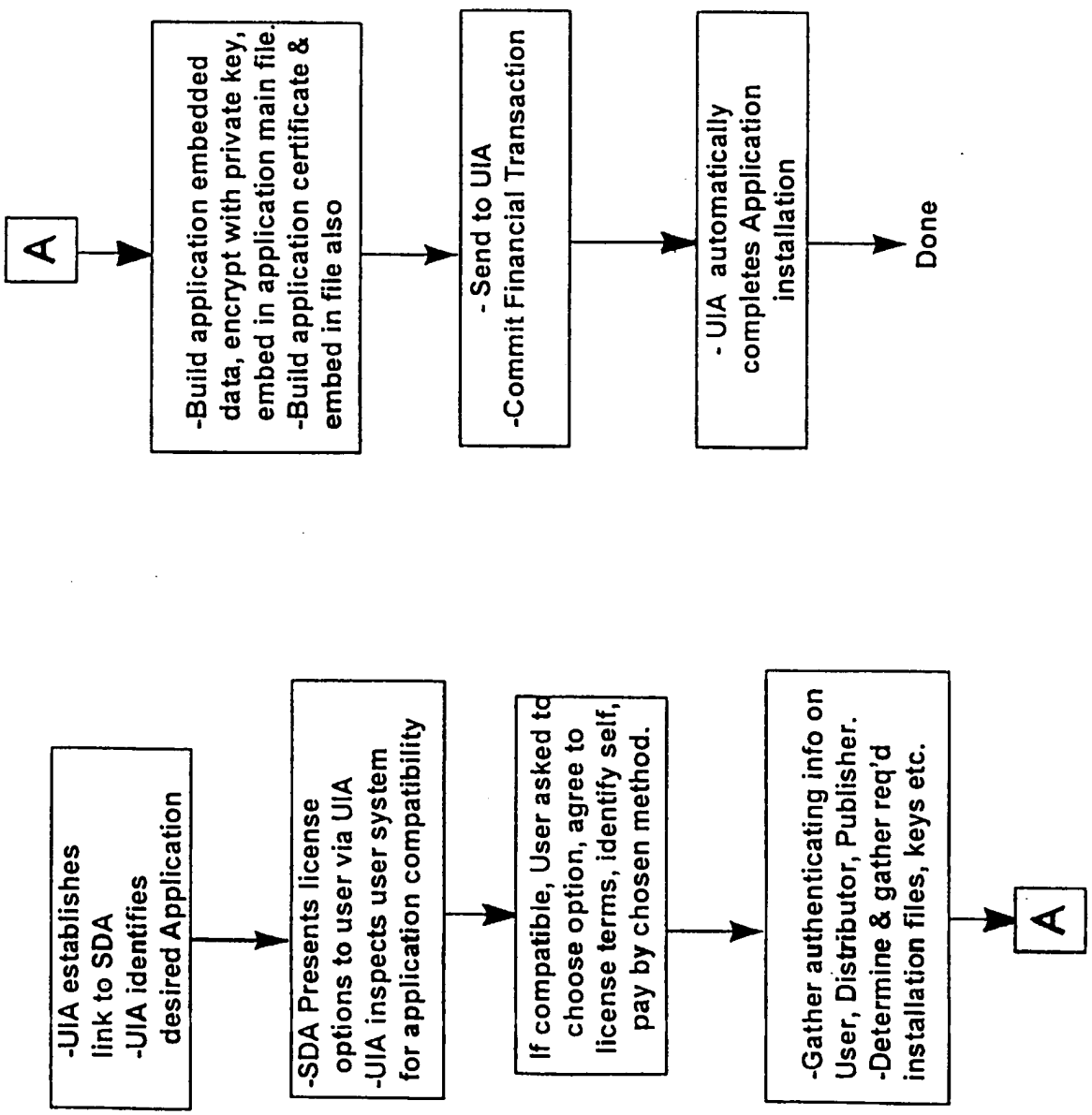

-UIA establishes link to SDA
-UIA identifies desired Application
-SDA Presents license options to user via UIA
-UIA inspects user system for application compatibility
If compatible, User asked to choose option, agree to license terms, identify self, pay by chosen method.
-Gather authenticating info on User, Distributor, Publisher.
-Determine & gather req'd installation files, keys etc.
A
A
-Build application embedded data, encrypt with private key, embed in application main file.
-Build application certificate & embed in file also
- Send to UIA
-Commit Financial Transaction
- UIA automatically completes Application installation
Done

FIG. 7

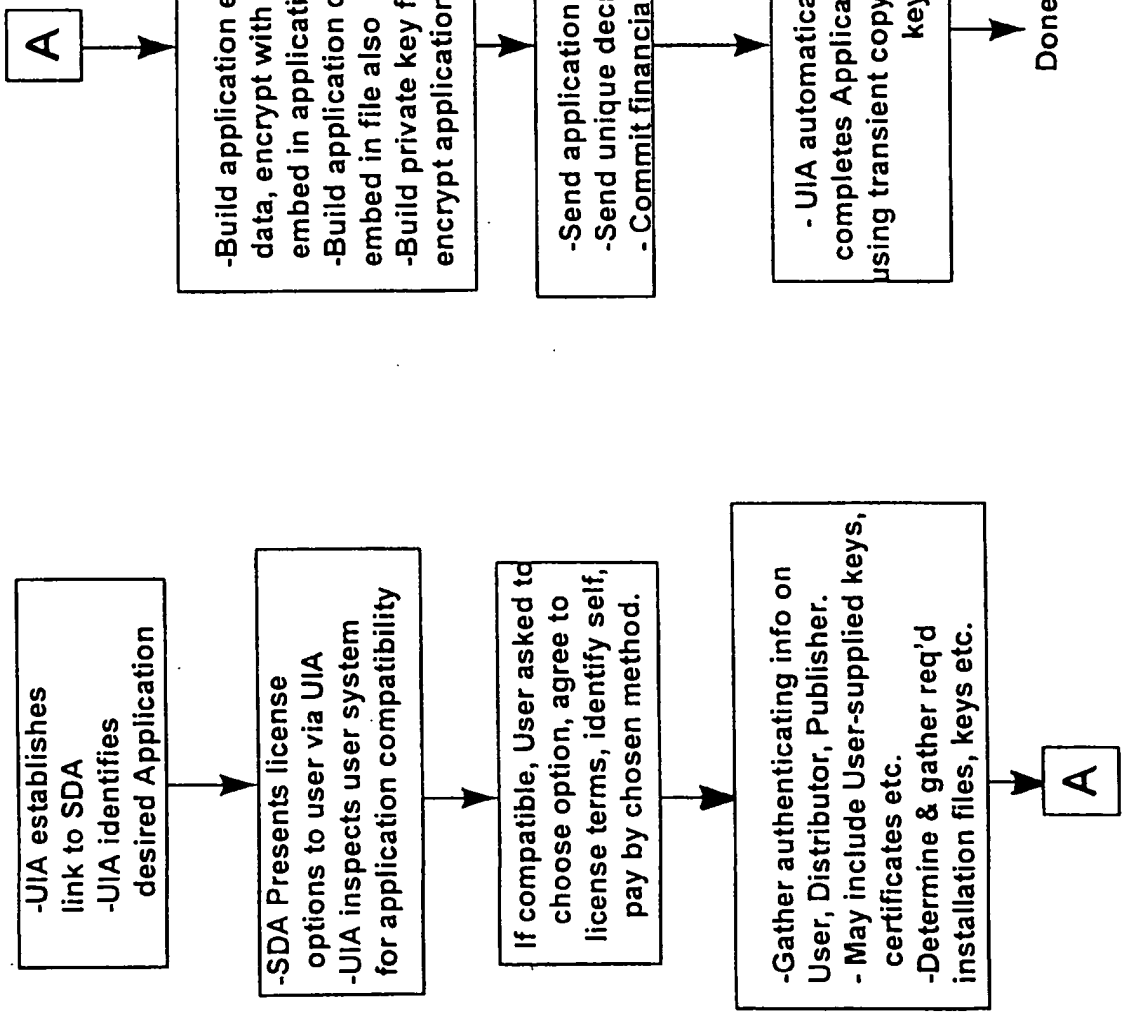
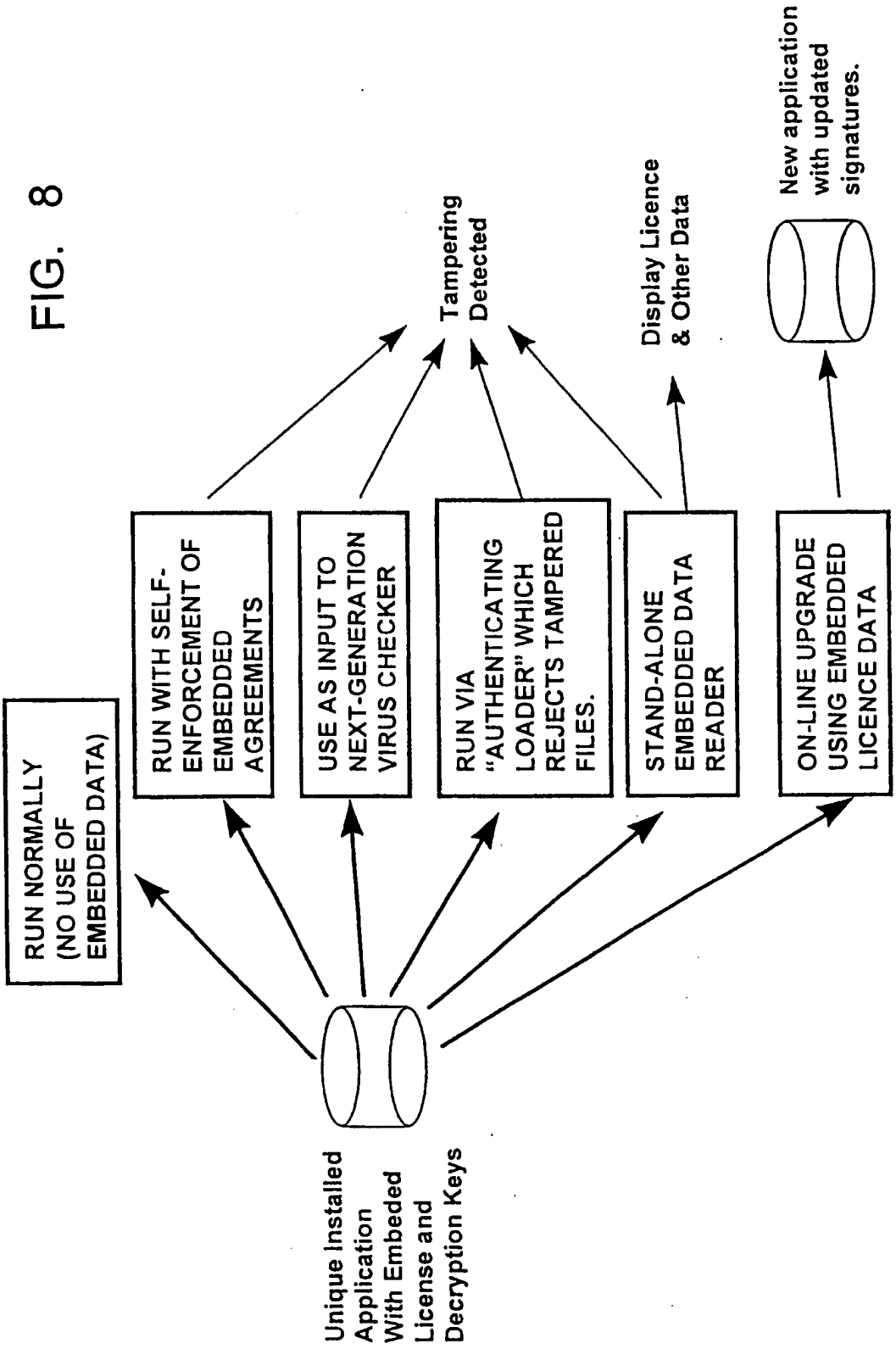

-UIA establishes link to SDA
-UIA identifies desired Application
-SDA Presents license options to user via UIA
-UIA inspects user system for application compatibility
If compatible, User asked to choose option, agree to license terms, identify self, pay by chosen method.
-Gather authenticating info on User, Distributor, Publisher.
- May include User-supplied keys, certificates etc.
-Determine & gather req'd installation files, keys etc.
A
A
-Build application embedded data, encrypt with private key, embed in application main file.
-Build application certificate & embed in file also
-Build private key for this user & encrypt application with this key.
-Send application to UIA
-Send unique decryption key
- Commit financial Transaction
- UIA automatically completes completes Application installation using transient copy of unique Public key
Done

FIG. 8


Unique Installed Application With Embeded License and Decryption Keys
RUN NORMALLY (NO USE OF EMBEDDED DATA)
RUN WITH SELF-ENFORCEMENT OF EMBEDDED AGREEMENTS
USE AS INPUT TO NEXT-GENERATION VIRUS CHECKER
RUN VIA "AUTHENTICATING LOADER" WHICH REJECTS TAMPERED FILES.
STAND-ALONE EMBEDDED DATA READER
ON-LINE UPGRADE USING EMBEDDED LICENCE DATA
Tampering Detected
Display Licence & Other Data
New application with updated signatures.

INTERNATIONAL SEARCH REPORT

Int :ional Application No
PCT/CA 98/00241

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.
X	EP 0 686 906 A (SUN MICROSYSTEMS INC) 13 December 1995	1-6,8, 10-13, 15-17
A	see column 4, line 1 - column 5, line 5; figures 4,6A,6B	7,9,14

A	LEIN HARN ET AL: "A SOFTWARE AUTHENTICATION SYSTEM FOR INFORMATION INTEGRITY" COMPUTERS & SECURITY INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, vol. 11, no. 8, 1 December 1992, pages 747-752, XP000332279 see page 750, left-hand column, paragraph 2 see page 750, left-hand column, paragraph 8 - right-hand column, paragraph 5	1-17

	-/--	

[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
8 July 1998	15/07/1998

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	Moens, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int onal Application No
PCT/CA 98/00241

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 5 509 074 A (CHOUDHURY ABHIJIT K ET AL) 16 April 1996 cited in the application see abstract	1-17
A	EP 0 717 337 A (IBM) 19 June 1996 see column 1, line 1 - column 2, line 40	1,8,12, 15-17

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int ional Application No
PCT/CA 98/00241

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0686906 A	13-12-1995	US 5724425 A	03-03-1998
		JP 8166879 A	25-06-1996
US 5509074 A	16-04-1996	CA 2137065 A	28-07-1995
		EP 0665486 A	02-08-1995
		JP 7239828 A	12-09-1995
EP 0717337 A	19-06-1996	JP 8221268 A	30-08-1996
		US 5745678 A	28-04-1998

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04L 9/08, G07C 9/00 — A1

(11) International Publication Number: **WO 97/25800**

(43) International Publication Date: 17 July 1997 (17.07.97)

(21) International Application Number: PCT/CA96/00847

(22) International Filing Date: 17 December 1996 (17.12.96)

(30) Priority Data:
08/584,375 8 January 1996 (08.01.96) US

(71) Applicant: MYTEC TECHNOLOGIES INC. [CA/CA]; Suite 430, 10 Gateway Boulevard, Don Mills, Ontario M3C 3A1 (CA).

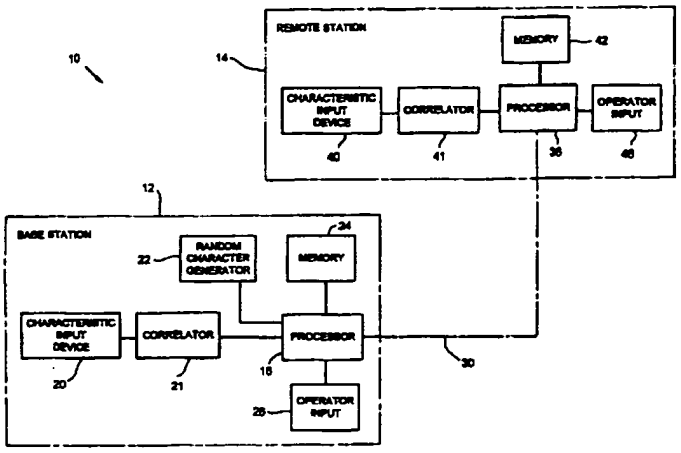
(72) Inventors: TOMKO, George, J.; Mytec Technologies Inc., Suite 430, 10 Gateway Boulevard, Don Mills, Ontario M3C 3A1 (CA). STOIANOV, Alexei; Mytec Technologies Inc., Suite 430, 10 Gateway Boulevard, Don Mills, Ontario M3C 3A1 (CA).

(74) Agent: FAGGETTER, Ronald, D.; Fetherstonhaugh & Co., Suite 2300, 439 University Avenue, P.O. Box 39, Station P, Toronto, Ontario M5S 2S6 (CA).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, 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, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, 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.

(54) Title: METHOD FOR SECURE DATA TRANSMISSION BETWEEN REMOTE STATIONS



(57) Abstract

A method for permitting the secure handling of data between two remote stations firstly involves the generation of an encrypted decryption key which is based on a fingerprint information signal from a user of a first station, a fingerprint information signal from a user of a second station, and a key representing function derived from a random key. The encrypted decryption key is of the type with the property that when it is written to a spatial light modulator (SLM) of an optical correlator, the output of the correlator is similar when input with either one of the fingerprint information signals. The encrypted key is then stored at both stations. Thereafter a message encrypted with the key may be decrypted at either station by retrieving the encrypted key, writing the encrypted key to a filter of an optical correlator, inputting one of the fingerprint information signals to the correlator in order to allow recovery of the decryption key, and applying the decryption key to the encrypted message.

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.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

METHOD FOR SECURE DATA TRANSMISSION BETWEEN REMOTE STATIONS

Background of the Invention

1. Field of the Invention

The present invention provides a method for permitting the secure passing of data between two remote stations.

2. Background of the Invention

While use of the internet has increased rapidly, concerns for the privacy and security of data transferred over the internet have remained. The present invention seeks to provide a method for permitting the secure handling of data between remote stations, such as remote computers hooked to the internet.

Summary of the Invention

In accordance with the present invention, there is provided a method for permitting the secure passing of data between two remote stations, comprising the steps of: obtaining from a user of a first of two remote stations, a first characteristic information signal; obtaining from a user of a second of two remote stations, a second characteristic information signal; generating a sequence of random characters to obtain a random key; obtaining a key function which represents said key; obtaining a Fourier transform of said key representing function; obtaining at least one encrypted version of said key based on said Fourier transform of said key representing function, and a least one of said first characteristic information signal and said second characteristic information signal such that said key may be recovered by writing said at least one encrypted version of said encrypted key to a spatial light modulator (SLM) of an optic correlator and inputting either one of said first characteristic information signal and said second characteristic information signal

to said optic correlator; storing said at least one encrypted version of said key at each of said first station and said second station, whereby thereafter any message encrypted in such a way that it may be decrypted by said key may be decrypted at either of said two remote stations by retrieving said stored encrypted key, writing said at least one encrypted version of said encrypted key to a spatial light modulator (SLM) of an optic correlator and inputting either one of said first characteristic information signal and said second characteristic information signal to said optic correlator.

In accordance with another aspect of the present invention, there is provided a method for the secure handling of data between two remote stations, comprising the steps of: at a base station, encrypting a message such that said message may be decrypted by a decryption key; passing said message to a remote station; at said remote station, obtaining from a user of said remote station a remote station user optical characteristic information signal; retrieving from storage an encrypted version of said decryption key, said encrypted decryption key having the property that when it is written to a SLM of an optical correlator, the output of said correlator is similar when input with either one of said remote station user characteristic information signal or a base station user optical characteristic information signal; writing a remote station optical correlator with said encrypted decryption key; inputting said remote station correlator with a Fourier transform of said remote station user optical characteristic information signal; regenerating said decryption key from an output of said remote station correlator; and decrypting said message with said decryption key.

Brief Description of the Drawings

Figure 1 is a schematic view of a system for use in the secure handing of data between two remote stations made in accordance with this invention,

figure 2 is a schematic detail of a portion of figure 1, and

figure 2A is a schematic representation of an alternative embodiment for a portion of figure 2.

Detailed Description of the Preferred Embodiments

Turning to figure 1, a system indicated generally at 10 for permitting the secure passing of data between two remote stations, comprises a base station indicated generally at 12 and a remote station indicated generally at 14. The base station comprises a processor 16 linked to a correlator 21, a random character generator 22, a memory 24, and an operator input device 26. The correlator 21 is optically linked to a characteristic input device 20. The processor 16 of the base station 12 is connected for two-way communication with a processor 36 of remote station 14 on line 30. The processor 36 of the remote station is linked to a correlator 41, a memory 42, and an operator input device 46. The correlator 41 is optically linked to a characteristic input device 40.

The characteristic input device 20 and correlator 21 of base station 12 are detailed in figure 2. Turning to figure 2, input device 20 comprises a source of coherent light 222 and input prism 224 with an optical output 225 to correlator 21. The correlator 21 comprises a Fourier transform lens 228, a full-complex spatial light modulator (SLM) 230, an inverse Fourier transform lens 232, a CCD camera 234 with an A\D convertor 236 outputting to processor 16 on line 237. The processor outputs to the input of SLM 230 on line 260. The characteristic input device 40 and correlator 41 of remote station 14 may be identically constructed.

System 10 is used, firstly, to develop an encrypted version of a message decryption key at the base station which may be transmitted to the remote station without concern for privacy and, subsequently, to encrypt messages at either of the stations for transmission to other of the stations where they may be decrypted.

(i) Developing an encrypted decryption key

Assuming the user of base station 12 wishes to communicate in a secure fashion with the user of remote station 14, the user of the base station first agrees upon a temporary secret key with the user of the remote station. This secret key can, for example,

be based on a Diffie-Hellman key derivation, an exponential key derivation scheme or public key system. The user of the remote station then utilizes input device 40 to develop an information signal impressed with characteristics peculiar to the remote station user. With the input device 40 and correlator 41 configured as shown in figure 2, the remote station user activates the light source of the input device and causes the processor 36 to make the SLM of the correlator transparent so that the correlator is effectively bypassed. Next the remote station user places his finger on the input prism creating an optical signal impressed with characteristics of the fingerprint of the user. This optical characteristic signal is imaged at the camera. This characteristic information signal is then digitized and passed to the processor 36. The previously agreed upon secret key is used to encode the digitized fingerprint and this encrypted fingerprint may then be passed to the base station 12 on line 30.

At the base station 12, referencing figure 2, the base station user may activate light source 222 and cause processor 16 to make SLM 230 transparent. The base station user may then place his fingerprint 226 on the input prism so that a fingerprint (characteristic) information signal is imaged at the camera 234. The digitized version of this signal is then passed to processor 16. Returning to figure 1, the processor decrypts the fingerprint information signal from the remote station utilizing the previously agreed upon method to generate a temporary secret key, which may either be derived by processor 16 and stored in memory 24 or input directly from the operator input 26. Next the processor 16 numerically determines spatial Fourier transforms of the remote station fingerprint information signal and the base station fingerprint information signal.

The processor now prompts random character generator 22 to generate a sequence of random characters which will comprise a decryption key. The processor 16 then develops a key function which represents the key. For example, the key representing function could be developed by applying each character of the decryption key as a coefficient to a set of normalized orthogonal basis functions, preferably, delta-shaped functions. The processor then numerically calculates a Fourier transform of the key representing function.

Next, the processor obtains an encrypted version of the decryption key. In the first embodiment of the invention, this step includes developing a composite filter based on the remote station fingerprint information signal, the base station fingerprint information signal, and the key representing function. This composite filter has the property that when it is written to the SLM, the output of the correlator is similar when input with either the remote station fingerprint information signal or the base station fingerprint information signal. Preferably, this output is a set of narrow peaks, the positions of which correspond to the maxima of the delta-shaped basis functions. Methods of obtaining a composite filter with these properties are known to those skilled in the art and described in, for example, an article entitled "Tutorial Survey of Composite Filter Designs for Optical Correlators" by B.V.K. Vijaya Kumar, Applied Optics, Volume 31, No. 23, pages 4773 to 4801. Briefly, the composite filter may be constructed as a linear combination of the complex conjugate Fourier transforms of the remote station fingerprint information signal and the base station fingerprint information signal multiplied by the Fourier transform of the key representing function. The coefficients of the linear combination are determined from a set of equations derived in accordance with certain criteria.

To illustrate the process of composite filter development, let us consider a case of two fingerprints, $f_1(x)$ and $f_2(x)$, where $f_1(x)$ and $f_2(x)$ are the base and the remote station fingerprint information signals, respectively (we use a one-dimensional spatial coordinate system for simplicity). The Fourier transforms of these signals are $F_1(q)$ and $F_2(q)$ respectively, where q is a coordinate in a Fourier domain.

The key representing function may be written as

$$k(x) = \sum_{n=1}^{N} k_n \, \delta(x - x_n) \ ,$$

where $\delta()$ is a delta-function; x_n are the coordinates of the narrow peaks and N is the number of the peaks; k_n are numerical coefficients. The Fourier transform of the key representing function is

$$K(q) = \sum_{n=1}^{N} k_n \exp(- iqx_n)$$

The composite filter, H(q), may be presented in the form

$$H(q) = K(q) (C_1F_1^*(q) + C_2F_2^*(q)) ,$$

where coefficients C_1 , C_2 should be determined; " * " means complex conjugation. If this filter is put on a SLM and the SLM is illuminated with the signal $f_1(x)$, we will get a correlation function, $B_1(x)$, at the output of the correlator, and a correlation function $B_2(x)$ for the signal $f_2(x)$. For the correlation functions we have:

$$B_1(x) = (1/2\pi)C_1 \sum_{n=1}^{N} k_n \int F_1(q)F_1^*(q) \exp(iq(x-x_n)) dq +$$

$$(1/2\pi)C_2 \sum_{n=1}^{N} k_n \int F_1(q)F_2^*(q) \exp(iq(x-x_n)) dq ,$$

$$B_2(x) = (1/2\pi)C_1 \sum_{n=1}^{N} k_n \int F_2(q)F_1^*(q) \exp(iq(x-x_n)) dq +$$

$$(1/2\pi)C_2 \sum_{n=1}^{N} k_n \int F_2(q)F_2^*(q) \exp(iq(x-x_n)) dq$$

Substituting $x = x_n$, $n = 1, 2, \ldots N$ into the equations and setting, for example, the sums $\Sigma B_1(x_n)$, $\Sigma B_2(x_n)$ equal to certain values, we can obtain as many algebraic equations as necessary to find the unknown variables C_1 , C_2 , k_n and to develop the composite filter. To make sure that the number of the equations equals the number of the unknown coefficients, one can use different criteria. For example, a sum (or a sum of squares, or a product, etc.) of the heights of the output narrow peaks is set equal to a certain value. In another embodiment, the height of each peak is set equal to a certain value, but in this case both users (i.e. at the base station and at the remote station) record a few fingerprint information signals, that is, the number of the signals equals or exceeds the number of the peaks in the key representing function.

In the second embodiment of the invention, the step of obtaining an encrypted version of the decryption key includes dividing the Fourier transform of the key representing function by the Fourier transform of the base station fingerprint information signal to obtain a first filter, and dividing the Fourier transform of the key representing function by the Fourier transform of the remote station fingerprint information signal to obtain a second filter. A concatenation of the two filters can now be stored and this yields the encrypted version of the decryption key for both base and remote station fingerprint information signal.

The encrypted version of the decryption key may be stored in memory 24. Also, because the decryption key is encrypted, it may be passed to the remote station on line 30 and will remain secure even if intercepted. The remote station stores the received encrypted decryption key in its memory 42.

In a third embodiment, the decryption key generated by the base station is encrypted by the temporary secret key and transmitted to the remote station over line 30. Each station may then develop a key representing function using the techniques aforedescribed. Then each station develops a filter based on the developed key representing function and the characteristic information signal of that station, again using techniques as aforedescribed. A number of alternative approaches for generating both key representing functions and filters are described in U.S. patent application No. 08/508,978 filed July 28, 1995 and PCT/CA95/00509 filed Sept. 6, 1995, the disclosures of which are incorporated herein by reference.

(ii) Sending messages

Once an encrypted version of the decryption key is present at both the base and remote stations, encrypted messages may be sent from either station to the other and decrypted by the recipient station. For example, if the base station user wished to send an encrypted message to the remote station, he could obtain the decryption key by applying his fingerprint to the characteristic input device 20 and prompting processor 16 to write

SLM 230 with the encrypted decryption key. This will return the key representing function at camera 234 from which the key can be extracted by the processor. The base station user may then input a message by way of operator input 26 which message may be encrypted with the decryption key and the encrypted message sent on line 30 to the remote station.

In the second embodiment of the invention, the processor 16 writes to the SLM each of the previously concatenated two filters of the encrypted decryption key either in sequence or simultaneously. If the fingerprint is the same as was used at the base station during developing the encrypted decryption key, the camera 234 will register a set of narrow peaks in the case of the first filter and a random pattern in the case of the second filter. The positions of the peaks correspond to the maxima of the delta-shaped basis functions and, thus, determine the decryption key.

At the remote station, the remote user may prompt processor 36 to retrieve the encrypted decryption key from memory and write same to the filter of correlator 41. Next this user may input his fingerprint to characteristic input device 40. This will cause the correlator to return the key representing function to the processor 36 so that the processor may determine the key from this function. The decryption key may then be used to decrypt the incoming message.

In a similar fashion, the remote station user could encrypt a message by obtaining the decryption key in the manner aforedescribed and inputting a message to be encrypted at operator input 46. The encrypted message could then be decrypted by the base station in the same fashion as the remote station decrypts messages passed in the other direction.

The only difference between the base station and the remote station is the presence of random character generator 22 at the base station. The roles of these stations may be easily reversed by including a random character generator at the remote station.

As described, the subject invention is suitable for use in secure communications between two computers where the decryption key is released only by applying the fingerprint of the proper user to an input device. Of course, the characteristic input device may be modified to accept other body parts of a user so that a different biometric, such as a vein structure, or an iris pattern of a user is input.

Where the base station user is an entity such as a corporation or other organization, it may not be desirable to have access controlled by a biometric of a single individual. Figure 2a illustrates an alternative characteristic input device 300 which may be used in such instance. Turning to figure 2a, input device 300 comprises a SLM 324 held in place by holder 318 in the light path of coherent light source 222. Processor 16 writes a corporation's proprietary characteristic information (PCI) on the SLM 324 which impresses the light beam with selected characteristics such that a characteristic information signal is generated. When not in use, the PCI would be stored in a secure location in the corporation.

If the base station is sufficiently secure, it may be preferred to store an unencrypted version of the decryption key in memory 24. In such instance, correlator 21 becomes unnecessary and may be replaced with an imaging lens, CCD camera, and A/D convertor. The only use made of the base station characteristic input device would then be during generation of the encrypted decryption key.

System 10 has been described in conjunction with a decryption key which is a symmetric private key. Alternatively, the decryption key could be the private key for public key encrypted messages.

Certain parts of the subject invention have been described as using Fourier Transforms which are an expansion on a set of complex exponential orthogonal basis functions. Alternatively, other orthogonal expansions on a set of basis function can also be used such as Walsh and wavelet functions.

Other modifications will be apparent to those skilled in the art and, therefore, the invention is defined in the claims.

WHAT IS CLAIMED IS:

1. A method for permitting the secure passing of data between two remote stations, comprising the steps of:

- obtaining from a user of a first of two remote stations, a first characteristic information signal;
- obtaining from a user of a second of two remote stations, a second characteristic information signal;
- generating a sequence of random characters to obtain a random key;
- obtaining a key function which represents said key;
- obtaining a Fourier transform of said key representing function;
- obtaining at least one encrypted version of said key based on said Fourier transform of said key representing function, and a least one of said first characteristic information signal and said second characteristic information signal such that said key may be recovered by writing said at least one encrypted version of said encrypted key to a spatial light modulator (SLM) of an optic correlator and inputting either one of said first characteristic information signal and said second characteristic information signal to said optic correlator;
- storing said at least one encrypted version of said key at each of said first station and said second station, whereby thereafter any message encrypted in such a way that it may be decrypted by said key may be decrypted at either of said two remote stations by retrieving said stored encrypted key, writing said at least one encrypted version of said encrypted key to a spatial light modulator (SLM) of an optic correlator and inputting either one of said first characteristic information signal and said second characteristic information signal to said optic correlator.

2. The method of claim 1 wherein the step of obtaining a first characteristic information signal comprises obtaining an optical beam modulated with a biometric image of a first body part of said user of said first station, registering said optical beam in a two-dimensional plane and digitizing said registered optical beam.

3. The method of claim 2 wherein the step of obtaining a second characteristic information signal comprises obtaining an optical beam modulated with a biometric image of a second body part of said user of said second station, registering said optical beam in a two-dimensional plane and digitizing said registered optical beam.

4. The method of claim 3 wherein the step of obtaining said key representing function comprises obtaining normalized orthogonal basis functions and, for each basis function, applying a character of said key as a co-efficient.

5. The method of claim 4 wherein said first characteristic information signal is obtained at said first station and including the steps of:
- encrypting said digitized registered optical beam modulated with a biometric of a first body part with a pre-selected key to obtain an encrypted first biometric signal;
- sending said encrypted first biometric signal to said second station;
- utilizing said pre-selected key at said second station to decrypt said encrypted biometric of said first body part; and
- obtaining said encrypted key at said second station.

6. The method of claim 4 wherein said key representing function is obtained at said first station and including the steps of:
- encrypting said key representing function with a pre-selected key to obtain an encrypted key representing function;
- sending said encrypted key representing function to said second station;
- utilizing said pre-selected key at said second station to decrypt said encrypted key representing function; and
- obtaining said encrypted key at said second station.

7. A method for the secure handling of data between two remote stations, comprising the steps of:
- at a base station, encrypting a message such that said message may be decrypted by a decryption key;

- passing said message to a remote station;
- at said remote station,
- obtaining from a user of said remote station a remote station user optical characteristic information signal;
- retrieving from storage an encrypted version of said decryption key, said encrypted decryption key having the property that when it is written to a SLM of an optical correlator, the output of said correlator is similar when input with either one of said remote station user characteristic information signal or a base station user optical characteristic information signal;
- writing a remote station optical correlator with said encrypted decryption key;
- inputting said remote station correlator with a Fourier transform of said remote station user optical characteristic information signal;
- regenerating said decryption key from an output of said remote station correlator; and
- decrypting said message with said decryption key.

8. The method of claim 7 wherein the step of encrypting a message at said base station comprises encrypting said message utilizing said decryption key.

9. The method of claim 8 wherein the step of encrypting a message at said base station comprises the steps of:
- obtaining from a base station user said base station optical characteristic information signal, such that said base station optical characteristic signal is impressed with characteristics of a body part of said base station user;
- retrieving from storage said encrypted version of said decryption key;
- writing a base station optical correlator with said encrypted decryption key;
- inputting said base station correlator with said base station user optical characteristic information signal;
- regenerating said decryption key from an output of said base station correlator; and
- encrypting said message with said regenerated decryption key.

10. The method of claim 4 wherein said step of obtaining at least one encrypted version of said key is based on both said first characteristic information and said second characteristic information signal.

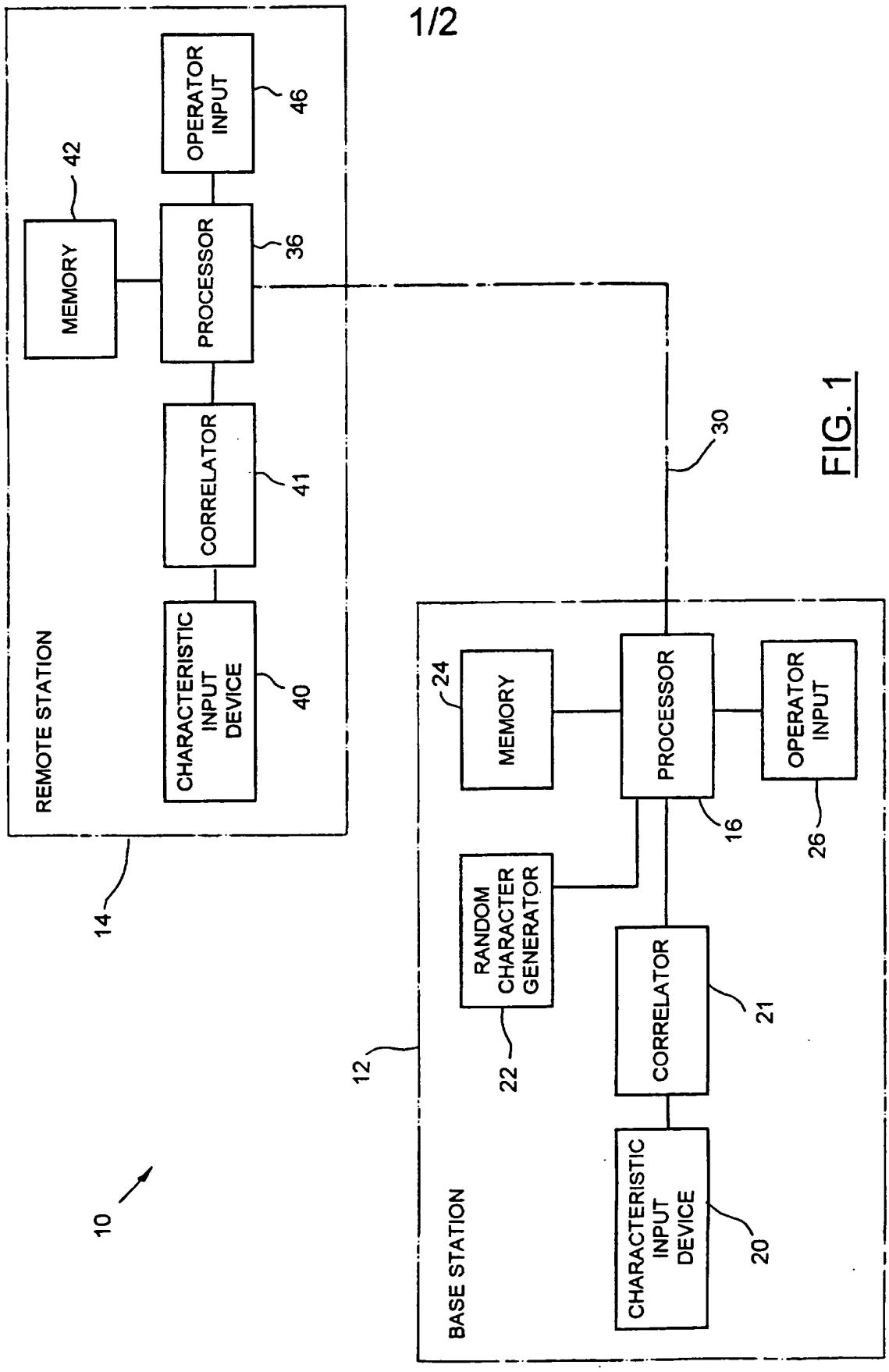


FIG. 1

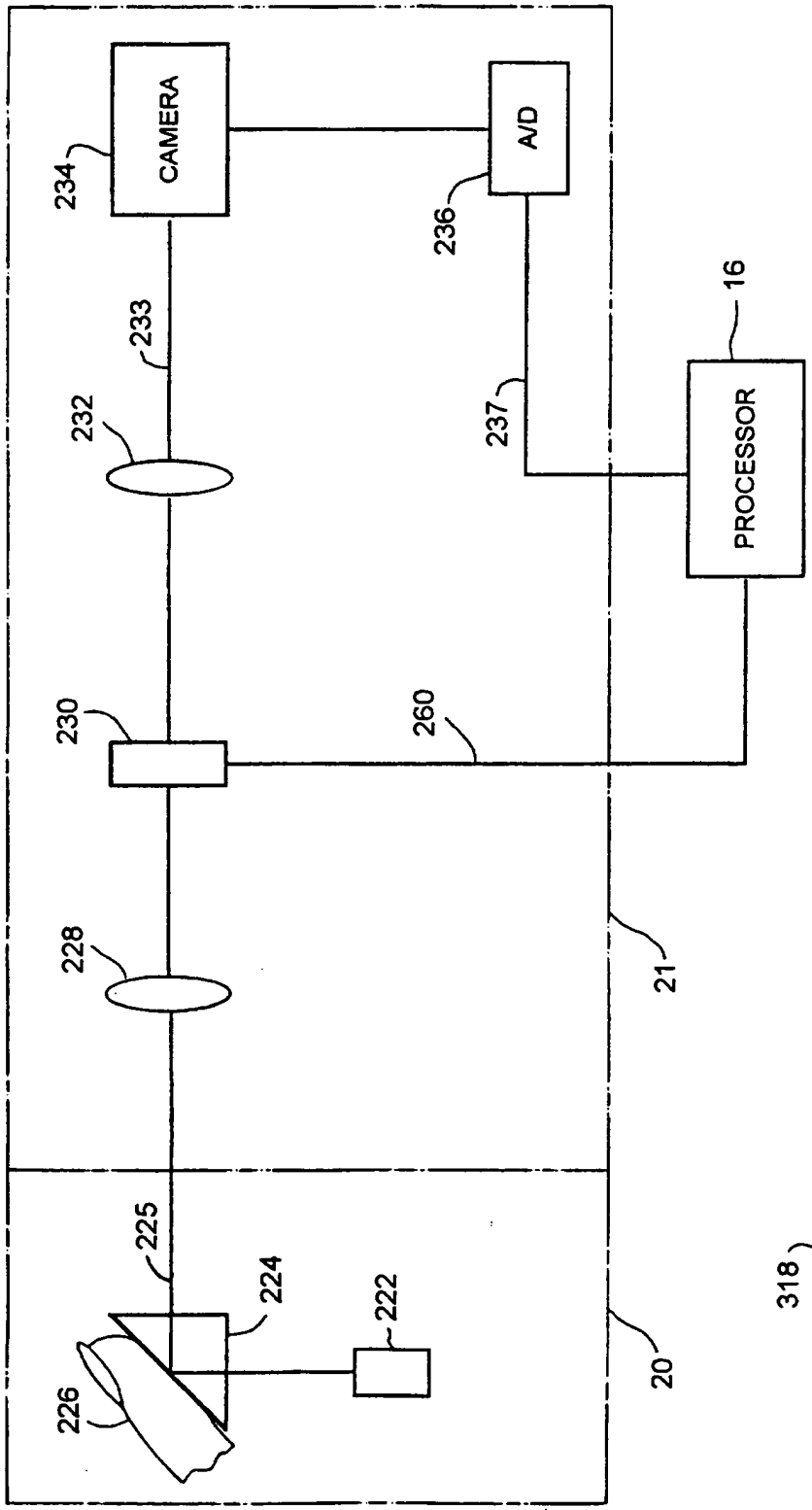


FIG. 2

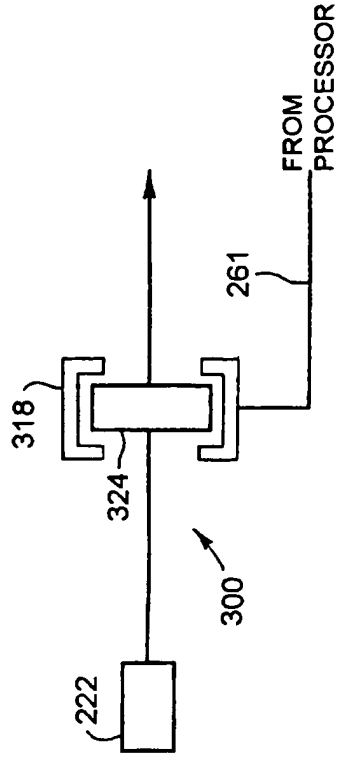


FIG. 2A

INTERNATIONAL SEARCH REPORT

Interna l Application No
PCT/ cA 96/00847

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L9/08 G07C9/00

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 H04L G07C

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.
A	US 4 532 508 A (RUELL) 30 July 1985 see column 1, line 57 - column 2, line 2 see column 2, line 18 - line 30 see column 3, line 44 - column 4, line 41 ---	1,2,7
A	ADVANCES IN CRYPTOLOGY, PROCEEDINGS OF CRYPTO 82, SANTA BARBARA, CA, USA, 23-25 AUG. 1982, ISBN 0-306-41366-3, 1983, NEW YORK, NY, USA, PLENUM, USA, pages 219-229, XP002029301 MUELLER-SCHLOER C ET AL: "Cryptographic protection of personal data cards" see page 226, line 2 - page 228, last line ---	1,5
A	US 5 095 194 A (BARBANELL) 10 March 1992 see column 3, line 43 - column 5, line 24 see column 6, line 28 - line 59 see column 7, line 35 - column 8, line 47 --- -/--	1,7

[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 April 1997	28.04.97

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	Holper, G

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DE 42 43 908 A (GAO) 30 June 1994 see column 2, line 32 - line 48 see column 3, line 30 - line 51 see column 4, line 18 - column 5, line 17 ---	1,7
A	US 5 050 220 A (MARSH ET AL.) 17 September 1991 see column 5, line 18 - column 6, line 24 -----	7

1

INTERNATIONAL SEARCH REPORT

..mation on patent family members

Interna al Application No
PCT/ A 96/00847

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 4532508 A	30-07-85	EP 0121222 A	10-10-84
US 5095194 A	10-03-92	NONE	
DE 4243908 A	30-06-94	NONE	
US 5050220 A	17-09-91	NONE	

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 3 : H04L 9/00	A1	(11) International Publication Number: WO 83/ 04461 (43) International Publication Date: 22 December 1983 (22.12.83)

(21) International Application Number: PCT/US83/00030

(22) International Filing Date: 11 January 1983 (11.01.83)

(31) Priority Application Number: 386,805

(32) Priority Date: 9 June 1982 (09.06.82)

(33) Priority Country: US

(71) Applicant: WESTERN ELECTRIC COMPANY, INC. [US/US]; 222 Broadway, New York, NY 10038 (US).

(72) Inventors: EVERHART, Joseph, Robert ; P.O. Box 228, 3 Old Mill Road, Holmdel, NJ 07733 (US). OSBORN, Jeffrey, George ; 242 Madison Gardens, Old Bridge, NJ 08857 (US).

(74) Agents: HIRSCH, A., E., Jr. et al.; Post Office Box 901, Princeton, NJ 08540 (US).

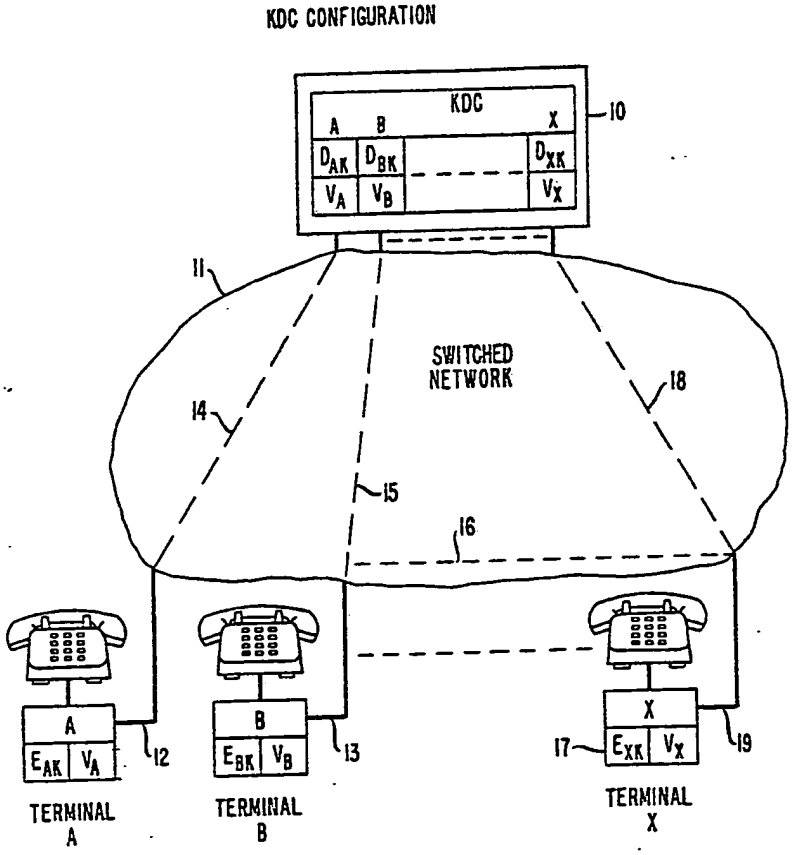
(81) Designated States: AT (European patent), AU, BE (European patent), CH (European patent), DE (European patent), FR (European patent), GB (European patent), JP, LU (European patent), NL (European patent), SE (European patent).

Published
With international search report.

(54) Title: ENCRYPTION SYSTEM KEY DISTRIBUTION METHOD AND APPARATUS

(57) Abstract

Encryption systems typically rely on the distribution of cipher keys between terminals for scrambling and unscrambling transmitted messages. Elaborate security precautions are necessary to protect the cipher keys since a compromise of the key could result in a compromise of the transmission. There is disclosed a key distribution method and apparatus which uses a channel (14, 15, 18) from identified terminals (A, B, X) to a central key distribution center (KDC) for the establishment, on a one-session basis, of the key which is to be used for the next session between those terminals. The key establishing link (16) is itself encoded using a cipher key which changes after each usage. Provision is made to verify, for each new connection, that a compromise has not priorly occurred.



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	**LI**	Liechtenstein
AU	Australia	**LK**	Sri Lanka
BE	Belgium	**LU**	Luxembourg
BR	Brazil	**MC**	Monaco
CF	Central African Republic	**MG**	Madagascar
CG	Congo	**MR**	Mauritania
CH	Switzerland	**MW**	Malawi
CM	Cameroon	**NL**	Netherlands
DE	Germany, Federal Republic of	**NO**	Norway
DK	Denmark	**RO**	Romania
FI	Finland	**SE**	Sweden
FR	France	**SN**	Senegal
GA	Gabon	**SU**	Soviet Union
GB	United Kingdom	**TD**	Chad
HU	Hungary	**TG**	Togo
JP	Japan	**US**	United States of America
KP	Democratic People's Republic of Korea		

ENCRYPTION SYSTEM KEY DISTRIBUTION METHOD AND APPARATUS

Background of the Invention

This invention relates to the establishment and distribution of cipher keys in a cryptographic system.

Cryptographic systems are now gaining favor, both for voice as well as data transmission. In such systems it is typically necessary that the parties to a particular transmission each have cryptographic keys to encrypt and decrypt the cipher transmissions. It follows that a compromise to a cryptographic key will in turn reduce the security of subsequent transmissions involving that key. Thus, great precautions must be taken to distribute the cryptographic keys among the system users. Such distribution, for example, using secure couriers to manually update the keys may be possible when the community of users is priorly known but becomes increasingly more difficult when either the number of parties is large or parties who seldom communicate with each other wish to do so. The responsibility for keeping the cryptographic key secure after distribution rests with each user and the longer the key remains effective the greater the risk of it becoming compromised.

Thus, from a practical point of view it is desirable to have the cryptographic key effective for a single session, requiring a new key for each new session. When couriers are used, however, this becomes costly and time consuming, especially when a party wishes to place many secure calls or have many secure sessions.

Attempts have been made to electronically distribute cryptographic keys between users from a key distribution center. One such example is shown in Rosenblum Patent No. 4,182,933, issued January 8, 1980. While such attempts have found some degree of success they all suffer from the problem that they are subject to

compromise because they usually rely on the security of the transmission media between the key distribution center and the terminal for the distribution of session key information. Thus, an intruder need only compromise the key distribution channel to obtain subsequent session keys. Elaborate systems have sometimes been established to detect such a compromise, all of which are either costly or minimally effective.

Another problem with key distribution centers is that the center can derive the information used to decrypt the secure data exchange between users and thus could theoretically monitor the secure session transmission.

Summary of the Invention

We have solved the above-identified problems by arranging a key distribution center (KDC) which communicates over a channel with the individual terminals. The channel, or data link, can be a dial-up telephone line, a packet-switched data network, dedicated lines, or other communications channel types, over which secure communication is possible. The terminals operate in conjunction with the KDC to establish a session key for secure transmission between two or more terminals. The session key at a terminal is constructed from information generated at that terminal in conjunction with information communicated from the KDC and is known fully only to the terminals involved in the session and not to the KDC. Thus, when two terminals have established a session key, they may securely communicate with each other for the duration of that session.

At the conclusions of the secure data exchange, the session keys should be destroyed, and when either station wishes to establish additional secure communication either between themselves or to other stations, a new session key will be established in cooperation with the KDC.

Both the terminal-KDC channel and the KDC-terminal channel, as mentioned above, are secure links in

that they are protected by cryptographic key information which is unique to each terminal and to the KDC on a one-call-only basis. Accordingly, whenever a connection is established between a terminal and the KDC, each has information previously stored, referred to as terminal-unique key information, and this priorly stored information is used to establish both new KDC-terminal link keys, referred to as call-setup key information, and new session key information. During the establishment of the session keys, the terminal and the KDC each modify their respective terminal-unique key information so that on a next call between the KDC and the same terminal, this new key information must be used in order to establish a secure communication path. The precise manner in which this happens will be discussed hereinafter. In this manner, an intruder on the key distribution between a terminal and the KDC must be adding and substituting information on the channel from the beginning and must stay on the channel throughout several calls, since once the intruder leaves it is possible to detect, at least by hindsight, that a compromise has occurred. This is a result of the fact that the intruder is substituting random information that may be monitored.

One aspect of our system is that an intruder, in order to obtain useful information exchanged between two valid users of the system, must gain the terminal-unique information that is stored at the terminal, and he must also gain the terminal-unique information that is stored in the key distribution center for that specific terminal. The intruder then, on the very next key exchange involving that terminal and the key distributing center, must actively participate, i.e., substitute his own generated key information on that channel. Then the intruder must also substitute information on the channel between the two communicating terminals, and also must continue the above substitutions on the channels for an indefinite period of time or risk detection.

Brief Description of the Drawing

These attributes of our invention, together with the operation and utilization of the invention in a specific embodiment, will be more fully apparent from the illustrative embodiment shown in conjunction with the drawing which:

FIG. 1 shows an overall system using a KDC and several terminals;

FIG. 2 shows an implementation of the initial establishment of information in both the KDC and the terminal within a secure area;

FIGS. 3 and 4 show a flow chart detailing what occurs within each terminal;

FIG. 5 shows a flow chart detailing what occurs within the KDC;

FIGS. 6-19 show, in sequence, an implementation of the establishment of key information and control data within each terminal; and

FIGS. 21-28 show, in sequence, an implementation of the establishment of key information and control data within the KDC. In this system we have a variety of terminals.

General Description

FIG. 1 shows a number of terminals, A, B and X, connectable to each other and to KDC 10 via some transport network (e.g., public switched network). These terminals should be able to set up a secure channel between themselves in order to exchange secure information. In this process they must both communicate with the KDC. The transmission line 12 from terminal A is connected through link 16 to transmission line 13 to initiate a secure call to terminal B. Once the users decide to initiate a secure data exchange, each terminal sets up a transmission line, such as link 14 for terminal A, to the KDC.

An exchange of information will then occur from terminal A to the KDC and from terminal B to the KDC. Once the KDC has received both of these messages, it will

formulate two distinct messages that will be sent respectively to terminal A via link 14 and to terminal B via link 15. These individual messages will contain session key information, as well as other pertinent information described below. This session key information has originated at terminal A and at terminal B and is exchanged through the KDC. Once the exchange has taken place between the two terminals and the KDC, link 14, which is the key distribution link between terminal A and the KDC, is then taken down, and key distribution link 15 between the KDC and terminal B is taken down. Link 16, which is the session link between terminals A and B, is re-established. Further key information is exchanged based on the prior partial exchanges so as to derive independently at both terminals the session key, and finally using that session key information, data (i.e., digital data or digital voice) can be transmitted in secure fashion on data link 16.

Since further session information was derived between terminals A and B independent of the KDC, a malicious operator of the KDC cannot derive the key information need to decrypt the secure messages sent between terminals A and B without actively substituting information on the session channel.

Also, at this point, as will be seen, contained within the messages that were sent between the KDC and the terminals was new terminal-unique key information to secure the next key distribution between the terminals and the KDC. This new information is independent of the previous information and therefore is unique to it.

Detailed Description

Turning now to FIG. 2 the initial setup between the terminal and the KDC must be made in an authentic manner such that the information transported to the terminals from the KDC is not modified. One implementation is where the transport is made within a secured area, such as secured area 23. Since subsequent communications

between the KDC and each terminal depend upon the prior communication, it is important that at some period in time they both contain the proper information for start-up, and ideally this is done in the secured area so that there can be no breach of security.

On the initial system setup (based on the secured area implementation shown in FIG. 2) the terminals are brought within the secured area 23, and the KDC can generate terminal-unique key pairs for each terminal. The exact function of these key pairs will be described later. The KDC will generate a terminal-unique decryption key for each terminal and the corresponding encryption key. This encryption key must be placed in the terminal-unique key storage for each terminal with the corresponding decryption key stored in the terminal-unique key storage at the KDC under the address of that terminal. In addition, a random number, Ua for terminal A, unique to each terminal is stored in the verification information storage at the KDC also at the address of this terminal. This same random number must be loaded and stored in the verification information storage in the terminals and will be used for a verification check on the first call setup to the KDC.

FIGS. 3 and 4 are flow charts representing the action that occurs within a terminal, for example, terminal A.

FIG. 5 is a flow chart representing what actions occur within the key distribution center.

The discussion which will follow is a discussion with respect to a time sequence between the terminal and the KDC to illustrate both how terminal-unique keys are updated, and how call-setup and session keys are distributed. This discussion will occur with respect to FIGS. 6 through 28. FIGS. 6 through 19 show the apparatus within the terminal and show on a step-by-step basis how the call-setup keys and the session keys are established. FIGS. 20 through 28 show the apparatus within the KDC, each figure showing a specific operational aspect of the

establishment of the keys.

Turning now to FIG. 6, we will discuss the specific apparatus used in the terminals. The actual generation of the numbers will be discussed hereinafter. Apparatus 72 is a random number generator which is a device or algorithm that produces bits (zeros and ones) that are equally likely to occur. This generation may be based upon a noisy diode and any number of algorithms can be used to attain statistically independent output of 0's and 1's. The more equally likely these random number generators are, i.e., the more random this function is, the higher the security level will be. The output of the random number generator is a serial stream of zeroes and ones where the correlation between one or a group of bits is zero. The bidirectional asymmetric key generator, apparatus 73, takes as input a random number from random number generator 72 and will compute an encryption key and the matching decryption key such that the encryption key cannot be derived from the decryption key and vice versa. The generation of these keys as an example could be done in accordance with the RSA algorithm, as described by Rivest, Shamir, and Adleman in a paper entitled, "A Method for Obtaining Digital Signatures and Public Key Crypto Systems," which publication is hereby incorporated by reference, which appeared in CACM, Vol. 21, No. 2, February, 1978, on pages 120-126.

Apparatus 74 implements a bidirectional asymmetric cryptographic algorithm (e.g., the RSA algorithm) that is, a cryptographic algorithm based on two distinct keys where the encryption key cannot be derived from the decryption key and vice versa. Apparatus 74 has two inputs (I and K) and one output (O). The input I is the bits to be encrypted or decrypted. The input K is the key, either encryption or decryption (the RSA algorithm performs the same function regardless of encryption or decryption). The output will be the inputted bits encrypted or decrypted with the supplied key. This

algorithm is also described in the aforementioned paper. Functionally, apparatus 75 is the embodiment of two functions f and g such that: given f(R, P) and P, one cannot determine R; g(R1, f(R2, P), P) = g(R2, f(R1, P), P); and given f(R1, P), f(R2, P), and P one cannot determine R1, R2, or g(R1, f(R2, P), P).

Apparatus 75 performs the above functions via, for example, the Diffie-Hellman algorithm, which is described in a paper by Diffie and Hellman entitled "New Directions in Cryptography," published by the IEEE Transactions on Information Theory, Vol. IP-22, November, 1976, on pages 644-655, which is hereby incorporated by reference. The input to this algorithm is a base Y, a modulus Q and an exponent EXP. The output is Y raised to the EXP power modulus the Q. The functions f and g are the same as discussed above in this example.

The storage requirements are depicted by registers 71, 70 and 76. These are the semi-permanent register 71 which contains both the verification information Va and the terminal-unique key information Eak used to encrypt messages to the KDC. Temporary register 70 can be in any state initially and is used during the interaction with the KDC on a secure call setup. The address register permanently contains the address (i.e., a public piece of information that uniquely identifies A to the KDC) of the terminal (terminal A in this case) where it is located. During a secure session (or call) setup, the address register will also contain the address of the terminal which is being called. The registers containing verification information and encryption and decryption information may vary in size depending upon the specific algorithm used but in this example should be on the order of 1,000 bits each. Information pertaining to the symmetric session key and the random number should be on the order of 100 bits, and the address information will be dependent upon a terminal numbering plan both unique and known to the KDC. For example, it could be the telephone

number of the specific terminal or it could be the serial number of the terminal.

Turning to FIG. 20, we will now discuss the working of the modules within the key distribution unit. The address register at the KDC, register 200, performs the same function as the address register at the terminal. The RSA function at the KDC, apparatus 210, performs the same function as the RSA function at the terminal, as previously described. The random number generator, apparatus 211, performs the same function as the random number generator at the terminal previously mentioned. The generator of the encryption and decryption keys apparatus 212 has the same function as described previously in the terminal. Apparatus 213 is a generator of the parameters used as inputs to the apparatus 75 described previously. For this particular example these parameters are the base and modulus for the Diffie-Hellman algorithm. It requires as input the output of the random number generator, apparatus 211. The method of generation is described in the aforementioned paper by Diffie.

There is a semi-permanent storage at the KDC, registers 214 and 216, which stores verification information Va and terminal-unique decryption key information Dak between calls. Semi-permanent registers 215 and 217 are used to store information during the call setup progress. These registers have the same functions as described previously for the terminal.

System Operation

The operation of the system will now be explained beginning with FIG. 3. Initially the key management equipment in the terminal will be in the wait state until a request is received from the terminal controller processor to initiate a secure call. At this point, as discussed, there is stored in the terminal the terminal-unique encryption key that will be used to encrypt information that is sent to the KDC. Also stored is the verification information. These two pieces of information were stored

from the last call (or from the initial setup) that was made by this terminal. This is shown in FIG. 6 as Va and Eak.

Once a request is received to initiate a secure call, the address of the called party must be given to the key management equipment via the controller processor. This is seen in FIG. 3, box 31. At this point, there are generated new call-setup keys. This is shown in box 32 and in FIG. 7 as Eka and Dka. In box 33 there is shown the generation of partial session keys that will be used to encrypt data on the link from terminal B to terminal A. This is shown in FIG. 8 as Eba and Dba.

At this point, the verification information is updated using the keys that were just generated. The update function is specified as follows:

$$Va1' = f(Va1, E1) \text{ and } Va2' = f(Va2, E2)$$

where ' denotes updated and Va1Va2 = Va. Va is the stored verification information and the E's are the just-generated encryption keys. The properties of f are as follows:

(1) for every V, E1, E2: $f(V, E1) \neq f(V, E2)$ where $E1 \neq E2$;

(2) for every V21, V2, E: $f(V1, E) \neq f(V2, E)$ where $V1 \neq V2$;

(3) given V and $V' \neq f(V, E)$ it is difficult to determine E; and

(4) in the case where E is an asymmetric encryption key, D cannot be determined from E.

For this example, Va' = Val'|Va2' where Va = Val|Va2, Val' is equal to Val encrypted with Eka, and Va2' is equal to Va2 encrypted with Eba. This update process is depicted in FIG. 9. The first half of the verification information Val is read from storage and provided as an input to the RSA algorithm. The key that is used to encrypt this information is the call-setup key, Eka, that was just generated. This becomes Val' and overwrites Val as seen in

FIG. 10. Next, the second half of the verification information Va2 is encrypted using Eba just generated. The result Va2' overwrites Va2 in the storage register. This is shown in FIG. 3, box 34, and in summary, the updated verification information Va" is the verification information stored from the previous call, or given to the terminal on the initial setup from the KDC, where half is encrypted using the encryption part of the partial session key generated on this call and the other half is encrypted using the call-setup key for that call.

At this point, as shown in box 36, FIG. 3, and in FIG. 11, the message can be formatted to the KDC. The contents of this message are the encryption parts of the two keys that were just generated. Both the partial session key to be established between terminal A and B, Eba, and the new call-setup key Eka are encrypted using the terminal-unique encryption key Eak stored from the previous call from the KDC to the terminal or given to the terminal on the initial setup. At this point, the information that can be destroyed from the terminal is the terminal-unique encryption key, Eak, stored at the terminal from the previous call, and both the call-setup encryption key, Eka, and the partial session encryption key, Eba, that were generated by the terminal. The encrypted message is then appended to the address, A, of the originating terminal followed by the address, B, of the called terminal. This message is now sent to the KDC.

The terminal now will enter a wait state waiting for the information to be received from the KDC. This is depicted in box 37 of FIG. 3.

As shown in FIG. 5, the KDC will be in a wait state until a message is received from terminal A. This is shown in FIG. 5, box 50. Once the message is received, the KDC reads the address information within the message into the address register which gives it the index of the decryption key that must be used to decrypt the message. The KDC has in its storage from the previous call the

matching verification information for each terminal and the terminal-unique decryption key for each terminal. This is depicted in FIG. 20, boxes 214 and 216.

The message from terminal A is decrypted using the terminal-unique decryption key corresponding to that terminal, Dak. The keys, both the new call setup key Eka and the partial session key Eba (to be distributed to terminal B) is temporarily stored in the KDC memory as depicted in FIG. 21.

At this point, as shown in FIG. 22, the KDC can update its verification information in the exact same manner as the terminal. This is done by encrypting each half of the stored verification information Va with the received session key information Eba and the received call-setup key information Eka, shown in FIG. 23. This produces the update verification information Va".

The key distribution center, as shown in FIG. 24, will now generate a bidirectional asymmetric encryption/decryption key pair, Eak', Dak'. The primes denote updated information. Eak' will be distributed to terminal A to be used on the next call setup to the key distribution center. The decryption key Dak' overwrites the decryption key Dak that was stored from the previous call.

Two other pieces of information are also generated at this time. These are the parameters that will be used by the terminals to create symmetric session keys; in this case they are the parameters of the Diffie-Hellman algorithm. One is the base Y and the other is the modulus Q as previously described. Functionally, the amount of information that is generated at the KDC and sent to each terminal may vary depending upon the precise algorithm. This information is stored in temporary storage and will be used as part of the message sent back to both terminal A and terminal B. This generation process is depicted in FIG. 25 and refers to the flow chart box 55, FIG. 5. By this point, as shown in FIG. 26, the KDC must

have received a message from terminal B in order to complete the call to terminal A. If not, the KDC process for terminal A must wait until the process for terminal B has reached this point. This is so it can give terminal A the partial session key information Eab generated at terminal B and also to be able to give terminal B the partial session key Eba generated at terminal A. Coordination between the processes must take place so that the same parameters generated by one process overwrites the parameters generated by the other process. This insures that the parameters sent to the terminals for the purpose of generating symmetric session keys are the same.

Once the internal exchange is made between the A registers and the B registers to coordinate the information inside the key distribution center, the messages can now be formatted for the terminals. This is shown in FIG. 27. The message to terminal A will consist of the new terminal-unique key information Eak' that will be used on a subsequent call to the KDC. It will also consist of the partial session key information Eab which it received from terminal B. It will also consist of the verification information Va" or a known reduction of Va" in terms of the number of bits. It will also consist of the base Y and the modulus Q of the Diffie-Hellman algorithm. These five pieces of information will be encrypted using the call-setup key Eka received in the message from terminal A. The KDC destroys Eka, Eba, Eak', Y, and Q corresponding to terminal A and destroys Ekb, Eab, Ebk', Y, and Q corresponding to terminal B. The KDC will then send this output message back to terminal A. An analogous encrypted message is sent from the KDC to terminal B. At this point the KDC is finished with its processing.

FIG. 28 shows the configuration of the KDC after the call to terminal A has been dropped. The KDC has updated verification information Va" and updated terminal-unique decrypt key information Dak' which will be used on a subsequent call between terminal A and the KDC.

Referring back to the flow chart, FIG. 3, for terminal A, the key management equipment at the terminal has been in a wait state while the KDC has been functioning. FIG. 12 shows the key information stored at the terminal during this wait state. It is the updated verification Va" information and both decrypt keys Dka and Dba corresponding to the previously generated encryption keys.

FIG. 13 shows how the information received from the KDC is used in accordance with the box 38, FIG. 3. The call-setup decryption key Dka is used to decrypt the message received from the KDC. The five values (previously discussed) sent from the KDC are now used in the following way. The first piece of information is the new distribution key Eak' that is stored in the semi-permanent register 71 and will be used on a following call made from this terminal to the KDC. It is the updated terminal-unique encryption key. The second piece of information is the partial session key Eab which was generated at B and sent through the KDC to terminal A. The third piece of information is the updated verification information Va", which can now be compared with the verification information stored at terminal A. The fourth and fifth pieces of information are the parameters to the Diffie-Hellman algorithm, the base Y and the modulus Q, which terminal A stores in temporary storage.

Referring to FIG. 4, box 40, at this point the terminal will compare the verification information it received from the KDC and either the verification information which is presently stored or some known reduction of that verification information - FIG. 14. If this matches, then the process will continue as normal. If this does not match, an alarm could be given to the terminal controller processor of a potential intruder threat on a previous call.

Assuming a success of the compared verification, the terminal can now take down the channel to the KDC and

establish a channel to terminal B, if not already established. At this point, terminal A and terminal B can communicate data securely using the asymmetric session keys Eab and Eba. If a symmetric session key is needed, the following steps can be taken. The calculation of the message to be sent to terminal B is shown in FIG. 15. First, the base Y and modulus Q of the Diffie-Hellman algorithm are used along with a random number Ra generated by the random number generator 72. These inputs are given to the Diffie-Hellman algorithm 75 and the output is then an input to the RSA function 73. The random number Ra is also stored in temporary storage. Eab is used as the key to the RSA function 73. At this point the session key information Eab received from terminal B and the base number Y may be destroyed. The output of the RSA algorithm is sent to terminal B.

Terminal A' key management equipment will now enter a wait state shown in FIG. 4, box 44, waiting for a message to be returned from terminal B. The idle state is depicted in FIG. 16 and in storage is the decrypt session key Dab which terminal A generated, the modulus Q of the Diffie-Hellman algorithm generated by the KDC and the random Ra number that was generated by terminal A.

As shown in FIG. 17, upon receipt of the message from terminal B, terminal A will decrypt the message using its decryption key Dba stored from the initial generation of the partial session key. Dba can now be destroyed. The output of this will be fed into the Diffie-Hellman algorithm as the base. The exponent will be the random number Ra which was priorly generated and the modulus Q is also input into the algorithm. The output of the Diffie-Hellman algorithm will be symmetric session key information which will equal the session key information that terminal B has calculated. Q and Ra can now be destroyed.

At this point, terminals A and B have established symmetric session key information between themselves that is not derivable by the KDC. This key information may be

used in a symmetric key algorithm like the Data Encryption Standard (DES) to encrypt data. What is stored now in the terminal until the next request for a secure session (or call), as shown in FIG. 18, is the updated verification information Va" and the terminal-unique key Eak' which it received from the KDC to be used to encrypt the next message to the KDC.

It should be noted that the actual generation of the desired data at the terminal and at the KDC is operative under control of a computer processor and is programmed in accordance with the flow charts shown in FIGS. 3-5 to perform the sequence of data transfers detailed herein. Such a processor, while not shown, can be any one of several well-known microprocessors, such as for example, the Intel 8086 microprocessor, working in conjunction with the terminal and KDC apparatus shown and detailed herein above.

It should also be noted that one skilled in the art could use different encryption algorithms and different equipments to achieve the same results disclosed herein without departing from the spirit and scope of our invention.

Claims

1. A key distribution method for communicating cipher keys between two terminals via a key distribution center, KDC, said method comprising

establishing between any one terminal and said key distribution center a terminal-unique cipher key,

cooperating between said KDC and said one terminal on a subsequent connection between said KDC and said one terminal to establish a session key for use by said one terminal in a subsequent secure transmission between said one terminal and a second terminal, and

changing in response to said subsequent connection between said one terminal and said KDC said priorly established terminal-unique cipher key.

2. The invention set forth in claim 1 wherein said session key is generated from the asymmetric exchange of information between said one terminal and said KDC plus the subsequent exchange of information between said first and second terminals.

3. The invention set forth in claim 2 wherein said session key at said one terminal is random with respect to information at said KDC.

4. The invention set forth in claim 2 wherein said session key at said one terminal is underivable with respect to any information at said KDC.

5. A key distribution center for controlling the dissemination of session cipher keys between remotely located terminals, said center arranged for switched access to a plurality of said terminals, said center comprising

means for establishing communication cipher keys between said center and each said terminal having access thereto, each cipher key unique to each said terminal,

means operative when one of said terminals accesses said center for bidirectional asymmetrically exchanging information with said accessed terminal using, as a foundation for said exchange, said priorly established communication cipher keys, and

means responsive to said exchanged information for communicating to said terminal information allowing said terminal to establish a session cipher key for use with an identified other terminal also having access to said center.

6. The invention set forth in claim 5 wherein said key distribution center further comprising means for changing said established communication cipher keys as a result of said exchanged information.

7. The invention set forth in claim 5 wherein said cipher key establishing means uses information from a prior transmission from a particular terminal for establishing said cipher keys to said particular terminal.

8. The invention set forth in claim 5 wherein said exchanged information includes information generated in part at said center for the random generation of said session key allowing said session key to be underivable with respect to any information at said center.

9. A key distribution center for controlling the distribution of cipher control information among a number of terminals, said center comprising

means for individually exchanging encoded information between any of said terminals, said exchange for any particular terminal based partially upon a last information exchange between said particular terminal and said center,

means for identifying at least two terminals where encrypted session information is to be exchanged and for accepting from said identified terminals certain encryption control information, and

means for modifying, according to a pre-established pattern, accepted information from said identified terminals and for communicating said modified information to the other of said terminals so as to allow each of said terminals to thereafter establish, independent of any information available at said center, a cipher key allowing said session information to be encrypted.

FIG. 1

KDC CONFIGURATION

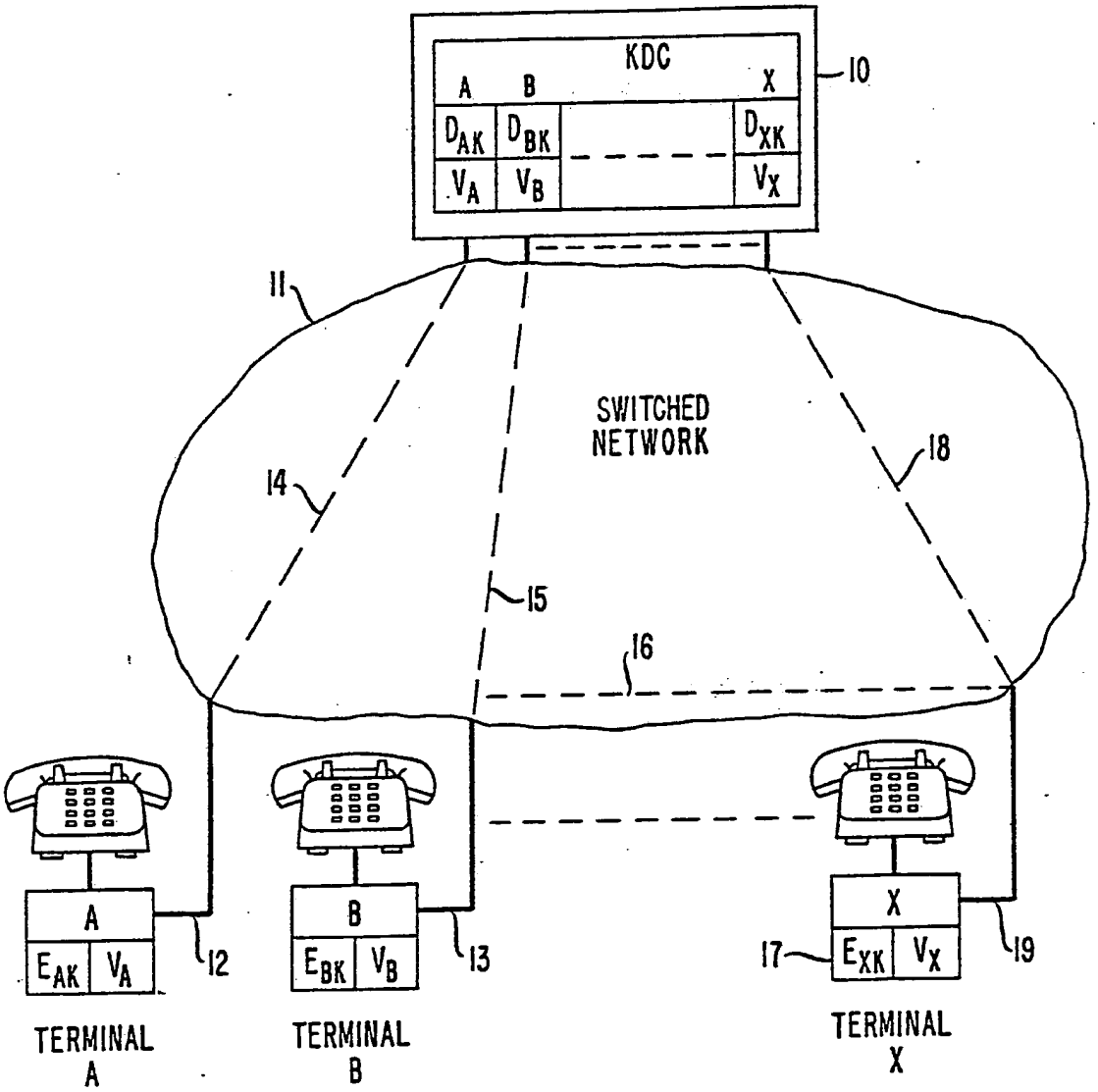


FIG. 2

INITIAL SYSTEM SETUP

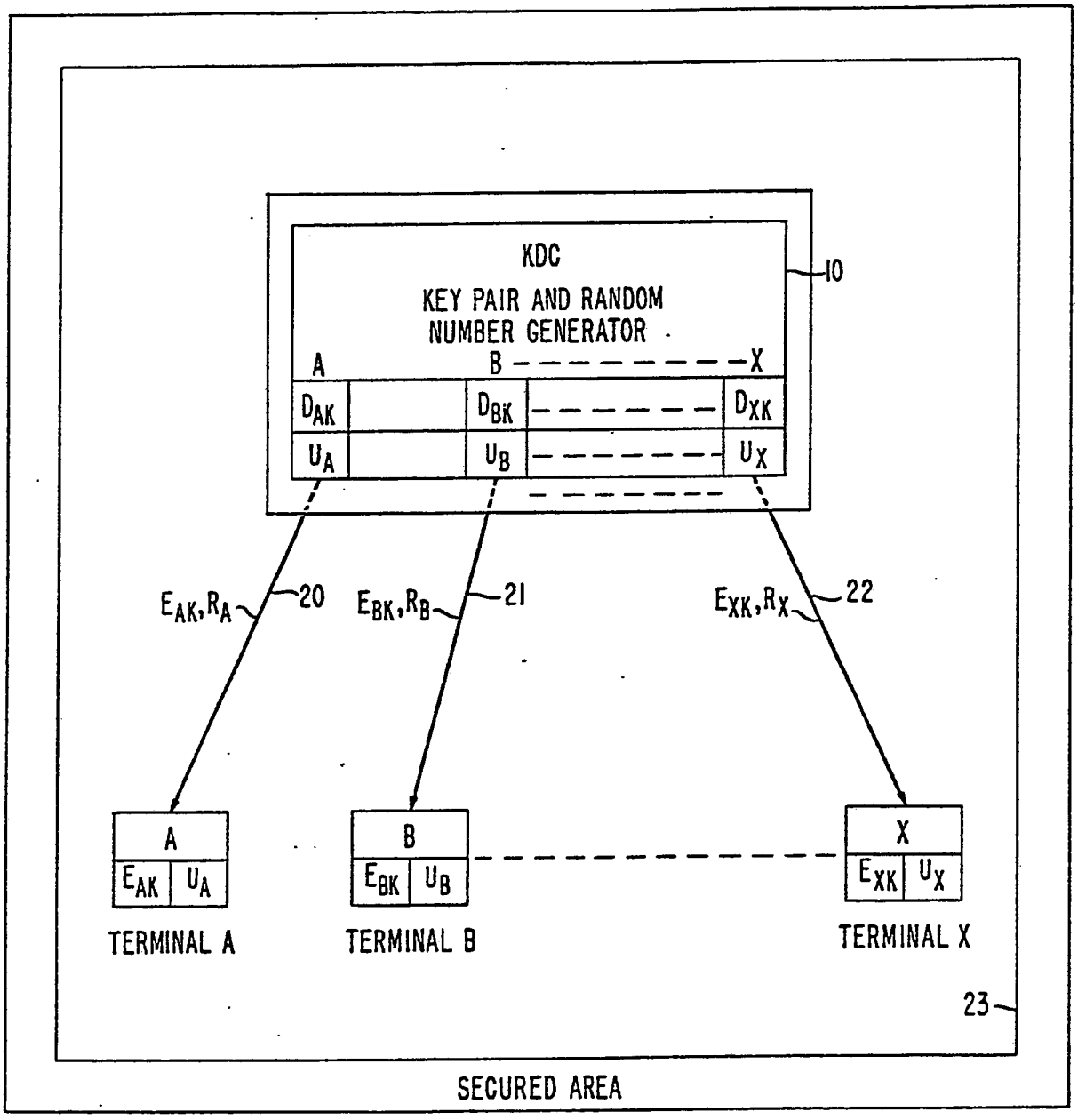


FIG. 3

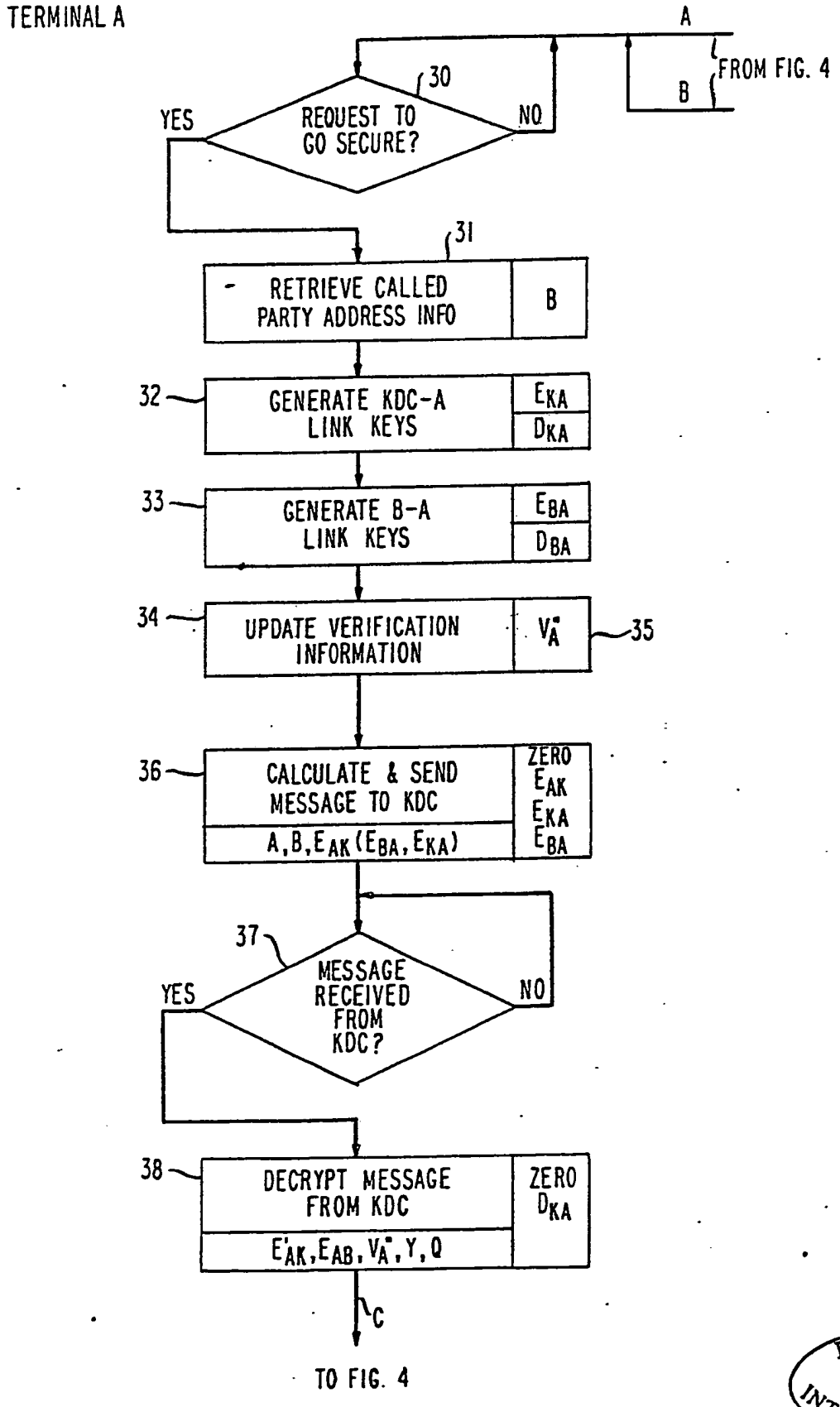


FIG. 4
TERMINAL A

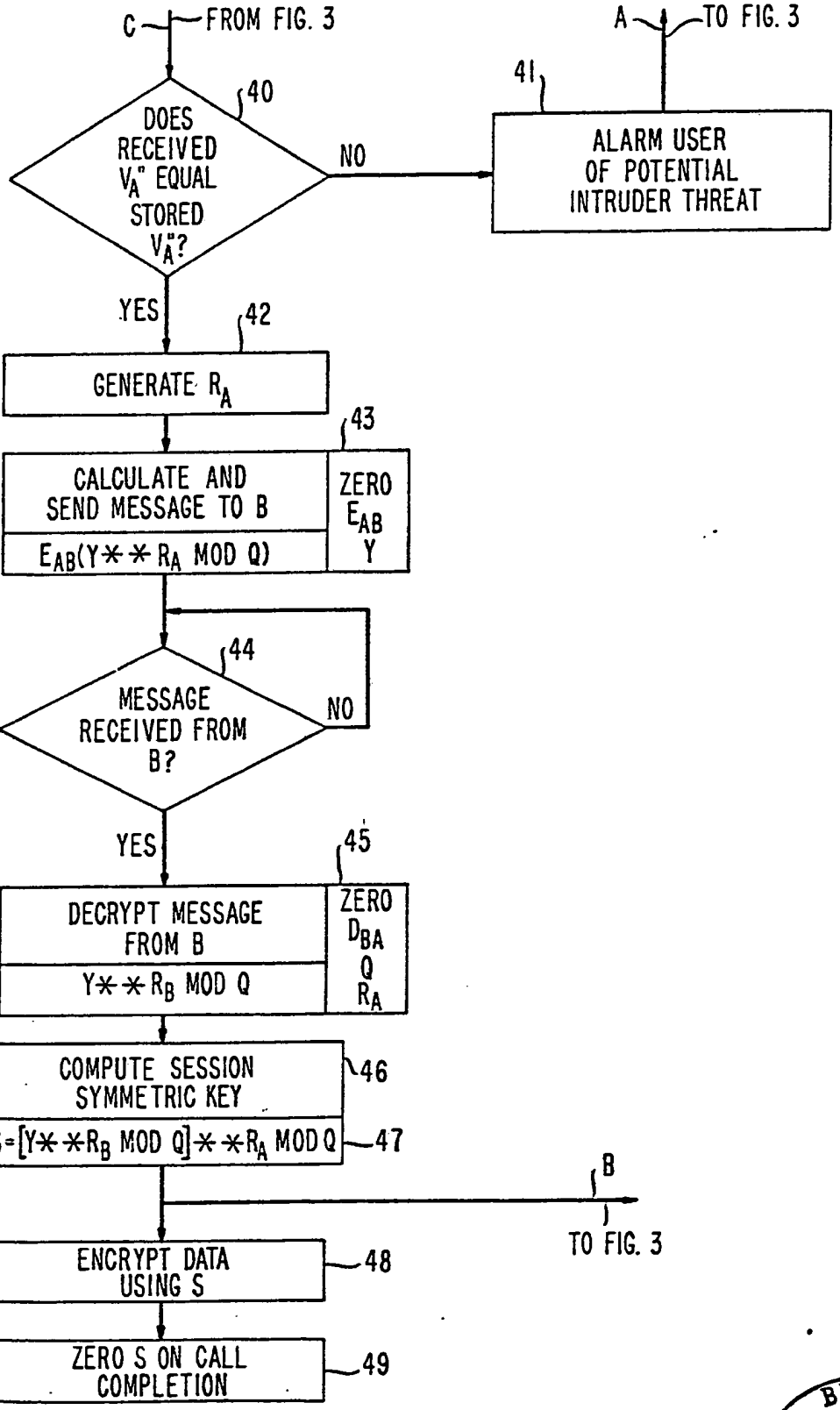


FIG. 5
KDC

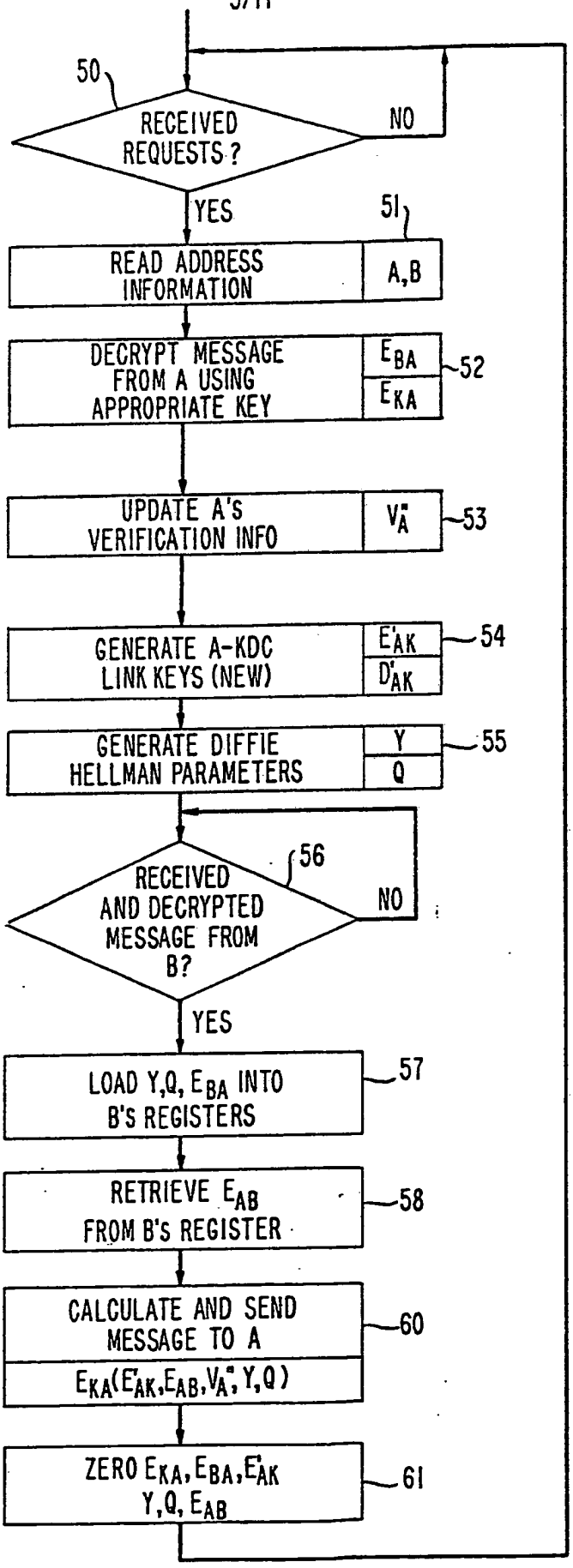


FIG. 6 BETWEEN CALLS IDLE STATE (FOR TERMINAL A)

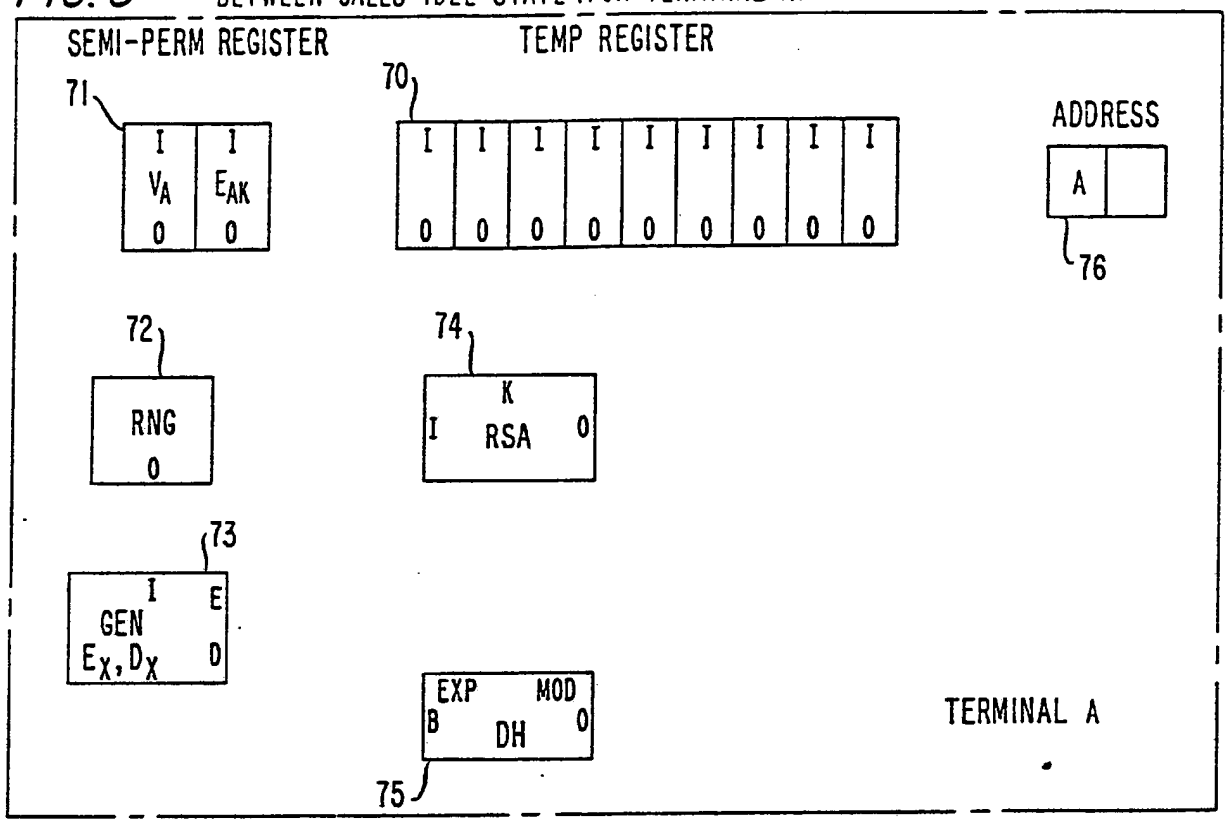


FIG. 7 START OF SECURE CALL (A TO B) SETUP - GENERATION OF KDC-A LINK KEYS

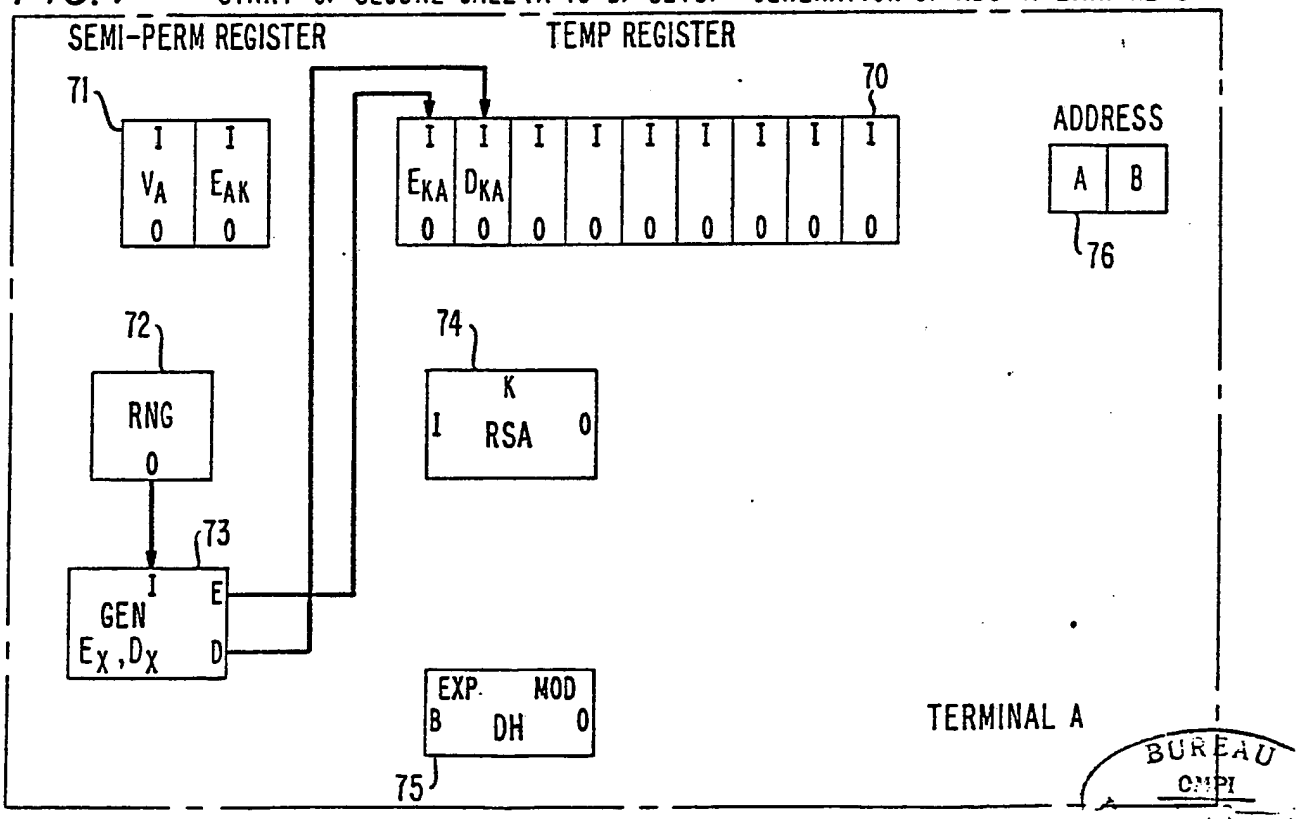


FIG. 8 GENERATION OF B-A LINK KEYS

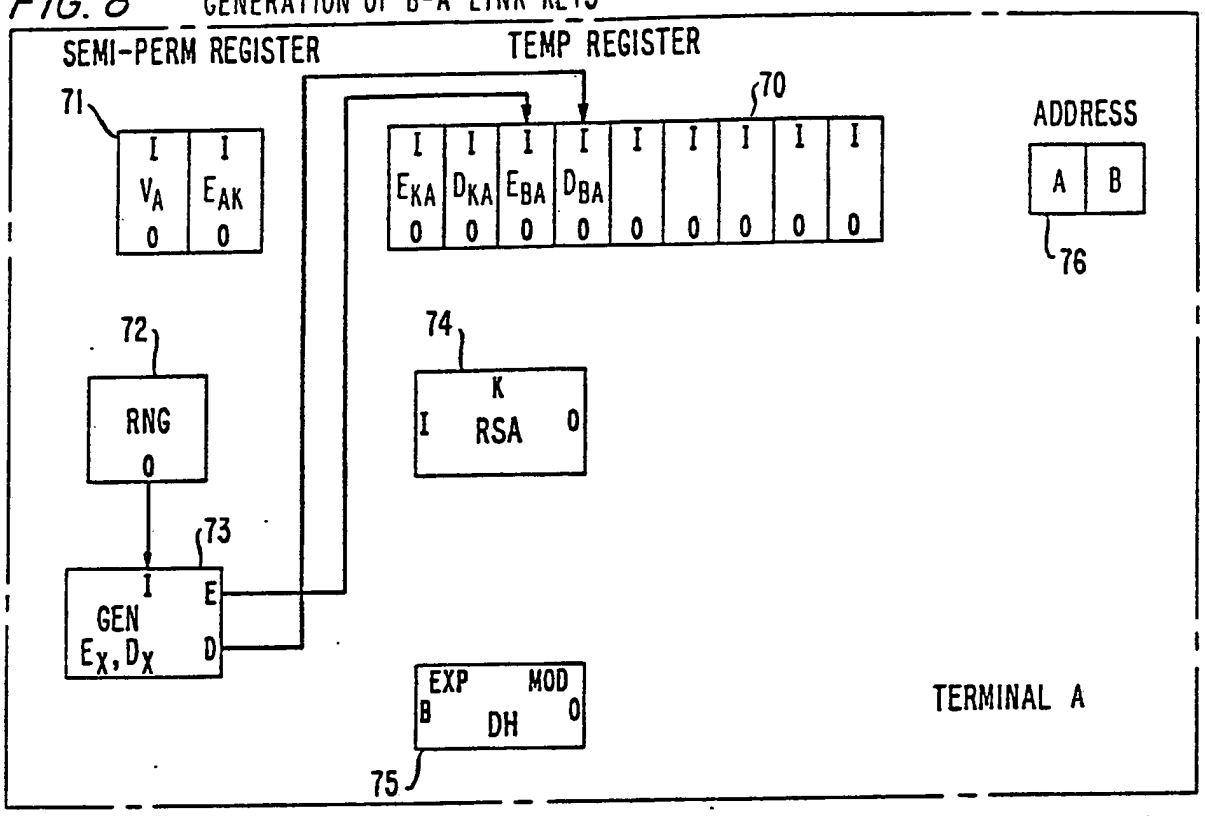


FIG. 9 FIRST STEP IN UPDATE OF VERIFICATION INFORMATION

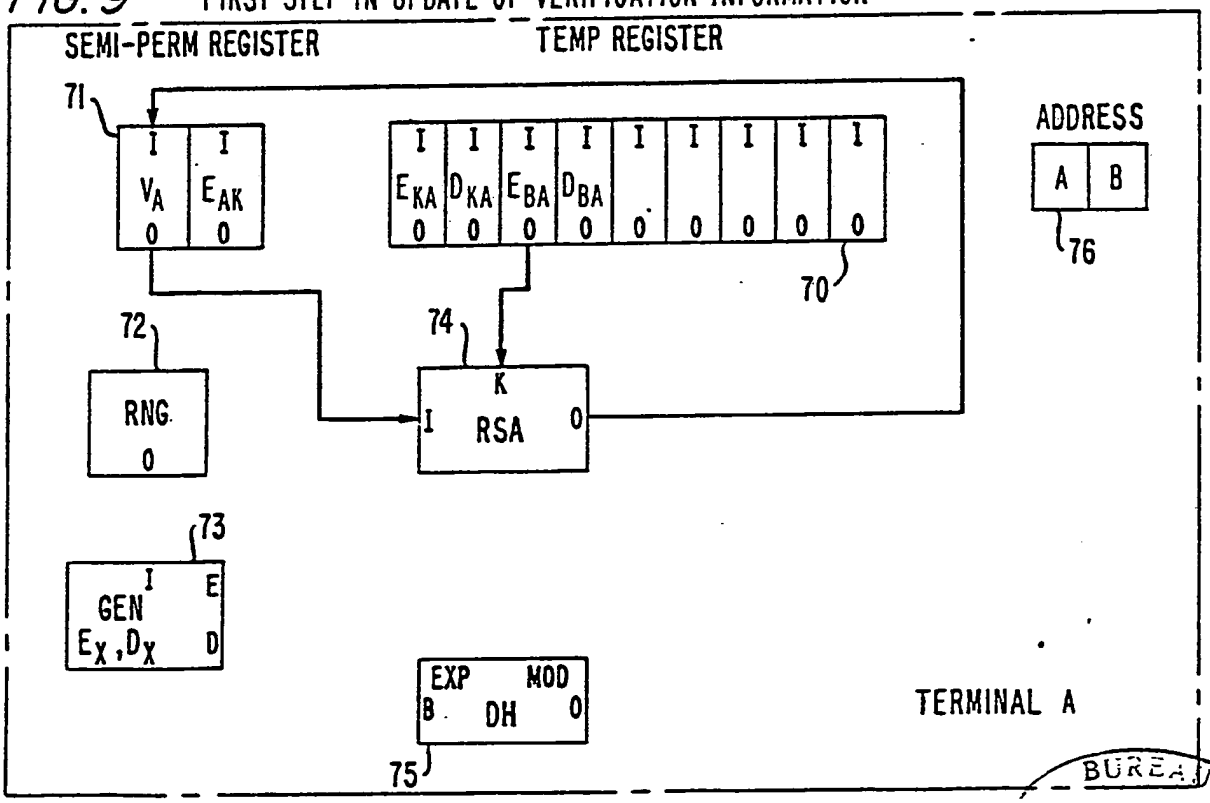


FIG. 10 SECOND STEP IN UPDATE OF VERIFICATION INFORMATION

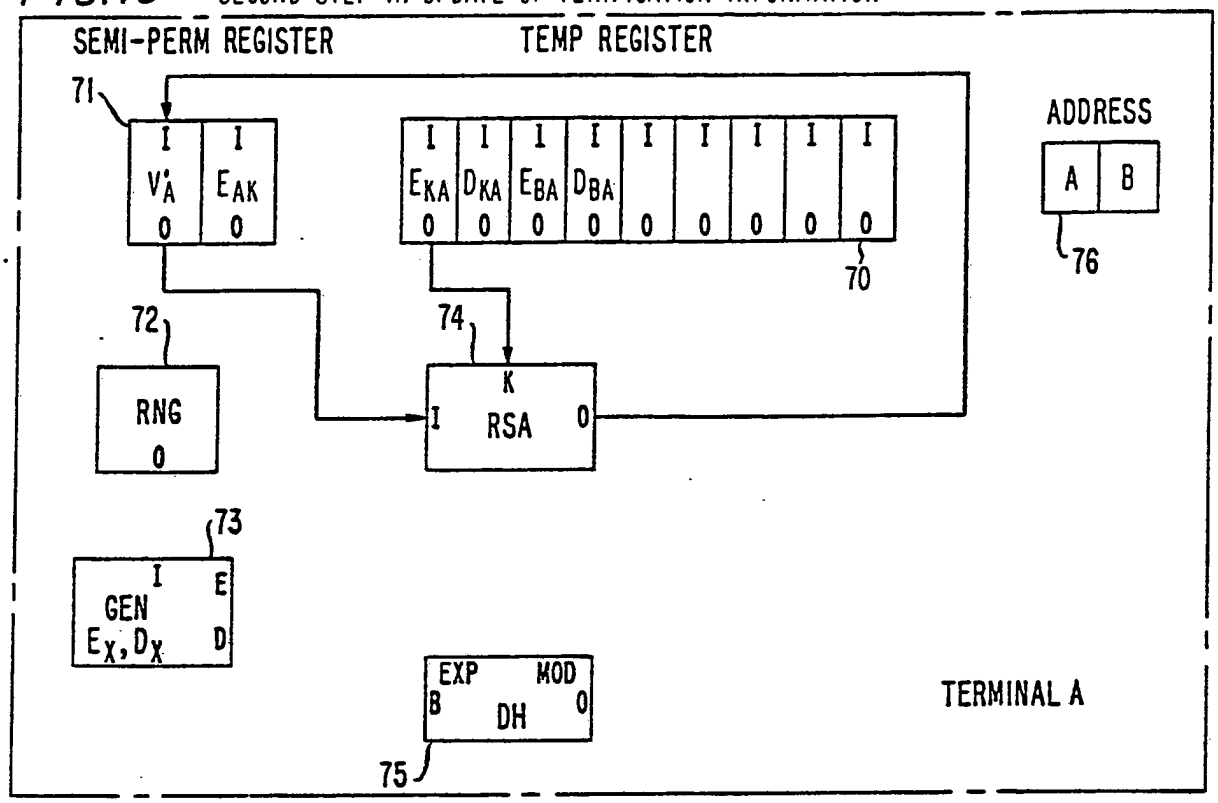


FIG. 11 COMPUTATION OF A-KDC MESSAGE

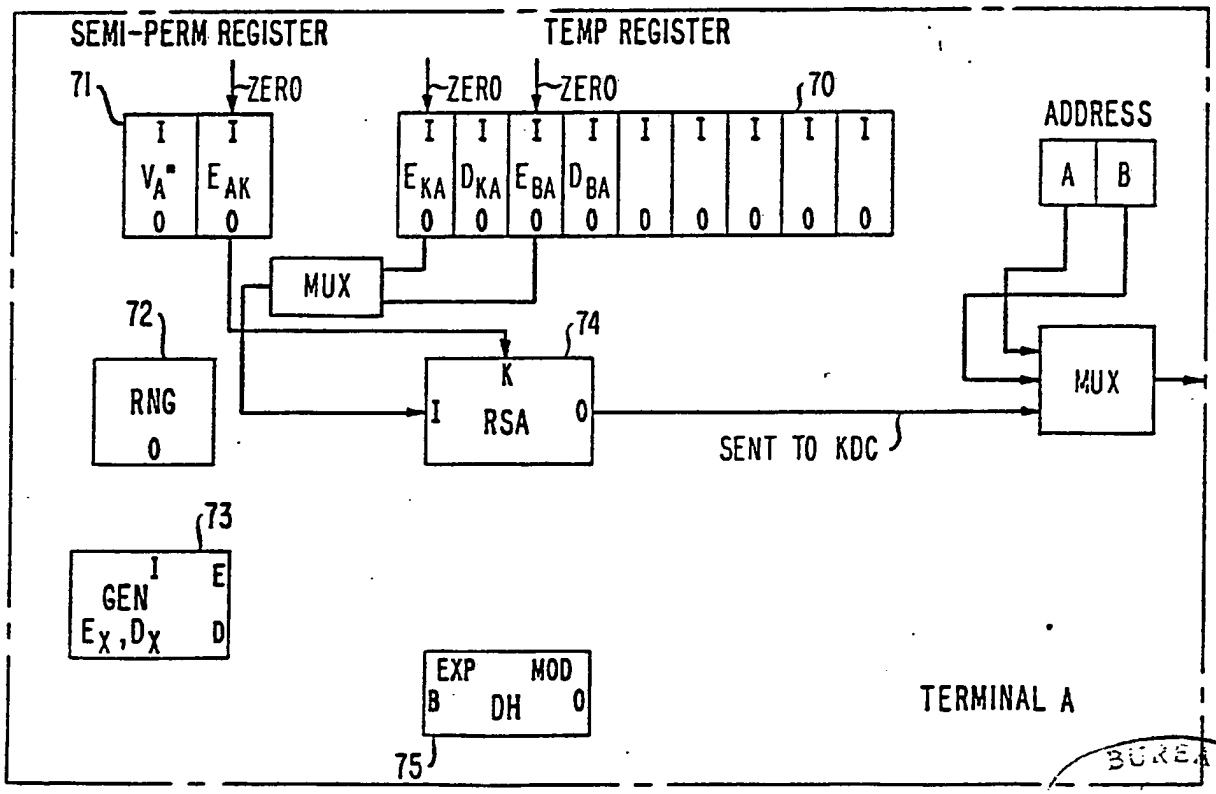


FIG. 12 IDLE STATE WHILE WAITING FOR RETURN MESSAGE FROM KDC

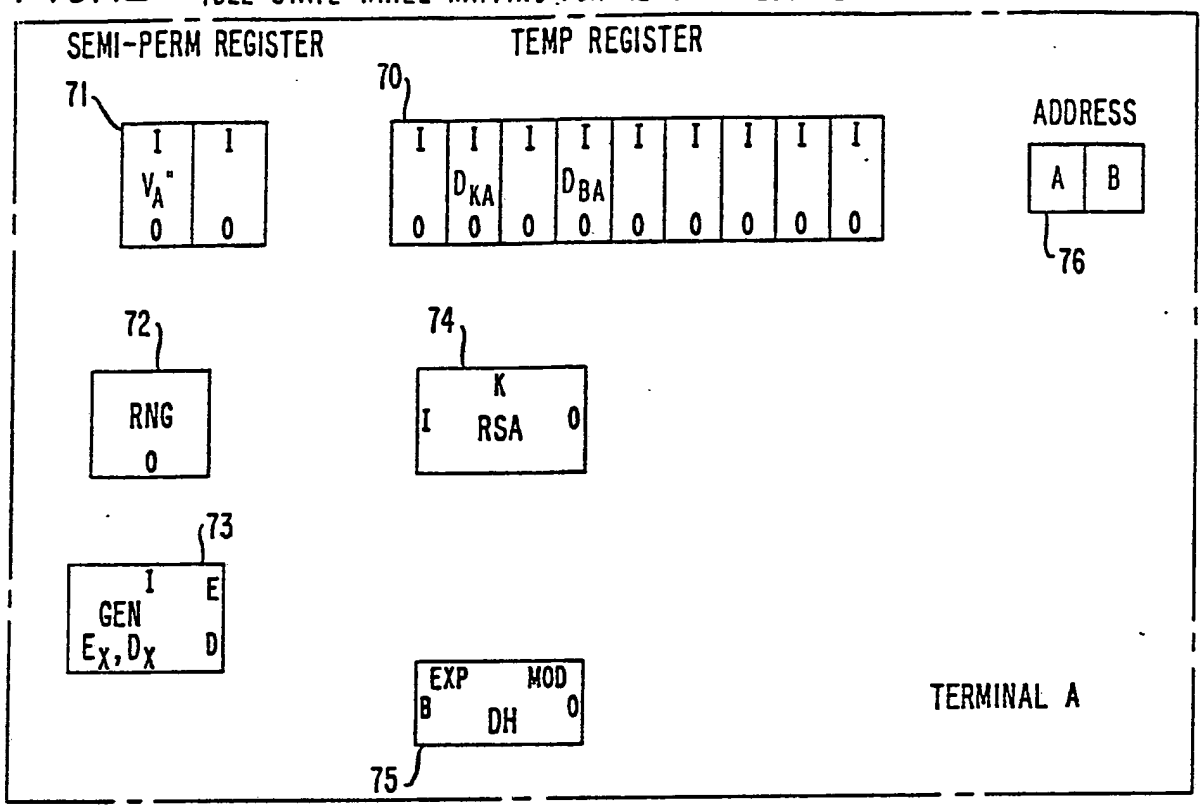


FIG. 13 DECRYPTION OF RETURN MESSAGE FROM KDC

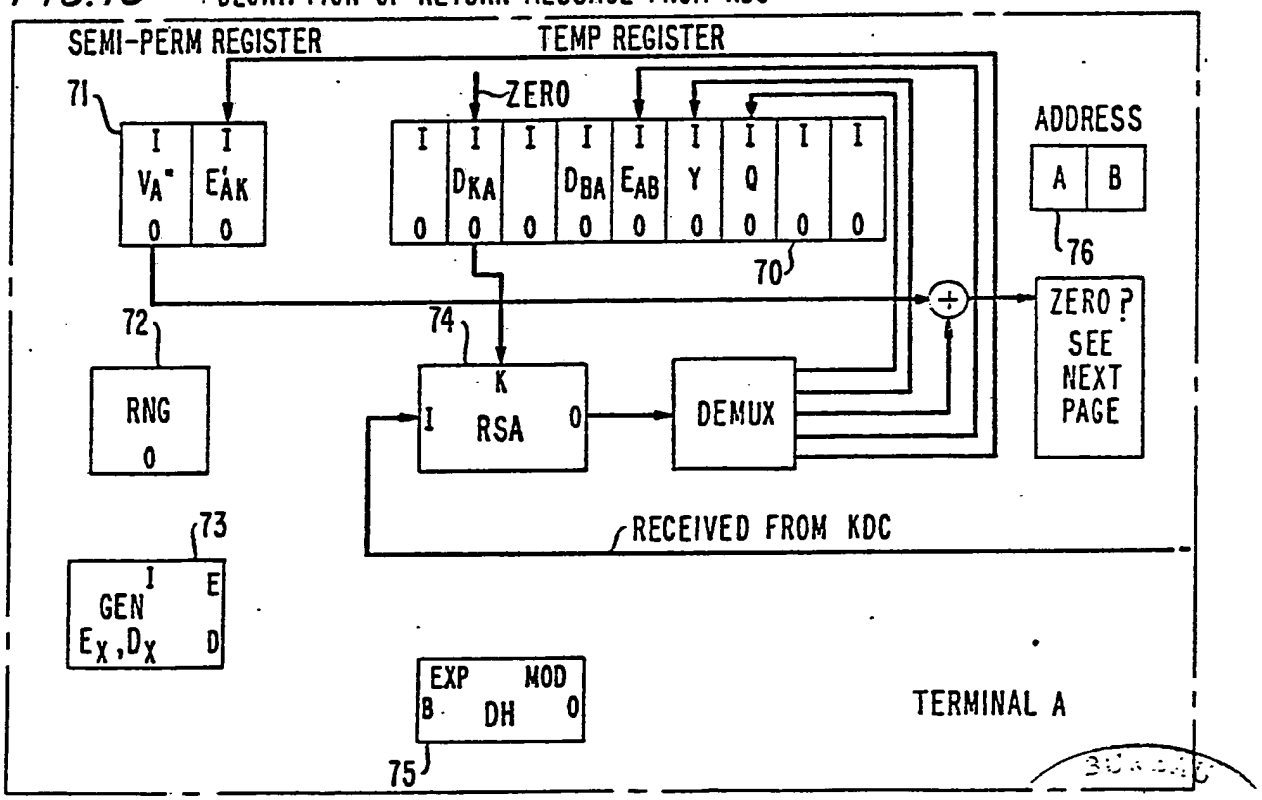


FIG. 14 VERIFICATION CHECK

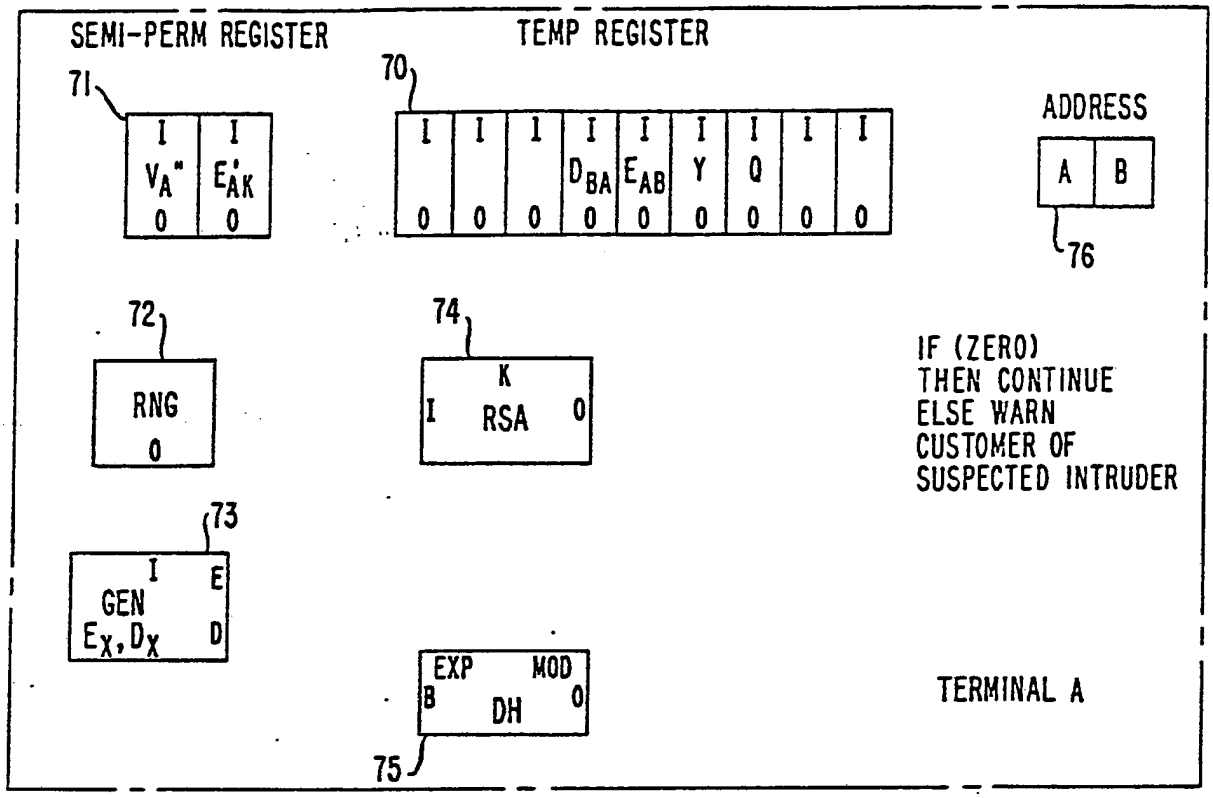


FIG. 15 START OF KEY EXCHANGE WITH B CALCULATION OF DIFFIE-HELLMAN KEYS

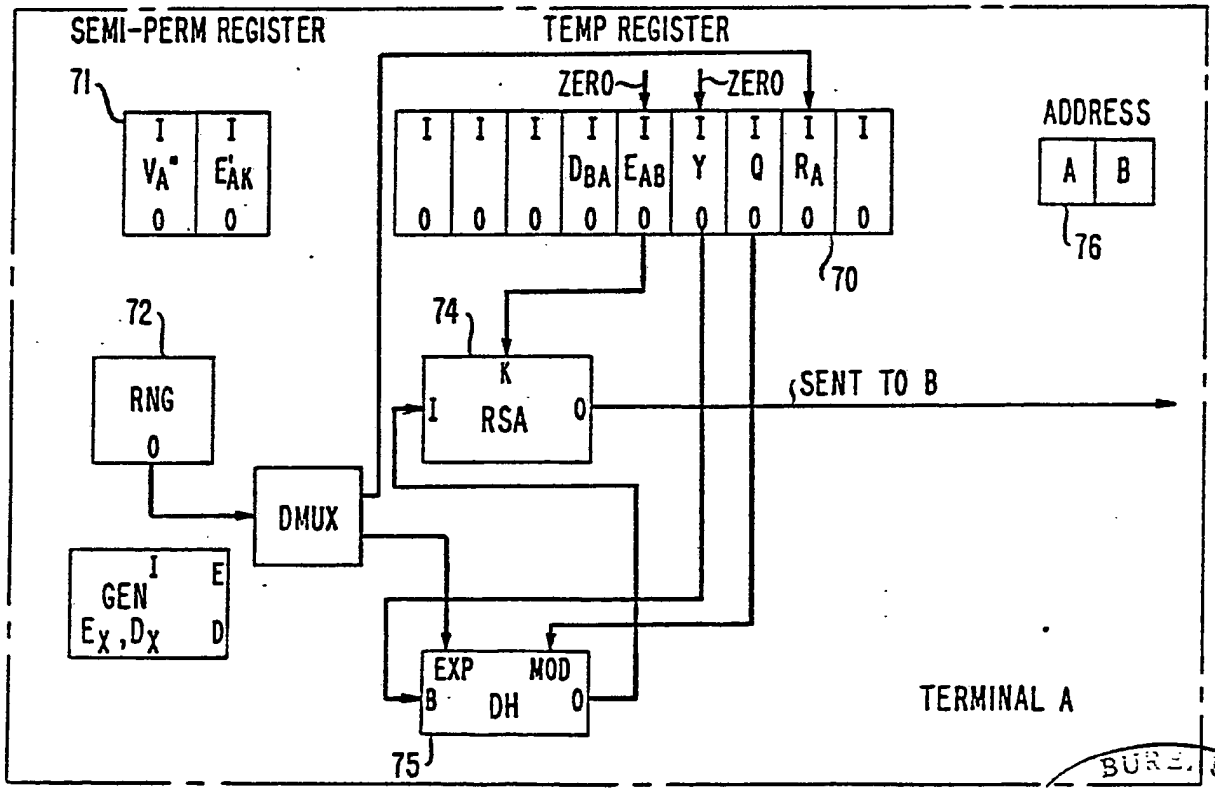


FIG. 16 IDLE WAIT STATE FOR RETURN MESSAGE FROM B

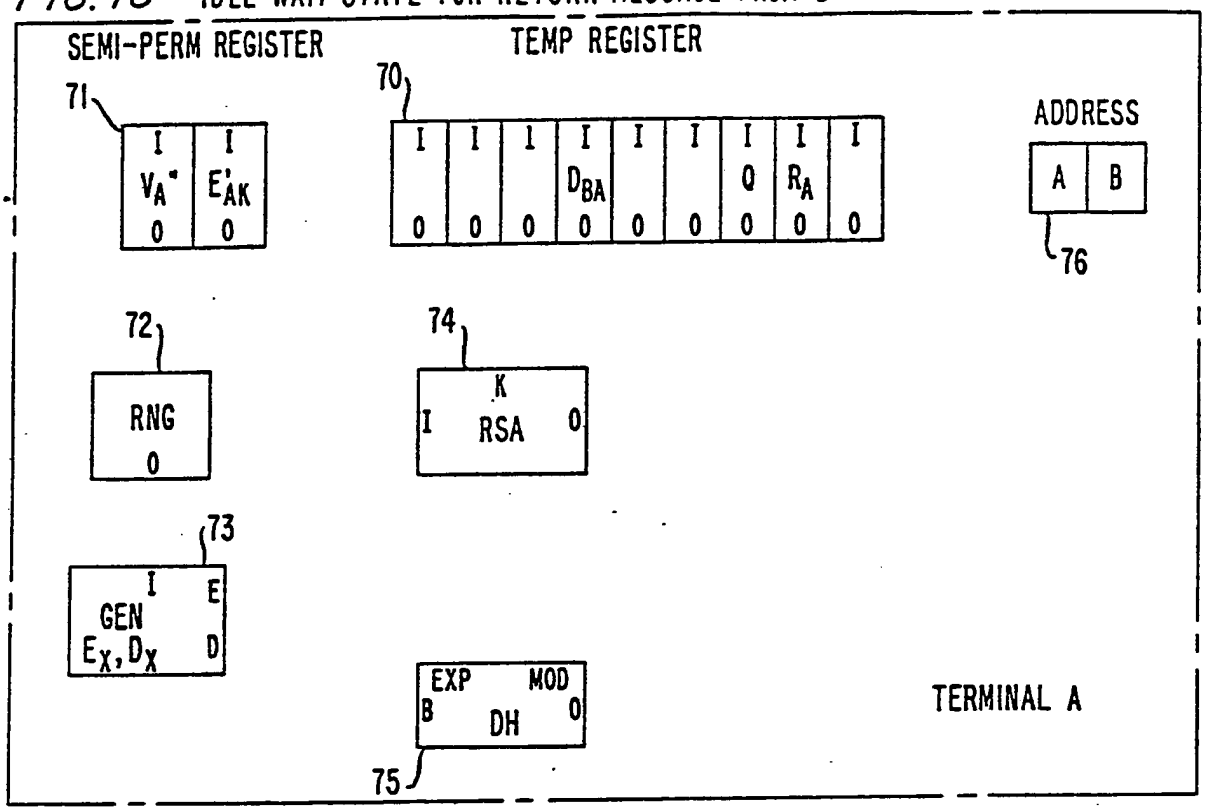


FIG. 17 DECRYPTION OF MESSAGE FROM B AND CALCULATION OF SESSION KEY-S

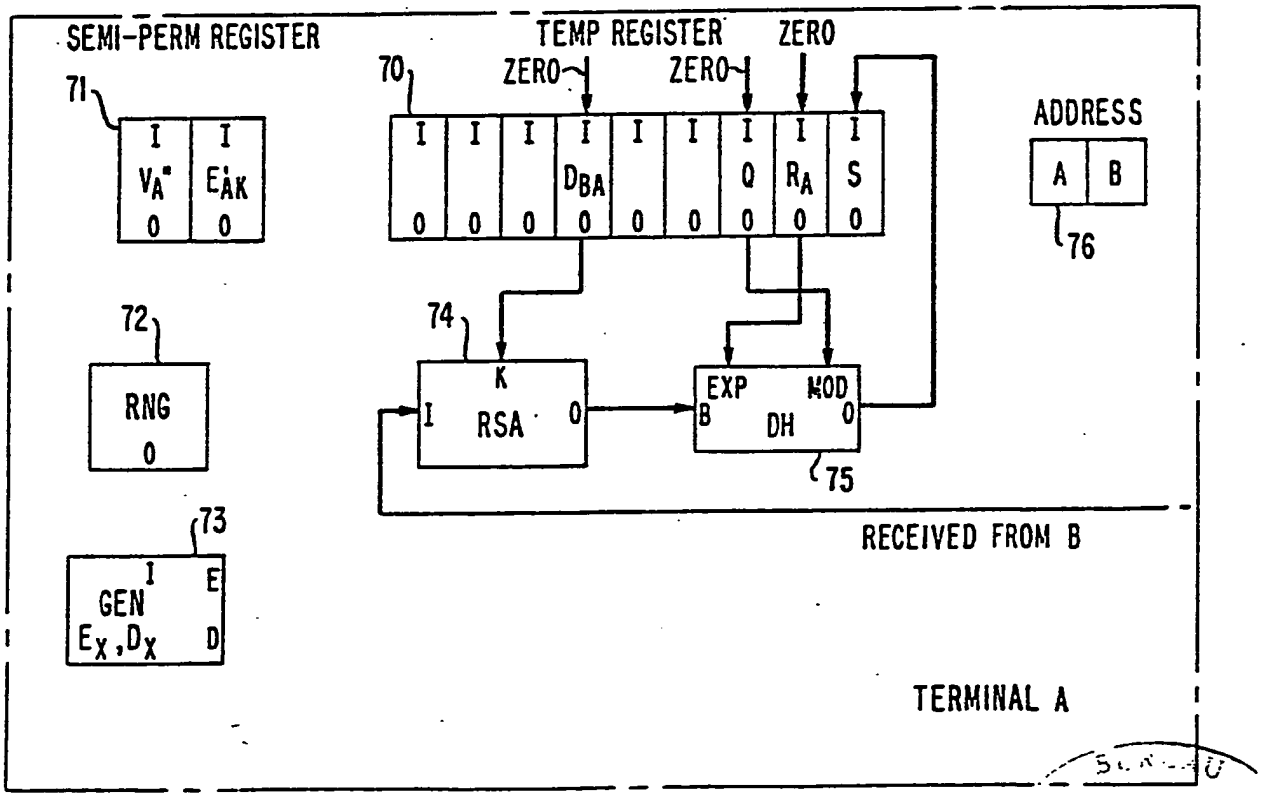


FIG. 18 KEY STORAGE DURING CALL

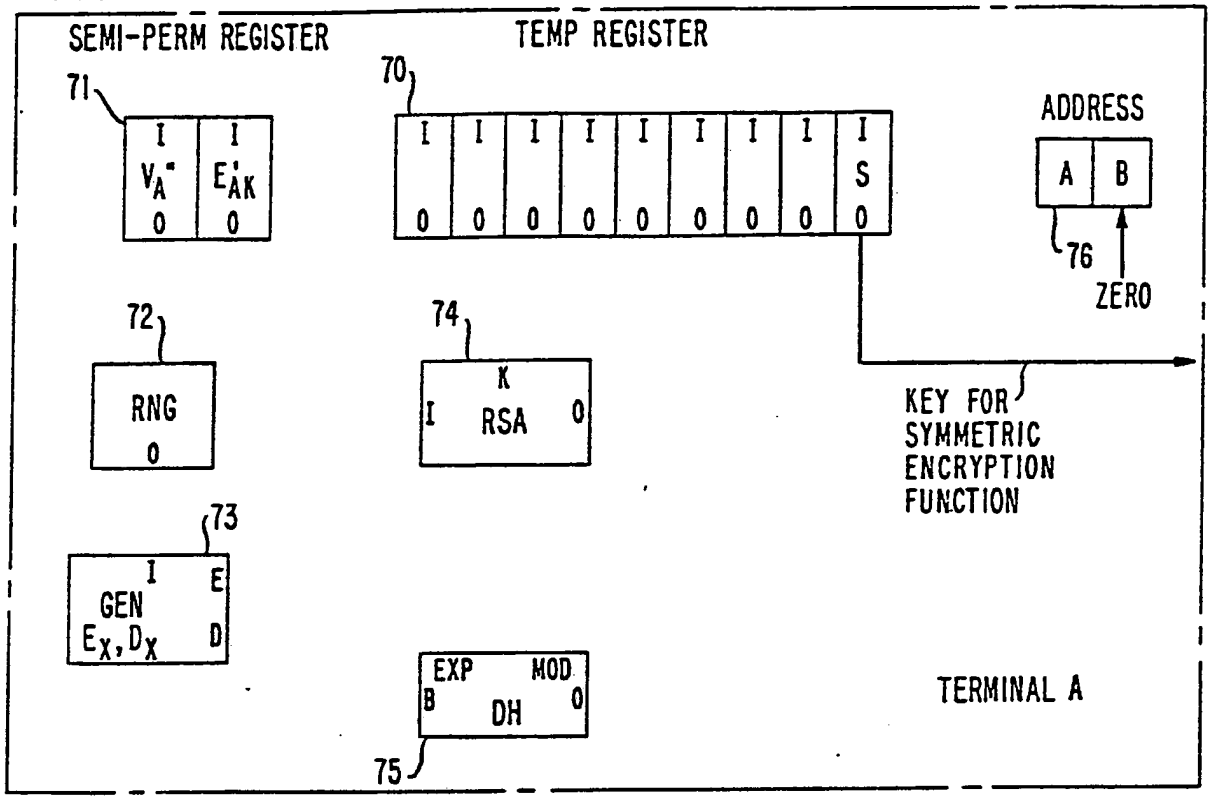


FIG. 19 IDLE STATE FOLLOWING CALL COMPLETION

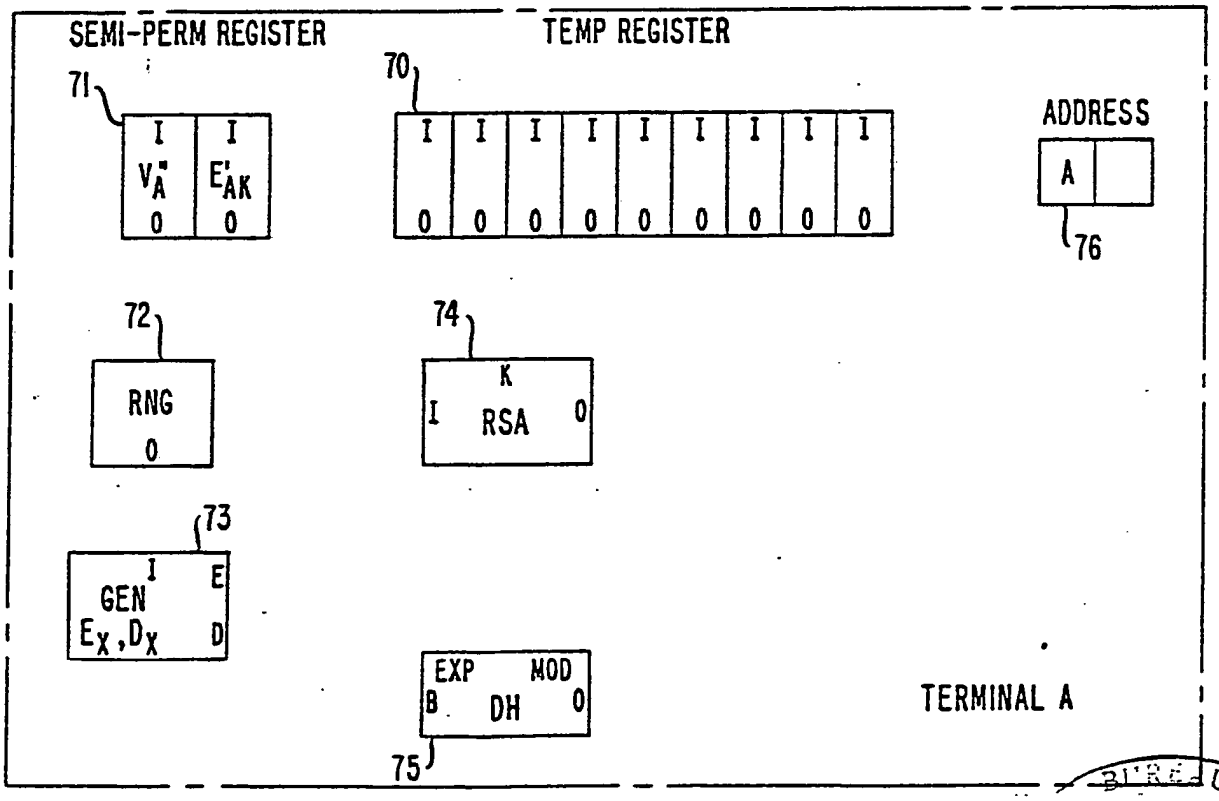


FIG. 20 IDLE STATE BETWEEN CALLS

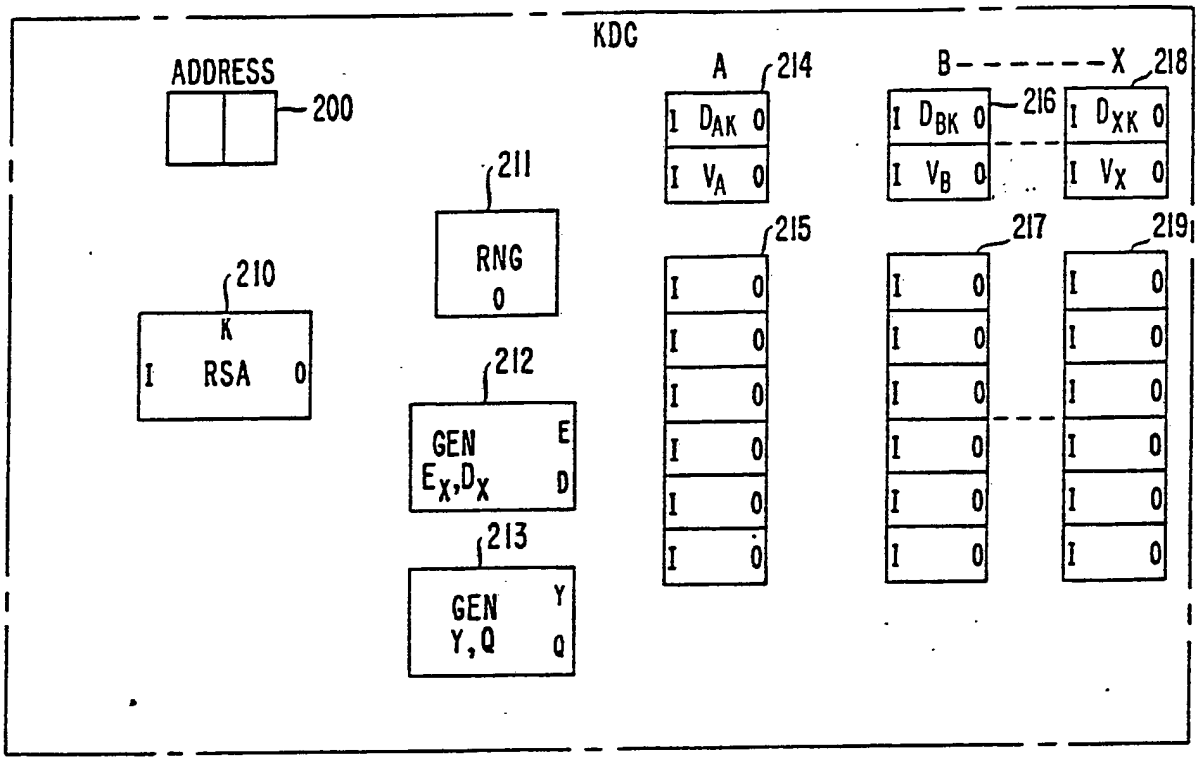


FIG. 21 DECRYPTION OF MESSAGE FROM A

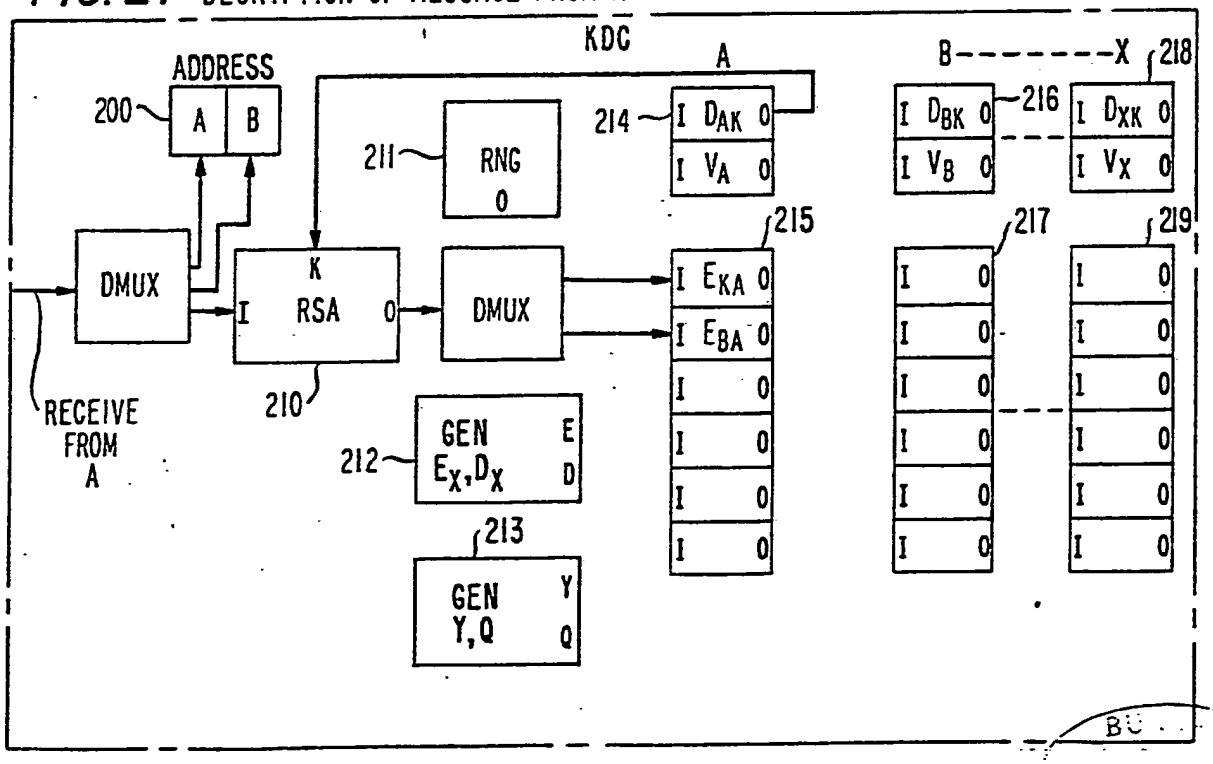


FIG. 22 FIRST STEP IN THE UPDATE OF VERIFICATION INFORMATION

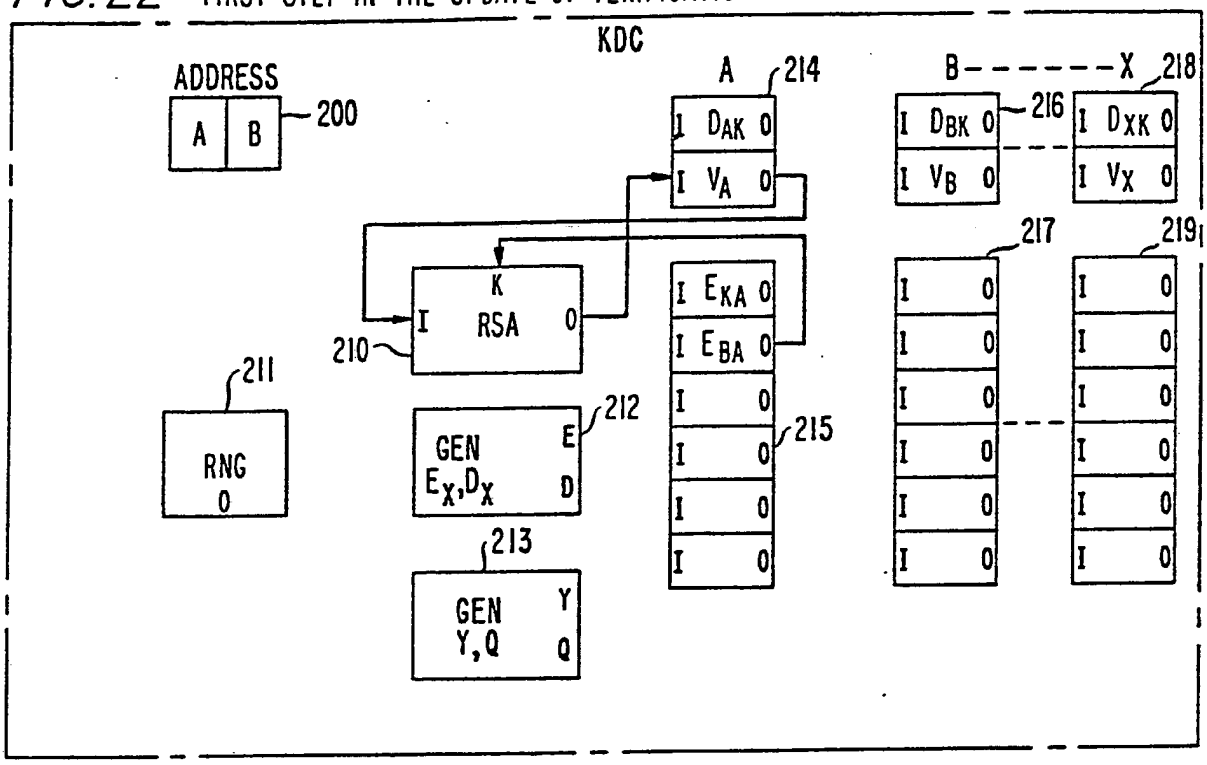


FIG. 23 SECOND STEP IN THE UPDATE OF VERIFICATION INFORMATION

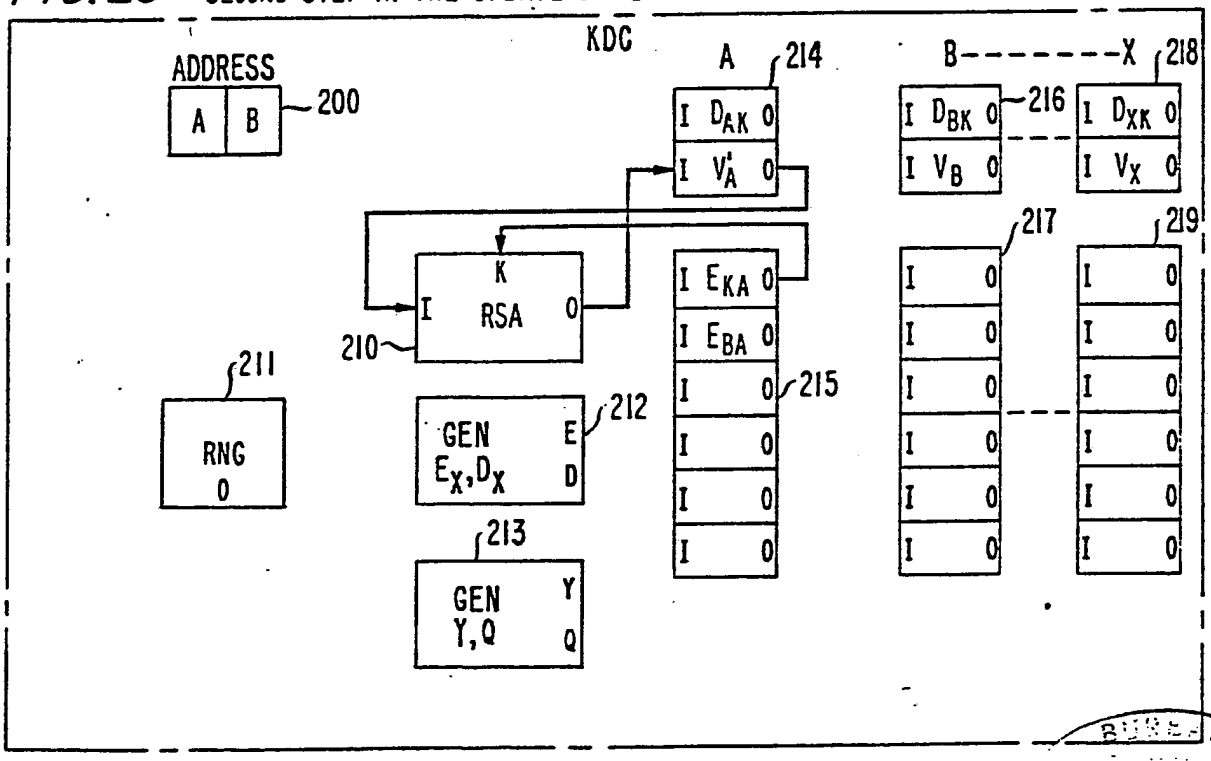


FIG. 24 GENERATION OF NEW KDC-A LINK KEYS

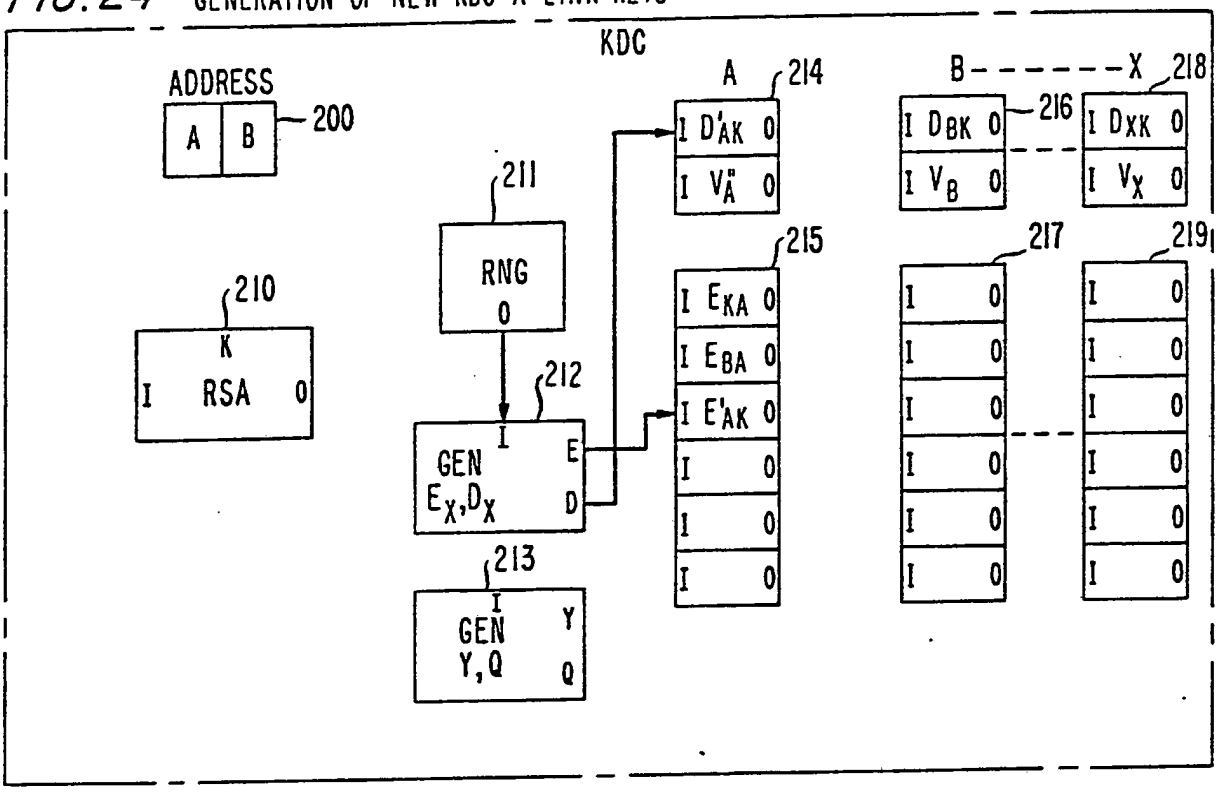


FIG. 25 GENERATION OF DIFFIE-HELLMAN ALGORITHM PARAMETERS

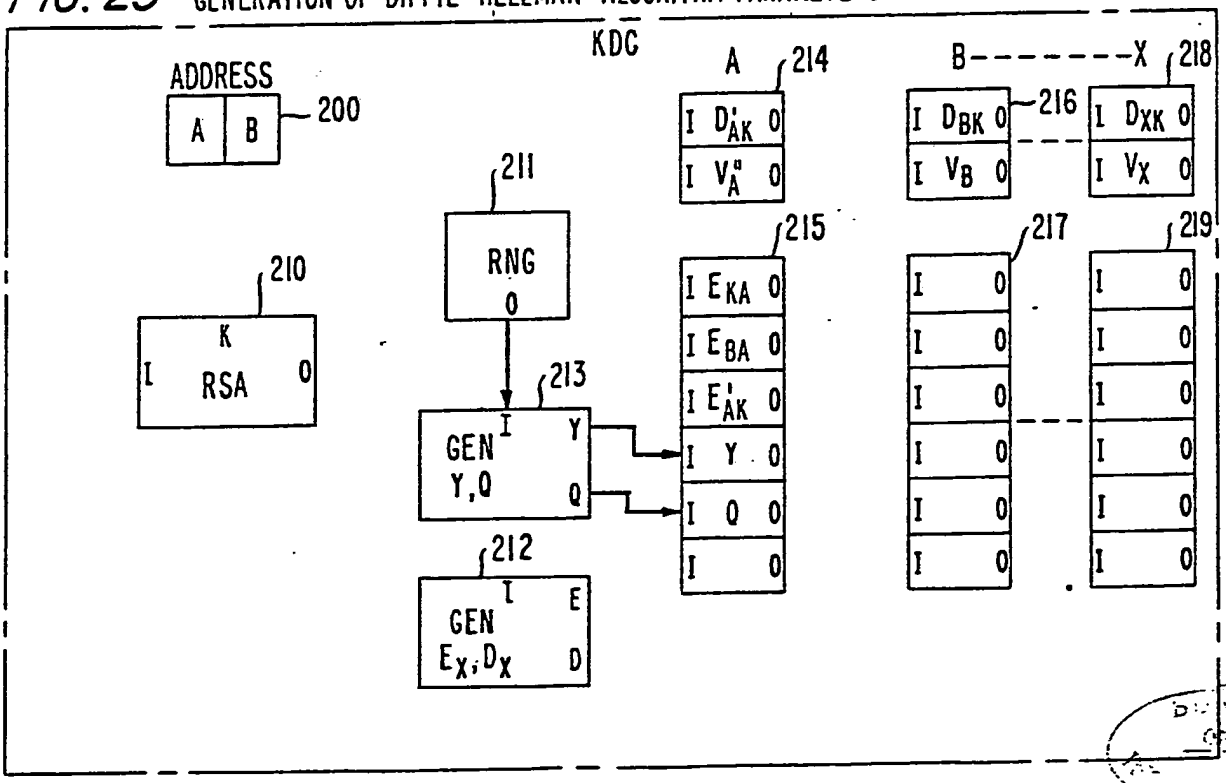


FIG. 26 INTERNAL EXCHANGE OF INFORMATION BETWEEN A & B'S REGISTERS

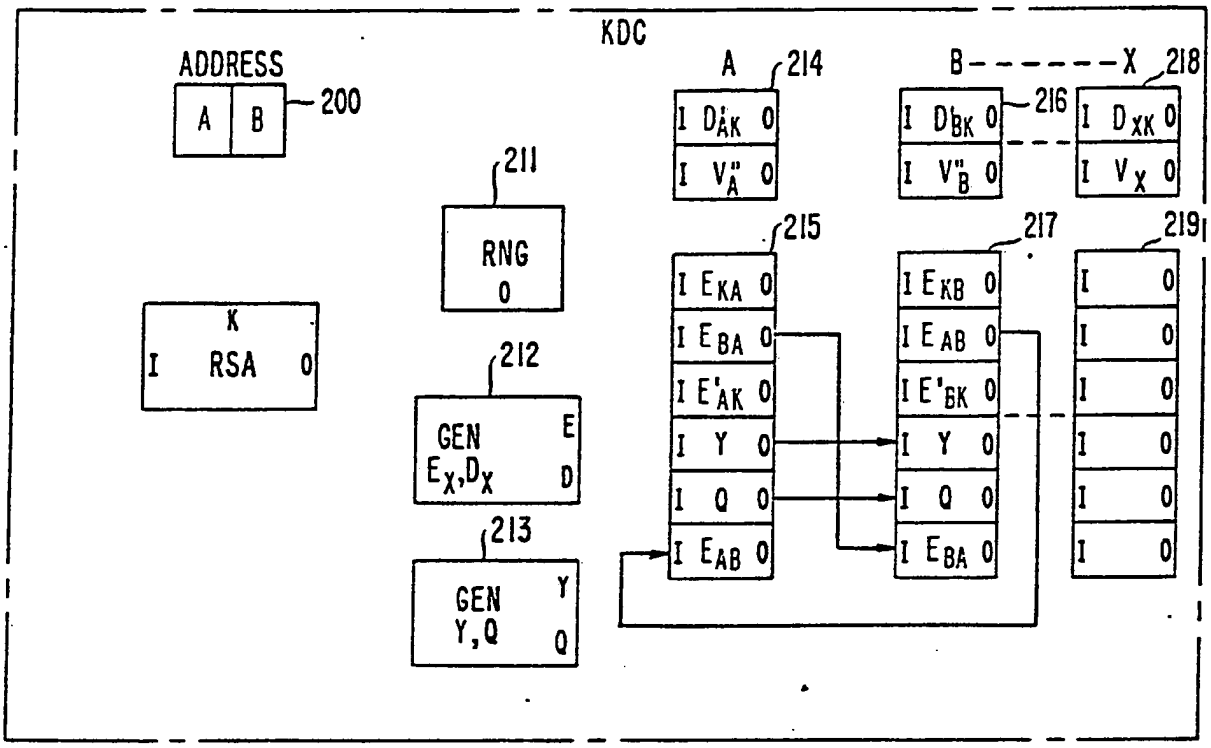


FIG. 27 COMPUTATION OF MESSAGE TO A

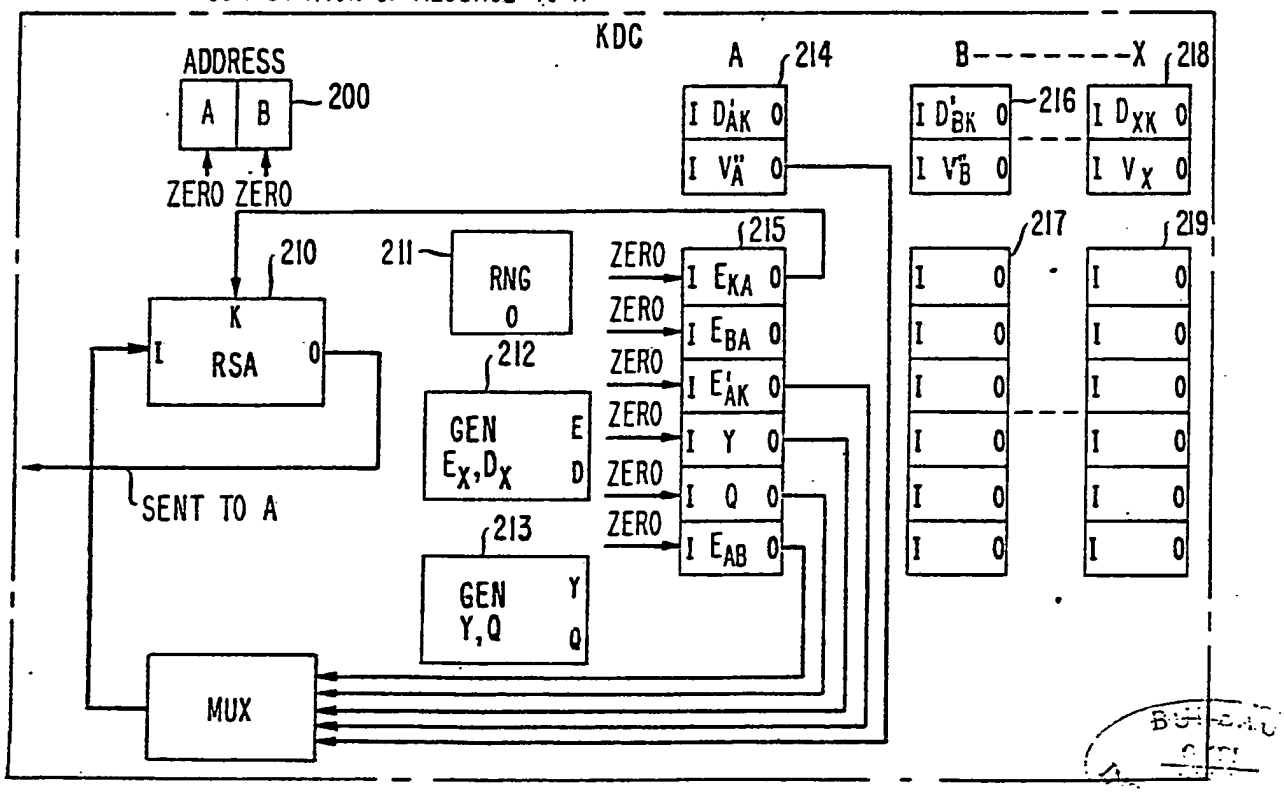
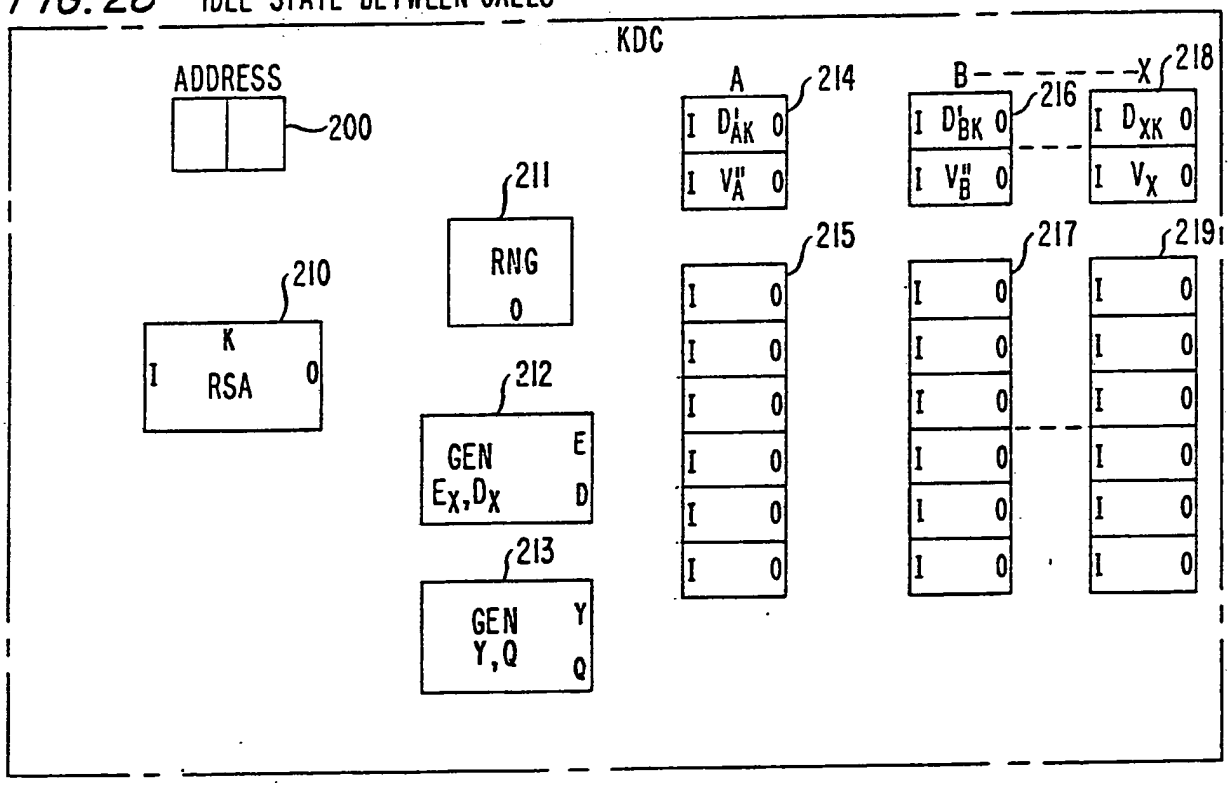


FIG. 28 IDLE STATE BETWEEN CALLS



INTERNATIONAL SEARCH REPORT

International Application No PCT/US 83/00030

I. CLASSIFICATION OF SUBJECT MATTER (if several classification symbols apply, indicate all) [3]

According to International Patent Classification (IPC) or to both National Classification and IPC

IPC[3]: H 04 L 9/00

II. FIELDS SEARCHED

Minimum Documentation Searched [4]

Classification System	Classification Symbols
IPC[3]	H 04 L 9/00; H 04 L 9/02; H 04 K 1/00; H 04 L 9/04

Documentation Searched other than Minimum Documentation to the Extent that such Documents are included in the Fields Searched [5]

III. DOCUMENTS CONSIDERED TO BE RELEVANT [14]

Category *	Citation of Document, [16] with indication, where appropriate, of the relevant passages [17]	Relevant to Claim No. [18]
A	EP, A1, 0048903 (LICENTIA) 7 April 1982 see page 2, line 23 - page 4, line 22 --	1,2
A	US, A, 4182933 (ROSENBLUM) 8 January 1980 see column 7, lines 1-21; column 8, lines 37-50; column 11, lines 17-41 cited in the application --	1
A	DATAMATION, vol. 22, no. 8, August 1976 (Barrington, US) Sykes: "Protecting data by encryption", pages 81-85, see page 84, left-hand column, lines 23-42 --	1
A	IBM-Technical Disclosure Bulletin, vol. 22, no. 2, July 1979 (New York, US) Lennon et al.: "Composite cryptographic session keys for enhanced communication security", pages 643-646, see page 643, first line to page 644, line 8 --	1,2
A	Fifth International Conference on Digital Satellite Communications, 23-26 March 1981 (New York, US) Bic et al.:	./.

* Special categories of cited documents: [15]

"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 [2]	Date of Mailing of this International Search Report [2]
20th April 1983	03 MAI 1983
International Searching Authority [1]	**Signature of Authorized Officer [20]**
EUROPEAN PATENT OFFICE	G.L.M. Kruydenberg

Form PCT/ISA/210 (second sheet) (October 1981)

FURTHER INFORMATION CONTINUED FRO.I THE SECOND SHEET

	"Privacy over digital satellite links", pages 243-249, see page 246, right-hand column, lines 32-43; page 247, left-hand column, lines 6-11; figure 3	1

V. ☐ OBSERVATIONS WHERE CERTAIN CLAIMS WERE FOUND UNSEARCHABLE [10]

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

1. ☐ Claim numbers _____, because they relate to subject matter [12] not required to be searched by this Authority, namely:

2. ☐ Claim numbers _____, 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 [13], specifically:

VI. ☐ OBSERVATIONS WHERE UNITY OF INVENTION IS LACKING [11]

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 of the international application.

2. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims of the international application for which fees were paid, specifically claims:

3. ☐ 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 claim numbers:

4. ☐ As all searchable claims could be searched without effort justifying an additional fee, the International Searching Authority did not invite payment of any additional fee.

Remark on Protest

☐ The additional search fees were accompanied by applicant's protest.

☐ No protest accompanied the payment of additional search fees.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G11B 20/00 — A2

(11) International Publication Number: **WO 97/43761**

(43) International Publication Date: 20 November 1997 (20.11.97)

(21) International Application Number: PCT/US97/08192

(22) International Filing Date: 15 May 1997 (15.05.97)

(30) Priority Data:

60/017,722	15 May 1996 (15.05.96)	US
60/018,132	22 May 1996 (22.05.96)	US
08/689,606	12 August 1996 (12.08.96)	US
08/689,754	12 August 1996 (12.08.96)	US
08/699,712	12 August 1996 (12.08.96)	US
PCT/US96/14262	4 September 1996 (04.09.96)	WO
(34) Countries for which the regional or international application was filed:		US et al.
60/037,931	14 February 1997 (14.02.97)	US

(71) Applicant *(for all designated States except US)*: INTERTRUST TECHNOLOGIES CORP. [US/US]; 460 Oakmead Parkway, Sunnyvale, CA 94086 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: SHEAR, Victor, H. [US/US]; 5203 Battery Lane, Bethesda, MD 20814 (US). SIBERT, Olin, W. [US/US]; 30 Ingleside Road, Lexington, MA 02173-2522 (US). VANWIE, David, M. [US/US]; Apartment 216, 965 E. El Camino Real, Sunnyvale, CA 94087 (US). WEBER, Robert, P. [US/US]; 215 Waverley Street #4, Menlo Park, CA 94025 (US).

(74) Agent: FARIS, Robert, W.; Nixon & Vanderhye P.C., 8th floor, 1100 North Glebe Road, Arlington, VA 22201-4714 (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, GE, HU, IL, 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, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, 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).

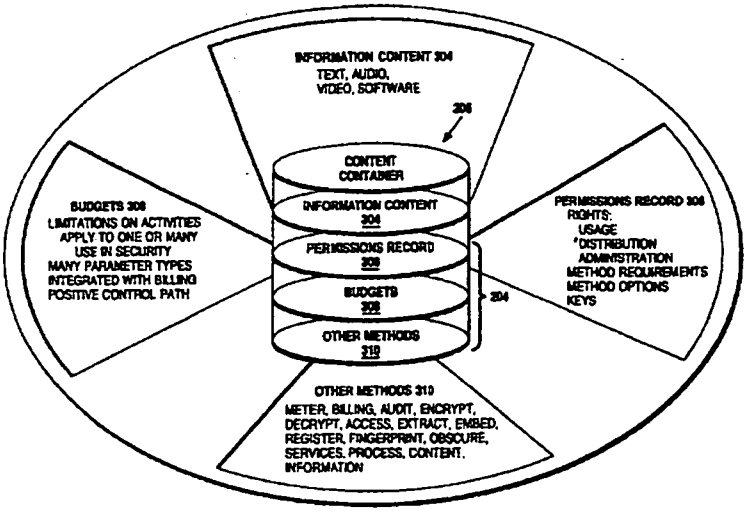
Published

Without international search report and to be republished upon receipt of that report.

(54) Title: CRYPTOGRAPHIC METHODS, APPARATUS AND SYSTEMS FOR STORAGE MEDIAELECTRONIC RIGHTS MANAGEMENT IN CLOSED AND CONNECTED APPLIANCES

(57) Abstract

A rights management arrangement for storage media such as optical digital video disks (DVDs, also called digital versatile disks) provides adequate copy protection in a limited, inexpensive mass-produceable, low-capability platform such as a dedicated home consumer disk player and also provides enhanced, more flexible security techniques and methods when the same media are used with platforms having higher security capabilities. A control object (or set) defines plural rights management rules for instance, price for performance or rules governing redistribution. Low capability platforms may enable only a subset of the control rules such as controls on copying or marking of played material. Higher capability platforms may enable all (or different subsets) of the rules. Cryptographically strong security is provided by encrypting at least some of the information carried by the media and enabling decryption based on the control set and/or other limitations. A secure "software container" can be used to protectively encapsulate (e.g., by cryptographic techniques) various digital property content (e.g., audio, video, game, etc.) and control object (i.e., set of rules) information. A standardized container format is provided for general use on/with various mediums and platforms. In addition, a special purpose container may be provided for DVD medium and appliances (e.g., recorders, players, etc.) that contains DVD program content (digital property) and DVD medium specific rules. The techniques, systems and methods disclosed herein are capable of achieving compatibility with other protection standards, such as for example, CGMA and Matsushita data protection standards adopted for DVDs. Cooperative rights management may also be provided, where plural networked rights management arrangements collectively control a rights management event on one or more of such arrangements.



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		

CRYPTOGRAPHIC METHODS, APPARATUS AND SYSTEMS FOR STORAGE MEDIA ELECTRONIC RIGHTS MANAGEMENT IN CLOSED AND CONNECTED APPLIANCES

Cross-Reference to Related Applications and Patents

The specifications and drawings of the following prior, commonly assigned published patent specifications are incorporated by reference into this patent specification:

PCT Publication No. WO 96/27155 dated 6 September 1996 entitled "Systems And Methods For Secure Transaction Management And Electronic Rights Protection", which is based on PCT application no. PCT/US96/02303 filed 13 February 1996 and U.S. patent application serial no. 08/388,107 of Ginter et al. entitled filed on February 13, 1995 (hereinafter "Ginter et al");

U.S. Patent No 4,827,508 entitled "Database Usage Metering and Protection System and Method" dated May 2, 1989;

U.S. Patent No. 4,977,594 entitled "Database Usage Metering and Protection System and Method" dated December 11, 1990;

U.S. Patent No. 5,050,213 entitled "Database Usage Metering and Protection System and Method" dated September 17, 1991; and

U.S. Patent No. 5,410,598 entitled "Database Usage Metering and Protection System and Method" dated April 25, 1995; and

European Patent No.EP 329681 entitled "Database Usage Metering and Protection System and Method" dated January 17, 1996.

In addition, the specifications and drawings of the following commonly-assigned prior-filed patent specifications are incorporated by reference into this patent application:

PCT Application No. PCT/US96/14262 filed 4 September 1996 entitled "Trusted Infrastructure Support Systems, Methods And Techniques For Secure Electronic Commerce, Electronic Transactions, Commerce Process Control And Automation, Distributed Computing, And Rights Management," which corresponds to U.S. patent application serial no. 08/699,712 filed on August 12, 1996 (hereinafter "Shear et al.");

PCT Application No. _____ filed _____, 1997 entitled "Steganographic Techniques For Securely Delivering Electronic Digital Rights Management Control Information Over Insecure Communications Channels," which corresponds to U.S. patent application serial no. 08/689,606 of Van Wie and Weber filed on August 12, 1996 (hereinafter "Van Wie and Weber"); and

PCT Application No. _____ filed _____, 1997 based on U.S. Patent Application serial no.08/689,754 entitled "Systems and Methods Using Cryptography To Protect Secure Computing Environments," of Sibert and Van Wie filed on August 12, 1996 (hereinafter "Sibert and Van Wie").

FIELD OF THE INVENTION

This invention relates to information protection techniques using cryptography, and more particularly to techniques using cryptography for managing rights to information stored on portable media -- one example being optical media such as Digital Video Disks (also known as "Digital Versatile Disks" and/or "DVDs"). This invention also relates to information protection and rights management techniques having selectable applicability depending upon, for example, the resources of the device being

used by the consumer (e.g., personal computer or standalone player), other attributes of the device (such as whether the device can be and/or typically is connected to an information network ("connected" versus "unconnected")), and available rights. This invention further relates, in part, to cooperative rights management -- where plural networked rights management arrangements collectively control a rights management event on one or more of such arrangements. Further, important aspects of this invention can be employed in rights management for electronic information made available through broadcast and/or network downloads and/or use of non-portable storage media, either independent of, or in combination with portable media.

BACKGROUND OF THE INVENTION

The entertainment industry has been transformed by the pervasiveness of home consumer electronic devices that can play video and/or audio from pre-recorded media. This transformation began in the early 1900s with the invention of the phonograph—which for the first time allowed a consumer to listen to his or her favorite band, orchestra or singer in his or her home whenever he or she wishes. The availability of inexpensive video

cassette recorders/players beginning in the early 1980s brought about a profound revolution in the movie and broadcast industries, creating an entirely new home consumer market for films, documentaries, music videos, exercise videos, etc.

The entertainment industry has long searched for optimal media for distributing content to home consumers. The original phonograph cylinders distributed by Thomas Edison and other phonograph pioneers had the advantage that they were difficult to copy, but suffered from various disadvantages such as high manufacturing costs, low resistance to breakage, very limited playback time, relatively low playback quality, and susceptibility to damage from wear, scratching or melting. Later-developed wax and vinyl disks could hold more music material but suffered from many of the same disadvantages. Magnetic tapes, on the other hand, could be manufactured very inexpensively and could hold a large amount of program material (e.g., 2, 4 or even 6 hours of video and/or audio). Such magnetic tapes could reproduce program material at relatively high quality, and were not as susceptible to damage or wearing out. However, despite the many clear advantages that magnetic tape provides over other media, the

entertainment industry has never regarded it as an ideal or optimum medium because of its great susceptibility to copying.

Magnetic tape has the very flexible characteristic that it can be relatively easily recorded on. Indeed, the process for recording a magnetic tape is nearly as straightforward as that required for playing back pre-recorded content. Because of the relative ease by which magnetic tape can be recorded, home consumer magnetic tape equipment manufacturers have historically provided dual mode equipment that can both record and play back magnetic tapes. Thus, home audio and video tape players have traditionally had a "record" button that allows a consumer to record his or her own program material on a blank (un-recorded) magnetic tape. While this recording ability has given consumers additional flexibility (e.g., the ability to record a child's first words for posterity, and the ability to capture afternoon soap operas for evening viewing), it has unfortunately also been the foundation of an illegal multi-billion dollar content pirating industry that produces millions of illegal, counterfeit copies every year. This illegal pirating operation—which is international in scope—leeches huge amounts of revenue every year from the world's major

entertainment content producers. The entertainment industry must pass along these losses to honest consumers—resulting in higher box office prices, and higher video and audio tape sales and rental prices.

In the mid 1980s, the audio entertainment industry developed the optical compact disk as an answer to some of these problems. The optical compact disk—a thin, silvery plastic platter a few inches in diameter—can hold an hour or more of music or other audio programming in digital form. Such disks were later also used for computer data. The disk can be manufactured very inexpensively, and provides extremely high quality playback that is resistant to noise because of the digital techniques used to record and recover the information. Because the optical disk can be made from plastic, it is light weight, virtually unbreakable, and highly resistant to damage from normal consumer handling (unlike the prior vinyl records that were easily scratched or worn down even by properly functioning phonographs). And, because recording on an optical disk is, so far, significantly more difficult than playing back an optical disk, home consumer equipment providing both recording and playback capabilities is unlikely, in

the near future, to be as cost-effective as play-only equipment—greatly reducing the potential for illicit copying. Because of these overwhelming advantages, the music industry has rapidly embraced the new digital compact disk technology—virtually replacing older audio vinyl disk media within the space of a few short years.

Indeed, the threat of widespread and easy unauthorized copying in the absence of rights management technologies apparently has been an important contributing factor to the demise of digital audio tape (DAT) as a media for music distribution and, more importantly, home audio recording. Rightsholders in recorded music vigorously opposed the widespread commercialization of inexpensive DAT technology that lacked rights management capabilities since the quality of the digital recording was completely faithful to the digital source on, for example, music CDs. Of course, the lack of rights management was not the only factor at work, since compared with optical media, tape format made random access difficult, for example, playing songs out of sequence.

The video entertainment industry is on the verge of a revolution similar to that wrought by music CDs based on movies in digital format distributed on high capacity read-only optical media. For example, digital optical disk technology has advanced to the point where it is now possible to digitally record, among other things, a full length motion picture (plus sound) on one side of a 5" plastic optical disk. This same optical disk can accommodate multiple high-quality digital audio channels (e.g., to record multi-channel "sensurround" sound for home theaters and/or to record film dialog in multiple different languages on the same disk). This same technology makes it possible to access each individual frame or image of a movie for still image reproduction or—even more exciting—to provide an unprecedented "random access" playback capability that has never before existed in home consumer equipment. This "random access" playback could be used, for example, to delete violence, foul language or nudity at time of playback so that parents could select a "PG" playback version of an "R" rated film at the press of a button. The "random access" capability also has exciting possibilities in terms of allowing viewers to interact with the pre-recorded content (e.g.,

allowing a health enthusiast to select only those portions of an exercise video helpful to a particular day's workout). See, for example, "Applications Requirements for Innovative Video Programming," DVD Conference Proceedings (Interactive Multimedia Association, 19-20 October 1995, Sheraton Universal Hotel, Universal City, California).

Non-limiting examples of the DVD family of optical media include:

- DVD (Digital Video Disk, Digital Versatile Disk), a non-limiting example of which includes consumer appliances that play movies recorded on DVD disks;

- DVD-ROM (DVD-Read Only Memory), a non-limiting example of which includes a DVD read-only drive and disk connected to a computer or other appliance;

- DVD-RAM (DVD Random Access Memory), a non-limiting example of which includes a read/write drive and optical media in, for example, consumer appliances for home recording and in a computer or other appliance

for the broadest range of specific applications; and

- Any other high capacity optical media presently known or unknown.

"DVDs" are, of course, not limited to use with movies. Like CDs, they may also be used for other kinds of information, for example:

- sound recordings
- software
- databases
- games
- karaoke
- multimedia
- distance learning
- documentation
- policies and manuals

- any kind of digital data or other information
- any combination of kinds of digital data or other information
- any other uses presently known or unknown.

The broad range of DVD uses presents a technical challenge: how can the information content distributed on such disks, which might be any kind or combination of video, sound, or other data or information broadly speaking, be adequately protected while preserving or even maximizing consumer flexibility? One widely proposed requirement for the new technology(mainly within the context of video), is, to the extent copying is permitted at all, to either: (a) allow a consumer to make a first generation copy of the program content for their own use, but prevent the consumer from making "copies of copies", or multi-generational copies of a given property (thus keeping honest people honest); or (b) to allow unlimited copying for those properties that rightsholders do not wish to protect against copying, or which consumers have made themselves.

However, providing only such simplistic and limited copy protection in a non-extensible manner may turn out to be extremely shortsighted—since more sophisticated protection and/or rights management objectives (e.g., more robust and selective application of copy protection and other protection techniques, enablement of pay-per-view models, the ability of the consumer to make use of enhanced functionality such as extracting material or interactivity upon paying extra charges, and receiving credit for redistribution, to name a few) could be very useful now or in the future. Moreover, in optimally approaching protection and rights management objectives, it is extremely useful to take differing business opportunities and threats into account that may relate to information delivered via DVD media, for example, depending upon available resources of the device and/or whether the device is connected or unconnected.

More sophisticated rights management capabilities will also allow studios and others who have rights in movies and/or sound recordings to better manage these important assets, in one example, to allow authorized parties to repurpose pieces of digital film, video and/or audio, whether specific and/or arbitrary pieces,

to create derivative works, multimedia games, in one non-limiting example. Solutions proposed to date for protecting DVD content have generally focused solely on limited copy protection objectives and have failed to adequately address or even recognize more sophisticated rights management objectives and requirements. More specifically, one copy protection scheme for the initial generation of DVD appliances and media is based on an encryption method developed initially by Matsushita and the simple CGMA control codes that indicate permitted copying: a one-generation copy, no copies, or unlimited copying.

SUMMARY OF THE INVENTIONS

Comprehensive solutions for protecting and managing information in systems that incorporate high capacity optical media such as DVD require, among other things, methods and systems that address two broad sets of problems: (a) digital to analog conversion (and vice versa); and (b) the use of such optical media in both connected and unconnected environments. The inventions disclosed herein address these and other problems. For example, in the context of analog to digital conversion (and vice versa), it is contemplated that, in accordance with the present

inventions, at least some of the information used to protect properties and/or describe rights management and/or control information in digital form could also be carried along with the analog signal. Devices that convert from one format and/or medium to another can, for example, incorporate some or all of the control and identifying information in the new context(s), or at least not actively delete such information during the conversion process. In addition, the present inventions provide control, rights management and/or identification solutions for the digital realm generally, and also critically important technologies that can be implemented in consumer appliances, computers, and other devices. One objective of the inventions is to provide powerful rights management techniques that are useful in both the consumer electronics and computer technology markets, and that also enable future evolution of technical capabilities and business models. Another non-limiting objective is to provide a comprehensive control, rights management and/or identification solution that remains compatible, where possible, with existing industry standards for limited function copy protection and for encryption.

The present inventions provide rights management and protection techniques that fully satisfy the limited copy protection objectives currently being voiced by the entertainment industry for movies while also flexibly and extensibly accommodating a wide range of more sophisticated rights management options and capabilities.

Some important aspects of the present inventions (that are more fully discussed elsewhere in this application) include:

- Selection of control information associated with information recorded on DVD media (for example, rules and usage consequence control information, that comprise non-limiting example elements of a Virtual Distribution Environment (VDE)) that is based at least in part on class of appliance, for example, type of appliance, available resources and/or rights;

- Enabling such selected control information to be, at least in part, a subset of control information used on other appliances and/or classes of appliance, or completely different control information;

- Protecting information output from a DVD device, such as applying rights management techniques disclosed in Ginter et al. and the present application to the signals transmitted using an IEEE 1394 port (or other serial interface) on a DVD player;

- Creation of protected digital content based on an analog source;

- Reflecting differing usage rights and/or content availability in different countries and/or regions of the world;

- Securely managing information on DVD media such that certain portions may be used on one or more classes of appliance (e.g., a standalone DVD player), while other portions may be used on the same or different classes of appliance (e.g., a standalone DVD player or a PC);

- Securely storing and/or transmitting information associated with payment, auditing, controlling and/or otherwise managing content recorded on DVD media, including techniques related to those disclosed in Ginter et al. and in Shear et al.;

- Updating and/or replacing encryption keys used in the course of appliance operation to modify the scope of information that may be used by appliances and/or classes of appliances;

- Protecting information throughout the creation, distribution, and usage process, for example, by initially protecting information collected by a digital camera, and continuing protection and rights management through the editing process, production, distribution, usage, and usage reporting.

- Allowing "virtual rights machines," consisting of multiple devices and/or other systems that participate and work together in a permanently or in a temporarily connected network to share some or all of the rights management for a single and/or multiple nodes including, for example, allowing resources available in plural such devices and/or other systems, and/or rights associated with plural parties and/or groups using and/or controlling such devices and/or other systems, to be employed in concert (according to rights related rules and controls) so as to govern one or more electronic

events on any one or more of such devices and/or other systems, such event governance including, for example: viewing, editing, subsetting, anthologizing, printing, copying, titling, extracting, saving, and/or redistributing rights protected digital content.

- Allowing for the exchange of rights among peer-to-peer relating devices and/or other systems, wherein such devices and/or other systems participate in a temporary or permanently connected network, and wherein such rights are bartered, sold for currency, and/or otherwise exchanged for value and/or consideration where such value and/or consideration is exchanged between such peer-to-peer participating commercial and/or consumer devices and/or other systems.

General Purpose DVD/Cost-effective Large Capacity Digital Media Rights Protection and Management

The inventions described herein can be used with any large capacity storage arrangement where cost-effective distribution media is used for commercial and/or consumer digital information delivery and DVD, as used herein, should be read to include any such system.

Copy protection and rights management are important in practical DVD systems and will continue to be important in other large capacity storage, playback, and recording systems, presently known or unknown, in the future. Protection is needed for some or all of the information delivered (or written) on most DVD media. Such protection against copying is only one aspect of rights management. Other aspects involve allowing rightsholders and others to manage their commercial interests (and to have them enforced, potentially at a distance in time and/or space) regardless of distribution media and/or channels, and the particular nature of the receiving appliance and/or device. Such rights management solutions that incorporate DVD will become even more significant as future generations of recordable DVD media and appliances come to market. Rightsholders will want to maintain and assert their rights as, for example, video, sound recordings, and other digital properties are transmitted from one device to another and as options for recording become available in the market.

The apparent convergence between consumer appliances and computers, increasing network and modem speeds, the declining cost of computer power and bandwidth, and the

increasing capacity of optical media will combine to create a world of hybrid business models in which digital content of all kinds may be distributed on optical media played on at least occasionally connected appliances and/or computers, in which the one-time purchase models common in music CDs and initial DVD movie offerings are augmented by other models, for example, lease, pay per view, and rent to own, to name just few. Consumers may be offered a choice among these and other models from the same or different distributors and/or other providers. Payment for use may happen over a network and/or other communications channel to some payment settlement service. Consumer usage and audit information may flow back to creators, distributors, and/or other participants. The elementary copy protection technologies for DVD now being introduced cannot support these and other sophisticated models.

As writable DVD appliances and media become available, additional hybrid models are possible, including, for example, the distribution of digital movies over satellite and cable systems. Having recorded a movie, a consumer may elect a lease, rental, pay-per-view, or other model if available. As digital television

comes to market, the ability of writable DVDs to make faithful copies of on-air programming creates additional model possibilities and/or rights management requirements. Here too, simplistic copy protection mechanisms currently being deployed for the initial read-only DVD technologies will not suffice.

Encryption Is A Means, Not An End

Encryption is useful in protecting intellectual properties in digital format, whether on optical media such as DVD, on magnetic media such as disk drives, in the active memory of a digital device and/or while being transmitted across computer, cable, satellite, and other kinds of networks or transmission means. Historically, encryption was used to send secret messages. With respect to DVD, a key purpose of encryption is to require the use of a copy control and rights management system in order to ensure that only those authorized to do so by rightsholders can indeed use the content.

But encryption is more of a means, rather than an end. A central issue is how to devise methods for ensuring, to the maximal extent possible, that only authorized devices and parties can decrypt the protected content and/or otherwise use information

only to the extent permitted by the rightsholder(s) and/or other relevant parties in the protected content.

The Present Inventions

The present inventions provide powerful right management capabilities. In accordance with one aspect provided by the present invention, encrypted digital properties can be put on a DVD in a tamper-resistant software "container" such as, for example, a "DigiBox" secure container, together with rules about "no copy" and/or "copy" and/or "numbers of permitted copies" that may apply and be enforced by consumer appliances. These same rules, and/or more flexible and/or different rules, can be enforced by computer devices or other systems that may provide more and/or different capabilities (e.g., editing, excerpting, one or more payment methods, increased storage capability for more detailed audit information, etc.). In addition, the "software container" such as for example, a "DigiBox" secure container, can store certain content in the "clear" (that is, in unencrypted form). For example, movie or music titles, copyright statements, audio samples, trailers, and/or advertising can be stored in the clear and/or could be displayed by any appropriate application or

device. Such information could be protected for authenticity (integrity) when available for viewing, copying, and/or other activities. At the same time, valuable digital properties of all kinds-film, video, image, text, software, and multimedia— may be stored at least partially encrypted to be used only by authorized devices and/or applications and only under permitted, for example rightsholder-approved, circumstances.

Another aspect provided in accordance with the present invention (in combination with certain capabilities disclosed in Ginter et al.) is that multiple sets of rules could be stored in the same "container" on a DVD disk. The software then applies rules depending on whether the movie, for example, was to be played by a consumer appliance or computer, whether the particular apparatus has a backchannel (e.g., an on-line connection), the national and/or other legal or geographic region in which the player is located and/or the movie is being displayed, and/or whether the apparatus has components capable of identifying and applying such rules. For example, some usage rules may apply when information is played by a consumer device, while other rules may apply when played by a computer. The choice of rules

may be left up to the rightsholder(s) and/or other participants-- or some rules may be predetermined (e.g., based on the particular environment or application). For example, film rightsholders may wish to limit copying and ensure that excerpts are not made regardless of the context in which the property is played. This limitation might be applied only in certain legal or geographic areas. Alternatively, rightsholders of sound recordings may wish to enable excerpts of predetermined duration (e.g., no more than 20 seconds) and that these excerpts are not used to construct a new commercial work. In some cases, governments may require that only "PG" versions of movies and/or the equivalent rating for TV programs may be played on equipment deployed in their jurisdiction, and/or that the applicable taxes, fees and the like are automatically calculated and/or collected if payments related to content recorded on DVD is requested and/or performed (e.g., pay-per-use of a movie, game, database, software product, etc.; and/or orders from a catalog stored at least in part on DVD media, etc.).

In a microprocessor controlled (or augmented) digital consumer appliance, such rules contemplated by the present

inventions can be enforced, for example, without requiring more than a relatively few additions to a central, controlling microprocessor (or other CPU, a IEEE 1394 port controller, or other content handling control circuitry), and/or making available some ROM or flash memory to hold the necessary software. In addition, each ROM (or flash or other memory, which such memory may be securely connected to, or incorporated into, such control circuitry in a single, manufactured component) can, in one example, contain one or more digital documents or "certificate(s)" that uniquely identifies a particular appliance, individual identity, jurisdiction, appliance class(es), and/or other chosen parameters. An appliance can, for example, be programmed to send a copy of a digital property to another digital device only in encrypted form and only inside a new, tamper-resistant "software container." The container may also, for example, carry with it a code indicating that it is a copy rather than an original that is being sent. The device may also put a unique identifier of a receiving device and/or class of devices in the same secure container. Consequently, for example, in one particular arrangement, the copy may be playable only on the intended receiving device,

class(es) of devices, and/or devices in a particular region in one non-limiting example and rights related to use of such copy may differ according to these and/or other variables.

The receiving device, upon detecting that the digital property is indeed a copy, can, for example, be programmed not to make any additional copies that can be played on a consumer device and/or other class(es) of devices. If a device detects that a digital property is about to be played on a device and/or other class(es) of devices other than the one it was intended for, it can be programmed to refuse to play that copy (if desired).

The same restrictions applied in a consumer appliance can, for example, be enforced on a computer equipped to provide rights management protection in accordance with the present inventions. In this example, rules may specify not to play a certain film and/or other content on any device other than a consumer appliance and/or classes of appliances, for example. Alternatively, these same powerful capabilities could be used to specify different usage rules and payment schemes that would apply when played on a computer (and/or in other appliances and/or classes of appliances), as the rightsholder(s) may desire, for example,

different pricing based upon different geographic or legal locales where content is played.

In addition, if "backchannels" are present—for example, set-top boxes with bi-directional communications or computers attached to networks—the present inventions contemplate electronic, independent delivery of new rules if desired or required for a given property. These new rules may, for example, specify discounts, time-limited sales, advertising subsidies, and/or other information if desired. As noted earlier, determination of these independently delivered rules is entirely up to the rightsholder(s) and/or others in a given model.

The following are two specific examples of a few aspects of the present invention discussed above:

1. An Analog To Digital Copying Example

a) Bob has a VHS tape he bought (or rented) and wants to make a copy for his own use. The analog film has copy control codes embedded so that they do not interfere with the quality of the signal. Bob has a writable DVD appliance

that is equipped to provide rights management protection in accordance with the present invention. Bob's DVD recorder detects the control codes embedded in the analog signal (for example, such recorder may detect watermarks and/or fingerprints carrying rights related control and/or usage information), creates a new secure container to hold the content rules and describe the encoded film, and creates new control rules (and/or delivers to a secure VDE system for storage and reporting certain usage history related information such as user name, time, etc.) based on the analog control codes and/or other information it detected and that are then placed in the DigiBox and/or into a secure VDE installation data store such as a secure data base. Bob can play that copy back on his DVD appliance whenever he chooses.

b) Bob gives the DVD disk he recorded to Jennifer who wishes to play it on computer that has a DVD drive. Her computer is equipped to provide rights management protection in accordance with the present invention. Her computer opens the "DigiBox," detects that this copy is being used on a device different from the one that recorded it (an unauthorized device) and refuses to play the copy.

c) Bob gives the DVD disk to Jennifer as before, but now Jennifer contacts electronically a source of new rules and usage consequences, which might be the studio, a distributor, and/or a rights and permissions clearinghouse, (or she may have sufficient rights already on her player to play the copy). The source sends a DigiBox container to Jennifer with rules and consequences that permit playing the movie on her

computer while at the same time charging her for use, even though the movie was recorded on DVD by Bob rather than by the studio or other value chain participant.

2. A Digital To Analog Copying Example

a) Jennifer comes home from work, inserts a rented or owned DVD into a player connected to, or an integral part of her TV, and plays the disk. In a completely transparent way, the film is decrypted, the format is converted from digital to analog, and displayed on her analog TV.

b) Jennifer wishes to make a copy for her own use. She plays the film on an DVD device incorporating rights management protection in accordance with the present invention, that opens the DigiBox secure container, accesses the control information, and decrypts the film.

She records the analog version on her VCR which records a high-quality copy.

c) Jennifer gives the VCR copy to Doug who wishes to make a copy of the analog tape for his own use, but the analog control information forces the recording VCR to make a lower-quality copy, or may prevent copying. In another non-limiting example, more comprehensive rights management information may be encoded in the analog output using the methods and/or systems described in more detail in the above referenced Van Wie and Weber patent application.

In accordance with one aspect provided by this invention, the same portable storage medium, such as a DVD, can be used with a range of different, scaled protection environments providing different protection capabilities. Each of the different environments may be enabled to use the information carried by the portable storage medium based on rights management techniques and/or capabilities supported by the particular environment. For

example, a simple, inexpensive home consumer disk player may support copy protection and ignore more sophisticated and complex content rights the player is not equipped to enable. A more technically capable and/or secure platform (e.g., a personal computer incorporating a secure processing component possibly supported by a network connection, or a "smarter" appliance or device) may, for example, use the same portable storage medium and provide enhanced usage rights related to use of the content carried by the medium based on more complicated rights management techniques (e.g., requiring payment of additional compensation, providing secure extraction of selected content portions for excerpting or anthologizing, etc.). For example, a control set associated with the portable storage medium may accommodate a wide variety of different usage capabilities—with the more advanced or sophisticated uses requiring correspondingly more advanced protection and rights management enablement found on some platforms and not others. Lower-capability environments can, as another example, ignore (or not enable or attempt to use) rights in the control set that they don't understand, while higher-capability environments (having awareness of the

overall capabilities they provide), may, for example, enable the rights and corresponding protection techniques ignored by the lower-capability environments.

In accordance with another aspect provided by the invention, a media- and platform-independent security component can be scaled in terms of functionality and performance such that the elementary rights management requirements of consumer electronics devices are subsets of a richer collection of functionality that may be employed by more advanced platforms. The security component can be either a physical, hardware component, or a "software emulation" of the component. In accordance with this feature, an instance of medium (or more correctly, one version of the content irrespective of media) can be delivered to customers independently of their appliance or platform type with the assurance that the content will be protected. Platforms less advanced in terms of security and/or technical capabilities may provide only limited rights to use the content, whereas more advanced platforms may provide more expansive rights based on correspondingly appropriate security conditions and safeguards.

In accordance with a further aspect provided by the present invention, mass-produced, inexpensive home consumer DVD players (such as those constructed, for example, with minimum complexity and parts count) can be made to be compatible with the same DVDs or other portable storage media used by more powerful and/or secure platforms (such as, for example, personal computers) without degrading advanced rights management functions the storage media may provide in combination with the more powerful and/or secure platforms. The rights management and protection arrangement provided and supported in accordance with this aspect of the invention thus supports inexpensive basic copy protection and can further serve as a commercial convergence technology supporting a bridging that allows usage in accordance with rights of the same content by a limited resource consumer device while adequately protecting the content and further supporting more sophisticated security levels and capabilities by (a) devices having greater resources for secure rights management, and/or (b) devices having connectivity with other devices or systems that can supply further secure rights management resources. This aspect of the invention allows

multiple devices and/or other systems that participate and work together in a permanently or temporarily connected network to share the rights management for at least one or more electronic events (e.g., managed through the use of protected processing environments such as described in Ginter et al.) occuring at a single, or across multiple nodes and further allows the rights associated with parties and/or groups using and/or controlling such multiple devices and/or other systems to be employed according to underlying rights related rules and controls, this allowing, for example, rights available through a corporate executive's device to be combined with or substitute for, in some manner, the rights of one or more subordinate corporate employees when their computing or other devices of these parties are coupled in a temporary networking relationship and operating in the appropriate context. In general, this aspect of the invention allows distributed rights management for DVD or otherwise packaged and delivered content that is protected by a distributed, peer-to-peer rights management.Such distributed rights management can operate whether the DVD appliance or other electronic information usage device is participating

permanently or temporarily connected network and whether or not the relationships among the devices and/or other systems participating in the distributed rights management arrangement are relating temporarily or have a more permanent operating relationship. In this way, the same device may have different rights available depending on the context in which that device is operating (e.g., in a corporate environment such as in collaboration with other individuals and/or with groups, in a home environment internally and/or in collaboration with external one or more specified individuals and/or other parties, in a retail environment, in a classroom setting as a student where a student's notebook might cooperate in rights management with a classroom server and/or instructor PC, in a library environment where multiple parties are collaboratively employing differing rights to use research materials, on a factory floor where a hand held device works in collaboration with control equipment to securely and appropriately perform proprietary functions, and so on).

For example, coupling a limited resource device arrangement, such as a DVD appliance, with an inexpensive network computer (NC), or a personal computer (PC), may allow

an augmenting (or replacing) of rights management capabilities and/or specific rights of parties and/or devices by permitting rights management to be a result of a combination of some or all of the rights and/or rights management capabilities of the DVD appliance and those of an Network or Personal Computer (NC or PC). Such rights may be further augmented, or otherwise modified or replaced by the availability of rights management capabilities provided by a trusted (secure) remote network rights authority.

These aspects of the present invention can allow the same device, in this example a DVD appliance, to support different arrays, e.g., degrees, of rights management capabilities, in disconnected and connected arrangements and may further allow available rights to result from the availability of rights and/or rights management capabilities resulting from the combination of rights management devices and/or other systems. This may include one or more combinations of some or all of the rights available through the use of a "less" secure and/or resource poor device or system which are augmented, replaced, or otherwise modified through connection with a device or system that is

"more" or "differently" secure and/or resource rich and/or possesses differing or different rights, wherein such connection employs rights and/or management capabilities of either and/or both devices as defined by rights related rules and controls that describe a shared rights management arrangement.

In the latter case, connectivity to a logically and/or physically remote rights management capability can expand (by, for example, increasing the available secure rights management resources) and/or change the character of the rights available to the user of the DVD appliance or a DVD appliance when such device is coupled with an NC, personal computer, local server, and/or remote rights authority. In this rights augmentation scenario, additional content portions may be available, pricing may change, redistribution rights may change (e.g., be expanded), content extraction rights may be increased, etc.

Such "networking rights management" can allow for a combination of rights management resources of plural devices and/or other systems in diverse logical and/or physical relationships, resulting in either greater or differing rights through the enhanced resources provided by connectivity with one or more

"remote" rights authorities. Further, while providing for increased and/or differing rights management capability and/or rights, such a connectivity based rights management arrangement can support multi-locational content availability, by providing for seamless integration of remotely available content, for example, content stored in remote, Internet world wide web-based, database supported content repositories, with locally available content on one or more DVD discs.

In this instance, a user may experience not only increased or differing rights but may use both local DVD content and supplementing content (i.e., content that is more current from a time standpoint, more costly, more diverse, or complementary in some other fashion, etc.). In such an instance, a DVD appliance and/or a user of a DVD appliance (or other device or system connected to such appliance) may have the same rights, differing, and/or different rights applied to locally and remotely available content, and portions of local and remotely available content may themselves be subject to differing or different rights when used by a user and/or appliance. This arrangement can support an overall, profound increase in user content opportunities that are seamlessly

integrated and efficiently available to users in a single content searching and/or usage activity by exploiting the rights management and content resources of plural, connected arrangements.

Such a rights augmenting remote authority may be directly coupled to a DVD appliance and/or other device by modem, or directly or indirectly coupled through the use of an I/O interface, such as a serial 1394 compatible controller (e.g., by communicating between a 1394 enabled DVD appliance and a local personal computer that functions as a smart synchronous or asynchronous information communications interface to such one or more remote authorities, including a local PC or NC or server that serves as a local rights management authority augmenting and/or supplying the rights management in a DVD appliance).

In accordance with yet another aspect provided by this invention, rights provided to, purchased, or otherwise acquired by a participant and/or participant DVD appliance or other system can be exchanged among such peer-to-peer relating devices and/or other systems through the use of one or more permenantly or temporarily networked arrangments. In such a case, rights may be

bartered, sold, for currency, otherwise exchanged for value, and/or loaned so long as such devices and/or other systems participate in a rights management system, for example, such as the Virtual Distribution Environment described in Ginter, et al., and employ rights transfer and other rights management capabilities described therein. For example, this aspect of the present invention allows parties to exchange games or movies in which they have purchased rights. Continuing the example, an individual might buy some of a neighbor's usage rights to watch a movie, or transfer to another party credit received from a game publisher for the successful superdistribution of the game to several acquaintances, where such credit is transferred (exchanged) to a friend to buy some of the friend's rights to play a different game a certain number of times, etc.In accordance with yet another aspect provided by this invention, content carried by a portable storage medium such as a DVD is associated with one or more encryption keys and a secure content identifier. The content itself (or information required to use the content) is at least partially cryptographically encrypted—with associated decryption keys being required to decrypt the content before the content can be

used. The decryption keys may themselves be encrypted in the form of an encrypted key block. Different key management and access techniques may be used, depending on the platform.

In accordance with still yet another aspect provided by this invention, electronic appliances that "create" digital content (or even analog content) —e.g., a digital camera/video recorder or audio recorder—can be readily equipped with appropriate hardware and/or software so as to produce content that is provided within a secure container at the outset. For example, content recorded by a digital camera could be immediately packaged in a secure container by the camera as it is recording. The camera could then output content already packaged in a secure container(s). This could preclude the need to encapsulate the content at a later point in time or at a later production stage, thus, saving at least one production-process step in the overall implementation of electronic rights management in accordance with the present invention. Moreover, it is contemplated that the very process of "reading" content for use in the rights management environment might occur at many steps along a conventional production and distribution process (such as during editing and/or

the so called "pressing" of a master DVD or audio disk, for example). Accordingly, another significant advantage of the present invention is that rights management of content essentially can be extended throughout and across each appropriate content creation, editing, distribution, and usage stages to provide a seamless content protection architecture that protects rights throughout an entire content life cycle.

In one example embodiment, the storage medium itself carries key block decryption key(s) in a hidden portion of the storage medium not normally accessible through typical access and/or copying techniques. This hidden key may be used by a drive to decrypt the encrypted key block—such decrypted key block then being used to selectively decrypt content and related information carried by the medium. The drive may be designed in a secure and tamper-resistant manner so that the hidden keys are never exposed outside of the drive to provide an additional security layer.

In accordance with another example embodiment, a video disk drive may store and maintain keys used to decrypt an encrypted key block. The key block decryption keys may be

stored in a drive key store, and may be updatable if the video disk drive may at least occasionally use a communications path provided, for example, by a set top box, network port or other communications route.

In accordance with a further example embodiment, a virtual distribution environment secure node including a protected processing environment such as a hardware-based secure processing unit may control the use of content carried by a portable storage medium such as a digital video disk in accordance with control rules and methods specified by one or more secure containers delivered to the secure node on the medium itself and/or over an independent communications path such as a network.

Certain conventional copy protection for DVD currently envisions CGMA copy protection control codes combined with certain encryption techniques first proposed apparently by Matsushita Corporation. Notwithstanding the limited benefits of this approach to digital property protection, the present invention is capable of providing a supplementary, compatible, and far more comprehensive rights management system while also providing

additional and/or different options and solutions. The following are some additional examples of advantageous features provided in accordance with the inventions:

- Strong security to fully answer content supplier needs.

- Value chain management automation and efficiencies including distributed rights protection, "piece of the tick" payment disaggregation to value chain participants, cost-effective micro-transaction management, and superdistribution, including offline micropayment and microtransaction support for at least occasionally connected devices.

- Simplified, more efficient channel management including support for the use of the same content deliverable on limited resource, greater resource, standalone, and/or connected devices.

- Can be used with any medium and application type and/or all forms of content and content models -- not just compressed video and sound as in some prior techniques and supports the use of copies of the same or materially the

same content containers across a wide variety of media delivery systems (e.g., broadcast, Internet repository, optical disc, etc) for operation on a wide variety of different electronic appliances (e.g., digital cameras, digital editing equipment, sound recorders, sound editing equipment, movie theater projectors, DVD appliances, broadcast tape players, personal computers, smart televisions, etc).

- Asset management and revenue and/or other consideration maximizing through important new content revenue and/or other consideration opportunities and the enhancement of value chain operating efficiencies.

- Is capable of providing 100% compatibility with the other protection techniques such as, for example, CGMA protection codes and/or Matsushita data scrambling approaches to DVD copy protection.

- Can be employed with a variety of existing data scrambling or protection systems to provide very high degrees of compatibility and/or level of functionality.

- Allows DVD technology to become a reusable, programmable, resource for an unlimited variety of entertainment, information commerce, and cyberspace business models.

- Enables DVD drive and/or semiconductor component manufacturers and/or distributors and/or other value adding participants to become providers of, and rights holders in, the physical infrastructure of the emerging, connected world of the Internet and Intranets where they may charge for the use of a portion (e.g., a portion they provided) of the distributed, physical infrastructure as that portion participates in commercial networks. Such manufacturers and/or distributors and/or other value adding participants can enjoy the revenue benefits resulting from participation in a "piece of the tick" by receiving a small portion of the revenue received as a result of a participating transaction.

- Provides automated internationalization, regionalization, and rights management in that:

  -- DVD content can be supplied with arrays of different rule sets for

automatic use depending on rights and identity of the user; and

-- Societal rights, including taxes, can be handled transparently.

In addition, the DVD rights management method and apparatus of the present invention provides added benefits to media recorders/publishers in that it:

- Works with a current "keep honest people honest" philosophy.
- Can provide 100% compatibility with other protection schemes such as for example, Matsushita data scrambling and/or CGMA encoded discs.
- Can work with and/or supplement other protection schemes to provide desired degree and/or functionality, or can be used in addition to or instead of other approaches to provide additional and/or different functionality and features.

- Provides powerful, extensible rights management that reaches beyond limited copy protection models to rights management for the digitally convergent world.

- Empowers recording/publishing studios to create sophisticated asset management tools.

- Creates important business opportunities through controlled use of studio properties in additional multimedia contexts.

- Uniquely ties internationalization, regionalization, superdistribution, repurposing, to content creation processes and/or usage control.

Other aspects of the present invention provide benefits to other types of rightsholders, such as for example:

- Persistent, transparent protection of digital content—globally, through value chain and process layers.

- Significant reduction in revenue loss from copying and pass-along.

- Converts "pass-along," copying, and many forms of copyright infringement from a strategic business threat to a fundamental business opportunity.

- A single standard for all digital content regardless of media and/or usage locality and other rights variables.

- Major economies of scale and/or scope across industries, distribution channels, media, and content type.

- Can support local usage governance and auditing within DVD players allowing for highly efficient micro-transaction support, including multiparty microtransactions and transparent multiparty microtransactions.

- Empowers rightsholders to employ the broadest range of pricing, business models, and market strategies—as they see fit.

Further aspects of the present invention which may prove beneficial to DVD and other digital medium appliance manufacturers are:

- Capable of providing bit for bit compatibility with existing discs.

- Content type independent.

- Media independent and programmable/reusable.

- Highly portable transition to next generation of appliances having higher density devices and/or a writable DVD and/or other optical media format(s).

- Participation in revenue flow generated using the appliance.

- Single extensible standard for all digital content appliances.

- Ready for the future "convergent" world in which many appliances are connected in the home using, as one example, IEEE 1394 interfaces or other means (e.g., some appliances will be very much like computers and some computers will be very much like appliances).

Aspects of the present inventions provide many benefits to computer and OS manufacturers such as for example:

- Implementation in computers as an extension to the operating system, via for example, at least one transparent plug-in, and does not require modifications to computer hardware and/or operating systems.

- Easy, seamless integration into operating systems and into applications.

- Extremely strong security, especially when augmented with "secure silicon" (i.e., hardware/firmware protection apparatus fabricated on chip).

- Transforms user devices into true electronic commerce appliances.

- Provides a platform for trusted, secure rights management and event processing.

- Programmable for customization to specialized requirements.

Additional features and advantages provided in accordance with the inventions include, for example:

- Information on the medium (for example, both properties and metadata) may be encrypted or not.

- Different information (for example, properties, metadata) may be encrypted using different keys. This provides greater protection against compromise, as well as supporting selective usage rights in the context of a sophisticated rights management system.

- There may be encrypted keys stored on the medium, although this is not required. These keys may be used to decrypt the protected properties and metadata. Encrypted keys are likely to be used because that allows more keying material for the information itself, while still keeping access under control of a single key.

- Multiple sets of encrypted keys may be stored on the medium, either to have different sets of keys associated with different information, or to allow multiple control regimes to use the

same information, where each control regime may use one or more different keys to decrypt the set of encrypted keys that it uses.

- To support the ability of the player to access rights managed containers and/or content, a decryption key for the encrypted keys may be hidden on the medium in one or more locations that are not normally accessible. The "not normally accessible" location(s) may be physically enabled for drives installed in players, and disabled for drives installed in computers. The enablement may be different firmware, a jumper on the drive, etc.

- The ability of the player to access rights managed containers and/or content may also be supported by one or more stored keys inside the player that decrypts certain encrypted keys on the medium.

- Keys in a player may allow some players to play different properties than others. Keys could be added to, and/or deleted from the player by a network connection (e.g., to a PC, a cable system, and/or a modem connection to a source of new and/or additional keys and/or

key revocation information) or automatically loaded by "playing" a key distribution DVD.

- Controlling computer use may be supported by some or all of the same techniques that control player use of content and/or rights managemnt information.

- Controlling computer use of content and/or rights management information may be supported by having a computer receive, through means of a trusted rights management system, one or more appropriate keys.

- A computer may receive additional keys that permit decryption of certain encrypted keys on the medium.

- A computer may receive additional keys that permit decryption of one or more portions of encrypted data directly. This may permit selective use of information on the medium without disclosing keys (e.g., a player key that decrypts any encrypted keys).

In accordance with further aspects provided by the present invention, a secure "software container" is provided that allows:

- Cryptographically protected encapsulation of content, rights rules, and usage controls.

- Persistent protection for transport, storage, and value chain management.

- Sophisticated rules interface architecture.

Elements can be delivered independently, such as new controls, for example, regarding discount pricing (e.g. sale pricing, specific customer or group discounts, pricing based on usage patterns, etc.) and/or other business model changes, can be delivered after the property has been distributed (this is especially beneficial for large properties or physical distribution media (e.g., DVD, CD-ROM) since redistribution costs may be avoided and consumers may continue to use their libraries of discs). In addition, encrypted data can be located "outside" the container. This can allow, for example, use of data stored independently from the controls and supports "streaming" content as well as "legacy" systems (e.g., CGMS).

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages provided in accordance with these inventions may be better and more completely understood by referring to the following detailed description of presently preferred examples in conjunction with the drawings, of which:

Figure 1A shows example home consumer electronics equipment for using portable storage media such as digital video disks;

Figure 1B shows example secure node equipment for using the same portable storage media but providing more advanced rights management capabilities;

Figure 1C shows an example process for manufacturing protected optical disks;

Figure 2A shows an example architecture of the Figure 1A consumer electronics equipment;

Figure 2B shows an example architecture for the Figure 1B secure node equipment;

Figure 3 shows example data structures used by the Figure 1A equipment;

Figure 3A and 3B show example control set definitions;

Figures 4A and 4B show example usage techniques provided by the Figure 1A appliance;

Figure 5 shows example data structures used by the Figure 1B secure node for accessing information on the storage medium;

Figure 6 shows an example usage technique performed by the Figure 1B secure node;

Figure 7 is a block diagram illustrating an example of a special secure software container contained on a DVD;

Figure 8 is a block diagram illustrating an example of a secure container along with the video property content stored on a DVD medium;

Figure 9 is a block diagram illustrating another example of a standard container stored on a DVD medium including an additional container having a more complex rule arrangement for use, for example, with a secure node;

Figure 10 shows an example use of a DVD having a container (i.e., stored on the medium) with a DVD player provided with a secure rights management node, and also shows use of the same DVD with a DVD player that does not have a secure rights management node;

Figure 11 is a block diagram illustrating use of a DVD that does not have a container on a DVD player that is provided with rights management secure node in accordance with the present invention as compared with use of the same DVD with a DVD player that does not have a secure node;

Figures 12-14 show example network configurations; and

Figures 15A-15C show an example virtual rights process.

DETAILED DESCRIPTION OF PRESENTLY PREFERRED EXAMPLE EMBODIMENTS

Overall Example Digital Video Disk Usage System

Figure 1A shows example inexpensive mass-produced home consumer electronics equipment 50 for using information stored on a storage medium 100 such as a portable digitally-encoded optical disk (e.g., a digital video disk or "DVD").

Consumer equipment 50 includes a dedicated disk player 52, that in some embodiments, may also have the capability to write optical media (writeable DVD disks, or "DVD-RAM") for example) as well, connected to a home color television set 54. A remote control unit 56 may be used to control the disk player 52 and/or television set 54.

In one example, disk 100 may store a feature length motion picture or other video content. Someone wishing to watch the content stored on disk 100 may purchase or rent the disk, insert the disk into player 52 and use remote control 56 (and/or controls 58 that may be provided on player 52) to control the player to play back the content via home television set 54.

In some embodiments, remote control 56 (and/or controls 58 that may be provided on device 52) may be used to control the recording of a movie, for example. Player 52 reads the digitized video and audio information carried by disk 100, converts it into signals compatible with home color television set 54, and provides those signals to the home color television set.

In some embodiments, television set 54 (and/or a set top box) provide the video signals to be recorded by device 52 on writable optical media, DVD-RAM in one non-limiting example. Television set 54 produces images on screen 54a and produces sounds through loudspeakers 54b based on the signals player 52 provides to the television set.

The same disk 100 may be used by a more advanced platform 60 shown in Figure 1B. Platform 60 may include, for example, a personal computer 62 connected to a display monitor 64, a keyboard 66, a mouse pointing device 68, and a loudspeaker 70. In this example, platform 60 may be able to play back the content stored on disk 100 in the same way as dedicated disk player 52, but may also be capable of more sophisticated and/or advanced uses of the content as enabled by the presence of secure node 72 within the platform. (In some embodiments, platform 60 may also be able to record content on writable optical media, DVD-RAM, in one non-limiting example.) For example, it may be possible, using platform 60 and its secure node 72, to interactively present the motion picture or other content such that the user may input choices via keyboard 66 and/or mouse pointing device 68

that, in real time, change the presentation provided via display 64 and loudspeaker 60.

As one example, the platform 60 user selects from options displayed on display 64 that cause the content presentation sequence to change (e.g., to provide one of a number of different endings, to allow the user to interactively control the flow of the images presented, etc.). Computer 62 may also be capable of using and manipulating digital data including for example computer programs and/or other information stored on disk 100 that player 52 cannot handle.

Secure node 72 provides a secure rights management facility that may, for example, permit more invasive or extensive use of the content stored on disk. For example, dedicated player 52 may prevent any copying of content stored by disk 100, or it may allow the content to be copied only once and never again. Platform 60 including secure node 72, on the other hand, may allow multiple copies of some or all of the same content—but only if certain conditions are met (e.g., the user of equipment 60 falls within a certain class of people, compensation at an agreed on rate is securely provided for each copy made, only certain excerpts of

the content are copied, a secure audit trail is maintained and reported for each copy so made, etc.). (In some embodiments, dedicated player 52 may send protected content only to devices authenticated as able to enforce securely rights management rules and usage consequences. In some embodiments, devices may authenticate using digital certificates, one non-limiting example being certificates conforming to the X.509 standard.) Hence, platform 60 including secure node 72 can, in this example, use the content provided by disk 100 in a variety of flexible, secure ways that are not possible using dedicated player 52—or any other appliance that does not include a secure node.

Example Secure Disk Creation and Distribution Process

Figure 1C shows an example secure process for creating a master multimedia DVD disk 100 for use with players 50, 60. In this example, a digital camera 350 converts light images (i.e., pictures) into digital information 351 representing one or a sequence of images. Digital camera 350 in this example includes a secure node 72A that protects the digital information 351 before it leaves camera 350. Such protection can be accomplished, for

example, by packaging the digital information within one or more containers and/or associating controls with the digital information.

In this example, digital camera 350 provides the protected digital image information 351 to a storage device such as, for example, a digital tape recorder 352. Tape recorder 352 stores the digital image information 351 (along with any associated controls) onto a storage medium such as magnetic tape cartridge 354 for example. Tape recorder 352 may also include a secure node 72B. Secure node 72B in this example can understand and enforce the controls that the digital camera secure node 72A applies to and/or associated with the digital information 351, and/or it may apply its own controls to the stored information.

The same or different tape recorder 352 may play back protected digital information 351 to a digital mixing board 356. Digital mixing board 356 may mix, edit, enhance or otherwise process the digital information 351 to generate processed digital information 358 representing one or a sequence of images. Digital mixing board 356 may receive additional inputs from other devices such as for example other tape recorders, other digital cameras, character generators, graphics generators, animators, or

any other image-based devices. Any or all of such devices may also include secure nodes 72 to protect the information they generate. In some embodiments, some of the digital information can be derived from equipment including a secure node, and other digital information can be derived from equipment that has no secure node. In still other embodiments, some of the digital information provided to digital mixer 356 is protected and some is not protected.

Digital mixing board 356 may also include a secure node 72C in this example. The digital mixing board secure node 72C may enforce controls applied by digital camera secure node 72A and/or tape recorder secure node 72B, and/or it may add its own protections to the digital information 358 it generates.

In this example, an audio microphone 361 receives sound and converts the sound into analog audio signals. The audio signals in this example are inputted to a digital audio tape recorder 362. In the example shown, tape recorder 362 and audio mixer 364 are digital devices. However, in other embodiments, one, the other or both of these devices may operate in the analog domain. In the example shown, digital audio tape recorder 362 converts the

analog audio signals into digital information representing the sounds, and stores the digital information (and any associated controls) onto a tape 362.

In this example, audio tape recorder 362 includes a secure node 72E that may associate controls with the information stored on tape 363. Such controls may be stored with the information on the tape 363. In another embodiment, microphone 361 may include its own internal secure node 72 that associates control information with the audio information (e.g., by steganographically encoding the audio information with control information). The tape recorder 362 may enforce such controls applied by microphone 361.

Alternatively, microphone 361 may operate in the digital domain and provide digital representations of audio, perhaps including control information supplied by secure node 72 optionally incorporated in microphone 361, directly to connected devices such as audio tape recorder 362. Digital representations may optionally be substituted for analog representations of any signals between the devices in the example Figure 1C.

The same or different tape recorder 362 may play back the information recorded on tape 363, and provide the information 366 to an audio mixer 364. Audio mixer 364 may edit, mix, or otherwise process the information 366 to produce information 368 representing one or a sequence of sounds. Audio mixer 364 may also receive inputs from other devices such as for example other tape recorders, other microphones, sound generators, musical synthesizers, or any other audio-based devices. Any or all of such devices may also include secure nodes 72 to protect the information they generate. In some embodiments, some of the digital information is derived from equipment including a secure node, and other digital information is derived from equipment that has no secure node. In still other embodiments, some of the digital information provided to audio mixer 364 is protected and some is not protected.

Audio mixer 364 in this example includes a secure node 72F that enforces the controls, if any, applied by audio tape recorder secure node 72E; and/or applies its own controls.

Digital image mixer 356 may provide digital information 358 to "DVD-RAM" equipment 360 that is capable of writing to

master disks 100 and/or to disks from which master dicks may be created Similarly, audio mixer 364 may provide digital information 368 to equipment 360. Equipment 360 records the image information 358 and audio information 368 onto master disk 100. In this example, equipment 360 may include a secure node 72D that enforces controls applied by digital camera secure node 72A, tape recorder secure node 72B, digital mixer secure node 72C audio tape recorder secure node 72E and/or audio mixer secure node 72F; and/or it may add its own protections to the digital information 358 it writes onto master disks 100. A disk manufacturer can then mass-produce disks 100(1)-100(N) based on the master disk 100 using conventional disk mass-production equipment for distribution through any channels (e.g., video and music stores, websites, movie theaters, etc.). Consumer appliances 50 shown in Figures 1A and 1B may play back the disks 100 – enforcing the controls applied to the information stored on the disks 100. Secure nodes 72 thus maintain end-to-end, persistent secure control over the images generated by digital camera 350 and the sounds generated by microphone 361 during the entire process of making, distributing and using disks 100.

In the Figure 1C example shown, the various devices may communicate with one another over so-called "IEEE 1394" high-speed digital serial busses. In this context, "IEEE 1394" refers to hardware and software standards set forth in the following standards specification incorporated by reference herein: 1394-1995 IEEE Standard for a High Performance Serial Bus, No. 1-55937-583-3 (Institute of Electrical and Electronics Engineers 1995). This specification describes a high-speed memory mapped digital serial bus that is self-configuring, hot pluggable, low cost and scalable. The bus supports isochronous and asynchronous transport at 100, 200 or 400 Mbps, and flexibly supports a number of different topologies. The specification describes a physical level including two power conductors and two twisted pairs for signalling. The specification further describes physical, link and transaction layer protocols including serial bus management. Alternatively, any other suitable electronic communication means may be substituted for the "IEEE 1394" medium shown in Figure 1C, including other wired media (e.g., Ethernet, universal serial bus), and/or wireless media based on radio-frequency (RF)

transmission, infra-red signals, and/or any other means and/or types of electronic communication.

Example Dedicated Player Architecture

Figure 2A shows an example architecture for dedicated player 52. In this example, player 52 includes a video disk drive 80, a controller 82 (e.g., including a microprocessor 84, a memory device such as a read only memory 86, and a user interface 88), and a video/audio processing block 90. Video disk drive 80 optically and physically cooperates with disk 100, and reads digital information from the disk. Controller 82 controls disk drive 80 based on program instructions executed by microprocessor 84 and stored in memory 86 (and further based on user inputs provided by user interface 88 which may be coupled to controls 58 and/or remote control unit 56). Video/audio processing block 90 converts digital video and audio information read by disk drive 80 into signals compatible with home color television set 54 using standard techniques such as video and audio decompression and the like. Video/audio processing block 90 may also insert a visual marking indicating the ownership and/or protection of the video program. Block 90 may also

introduce a digital marking indicating to a standard recording device that the content should not be recorded.

Example Secure Node Architecture

Figure 2B shows an example architecture for platform 60 shown in Figure 1B—which in this example is built around a personal computer 62 but could comprise any number of different types of appliances. In this example, personal computer 62 may be connected to an electronic network 150 such as the Internet via a communications block 152. Computer equipment 62 may include a video disk drive 80' (which may be similar or identical to the disk drive 80 included within example player 52). Computer equipment 62 may further include a microprocessor 154, a memory 156 (including for example random access memory and read only memory), a magnetic disk drive 158, and a video/audio processing block 160. Additionally, computer equipment 62 may include a tamper-resistant secure processing unit 164 or other protected processing environment. Secure node 72 shown in Figure 1B may thus be provided by a secure processing unit 164, software executing on microprocessor 154, or a combination of

the two. Different embodiments may provide secure node 72 using software-only, hardware-only, or hybrid arrangements.

Secure node 72 in this example may provide and support a a general purpose Rights Operating System employing reusable kernel and rights language components. Such a commerce-enabling Rights Operating System provides capabilities and integration for advanced commerce operating systems of the future. In the evolving electronic domain, general purpose, reusable electronic commerce capabilities that all participants can rely on will become as important as any other capability of operating systems. Moreover, a rights operating system that provides, among other things, rights and auditing operating system functions can securely handle a broad range of tasks that relate to a virtual distribution environment. A secure processing unit can, for example, provide or support many of the security functions of the rights and auditing operating system functions. The other operating system functions can, for example, handle general appliance functions. The overall operating system may, for example, be designed from the beginning to include the rights and auditing operating system functions plus the other operating

system functions, or the rights and auditing operating system functions may, in another example, be an add-on to a preexisting operating system providing the other operating system functions. Any or all of these features may be used in combination with the invention disclosed herein.

Example Disk Data Structures and Associated Protections

Figure 3 shows some example data structures stored on disk 100. In this example, disk 100 may store one or more properties or other content 200 in protected or unprotected form. Generally, in this example, a property 200 is protected if it is at least in part encrypted and/or associated information needed to use the property is at least in part encrypted and/or otherwise unusable without certain conditions having being met. For example, property 200(1) may be completely or partially encrypted using conventional secure cryptographic techniques. Another property 200(2) may be completely unprotected so that it can be used freely without any restriction. Thus, in accordance with this example, disk 100 could store both a movie as a protected property 200(1) and an unprotected interview with the actors and producers or a

"trailer" as unprotected property 200(2). As shown in this example, disk 100 may store any number of different properties 200 in protected or unprotected form as limited only by the storage capacity of the disk.

In one example, the protection mechanisms provided by disk 100 may use any or all of the protection (and/or other) structures and/or techniques described in the above-referenced Shear patents. The Shear patents describe, by way of non-exhaustive example, means for solving the problem of how to protect digital content from unauthorized use. For example, the Shear patent specifications describe, among other things, means for electronically "overseeing" -- through distributed control nodes present in client computers -- the use of digital content. This includes means and methods for fulfilling the consequences of any such use.

Non-limiting examples of certain elements described in the Shear patent specifications include:

(a) decryption of encrypted information,

(b) metering,

(c) usage control in response to a combination of derived metering information and rules set by content providers,

(d) securely reporting content usage information,

(e) use of database technology for protected information storage and delivery,

(f) local secure maintenance of budgets, including, for example, credit budgets,

(g) local, secure storage of encryption key and content usage information,

(h) local secure execution of control processes, and

(i) in many non-limiting instances, the use of optical media.

Any or all of these features may be used in combination in or with the inventions disclosed herein.

Certain of the issued Shear patents' specifications also involve database content being local and remote to users.

Database information that is stored locally at the end-user's system and complemented by remote, "on-line" database information, can, for example, be used to augment the local information, which in one example, may be stored on optical media (for example, DVD and/or CD-ROM). Special purpose semiconductor hardware can, for example, be used to provide a secure execution environment to ensure a safe and reliable setting for digital commerce activities.

The Shear patents also describe, among other things, database usage control enabled through the use of security, metering, and usage administration capabilities. The specifications describe, *inter alia*, a metering and control system in which a database, at least partially encrypted, is delivered to a user (e.g., on optical media). Non-limiting examples of such optical media may, for example, include DVD and CD-ROM. Subsequent usage can, for example, be metered and controlled in any of a variety of ways, and resulting usage information can be transmitted to a responsible party (as one example).

The Shear patent specifications also describe the generation of a bill in response to the transmitted information. Other

embodiments of the Shear patents provide, for example, unique information security inventions which involve, for example, digital content usage being limited based on patterns of usage such as the quantity of particular kinds of usage. These capabilities include monitoring the "contiguousness," and/or "logical relatedness" of used information to ensure that the electronic "conduct" of an individual does not exceed his or her licensed rights. Still other aspects of the Shear patents describe, among other things, capabilities for enabling organizations to securely and locally manage electronic information usage rights. When a database or a portion of a database is delivered to a client site, some embodiments of the Shear patents provide, for example, optical storage means (non-exhaustive examples of which include DVD and CD-ROM) as the mechanism of delivery. Such storage means can store, for example, a collection of video, audio, images, software programs, games, etc., in one example, on optical media, such as DVD and/or CD-ROM, in addition to other content such as a collection of textual documents, bibliographic records, parts catalogs, and copyrighted or uncopyrighted materials of all kinds.

Any or all of these features may be used in the embodiments herein.

One specific non-limiting embodiment could, for example, involve a provider who prepares a collection of games. The provider prepares a database "index" that stores information pertaining to the games, such as for example, the name, a description, a creator identifier, the billing rates, and the maximum number of times or total elapsed time each game may be used prior to a registration or re-registration requirement. Some or all of this information could be stored in encrypted form, in one example, on optical media, non-limiting examples of which include DVD and CD-ROM. The provider may then encrypt some or all portions of the games such that a game could not be used unless one or more encrypted portions were decrypted. Typically, decryption would not occur unless provider specified conditions were satisfied, in one example, unless credit was available to compensate for use and audit information reflecting game usage was being stored. The provider could determine, for example: which user activities he or she would allow, whether to meter such activities for audit and/or control purposes, and what, if any, limits

would be set for allowed activities. This might include, for example, the number of times that a game is played, and the duration of each play. Billing rates might be discounted, for example, based on total time of game usage, total number of games currently registered for use, or whether the customer was also registered for other services available from the same provider, etc.

In the non-limiting example discussed above, a provider might, for example, assemble all of the prepared games along with other, related information, and publish the collection on optical media, non-limiting examples of which include CD-ROM and/or DVD. The provider might then distribute this DVD disk to prospective customers. The customers could then select the games they wish to play, and contact the provider. The provider, based on its business model, could then send enabling information to each authorized customer, such as for example, including, or enabling for use, decryption keys for the encrypted portion of the selected games (alternatively, authorization to use the games may have arrived with the DVD and/or CD-ROM disk, or might be automatically determined, based on provider set criteria, by the

user's secure client system, for example, based on a user's participation in a certified user class). Using the user's client decryption and metering mechanism the customer could then make use of the games. The mechanism might then record usage information, such as for example, the number of times the game was used, and, for example, the duration of each play. It could periodically transmit this information the game provider, thus substantially reducing the administration overhead requirements of the provider's central servers. The game provider could receive compensation for use of the games based upon the received audit information. This information could be used to either bill their customers or, alternatively, receive compensation from a provider of credit.

Although games provide one convenient, non-limiting example, many of these same ideas can be easily applied to all kinds of content, all kinds of properties, including, by way of non-limiting examples:

- video,
- digitized movies,

- audio,
- images,
- multimedia,
- software,
- games,
- any other kind of property
- any combination of properties.

Other non-limiting embodiments of the Shear patent specifications support, for example, securely controlling different kinds of user activities, such as displaying, printing, saving electronically, communicating, etc. Certain aspects further apply different control criteria to these different usage activities. For example, information that is being browsed may be distinguished from information that is read into a host computer for the purpose of copying, modifying, or telecommunicating, with different cost rates being applied to the different activities (so that, for example,

the cost of browsing can be much less than the cost of copying or printing).

The Shear patent specifications also, for example, describe management of information inside of organizations by both publishers and the customer. For example, an optional security system can be used to allow an organization to prevent usage of all or a portion of an information base unless the user enters his security code. Multiple levels of security codes can be supported to allow restriction of an individual's use according to his security authorization level. One embodiment can, for example, use hardware in combination with software to improve tamper resistance, and another embodiment could employ an entirely software based system. Although a dedicated hardware/software system may under certain circumstances provide assurance against tampering, techniques which may be implemented in software executing on a non-dedicated system may provide sufficient tamper resistance for some applications. Any or all of these features may be used in combination with the technology disclosed in this patent specification.

Figures 3 Disks May Also Store Metadata, Controls and Other Information

In this example, disk 100 may also store "metadata" in protected and/or unprotected form. Player 52 uses metadata 202 to assist in using one or more of the properties 200 stored by disk 100. For example, disk 100 may store one metadata block 202(1) in unprotected form and another metadata block 202(2) in protected form. Any number of metadata blocks 202 in protected and/or unprotected form may be stored by disk 100 as limited only by the disk's storage capacity. In this example, metadata 202 comprises information used to access properties 200. Such metadata 202 may comprise, for example, frame sequence or other "navigational" information that controls the playback sequence of one or more of the properties 200 stored on disk 100. As one example, an unprotected metadata block 202 may access only selected portions of a protected property 200 to generate an abbreviated "trailer" presentation, while protected metadata block 202 may contain the frame playback sequence for the entire video presentation of the property 200. As another example, different metadata blocks 202 may be provided for different "cuts" of the

same motion picture property 200 (e.g., an R-rated version, a PG-rated version, a director's cut version, etc.).

In this example, disk 100 may store additional information for security purposes. For example, disk 100 may store control rules in the form of a control set 204—which may be packaged in the form of one or more secure containers 206. Commerce model participants can securely contribute electronic rules and controls that represent their respective "electronic" interests. These rules and controls extend a "Virtual Presence™" through which the commerce participants may govern remote value chain activities according to their respective, mutually agreed to rights. This Virtual Presence may take the form of participant specified electronic conditions (e.g., rules and controls) that must be satisfied before an electronic event may occur. These rules and controls can be used to enforce the party's rights during "downstream" electronic commerce activities. Control information delivered by, and/or otherwise available for use with, VDE content containers may, for example, constitute one or more "proposed" electronic agreements which manage the use and/or consequences of the use of such content and which can enact the

terms and conditions of agreements involving multiple parties and their various rights and obligations.

The rules and controls from multiple parties can be used, in one example, to form aggregate control sets ("Cooperative Virtual Presence™") that ensure that electronic commerce activities will be consistent with the agreements amongst value chain participants. These control sets may, for example, define the conditions which govern interaction with protected digital content (disseminated digital content, appliance control information, etc.). These conditions can, for example, be used to control not only digital information use itself, but also the consequences of such use. Consequently, the individual interests of commerce participants are protected and cooperative, efficient, and flexible electronic commerce business models can be formed. These models can be used in combination with the present invention.

Disks May Store Encrypted Information

Disk 100 may also store an encrypted key block 208. In this example, disk 100 may further store one or more hidden keys 210. In this example, encrypted key block 208 provides one or more cryptographic keys for use in decrypting one or more properties

200 and/or one or more metadata blocks 202. Key block 208 may provide different cryptographic keys for decrypting different properties 200 and/or metadata blocks 202, or different portions of the same property and/or metadata block. Thus, key block 208 may comprise a large number of cryptographic keys, all of which are or may be required if all of the content stored by disk 100 is to be used. Although key block 208 is shown in Figure 3 as being separate from container 206, it may be included within or as part of the container if desired.

Cryptographic key block 208 is itself encrypted using one or more additional cryptographic keys. In order for player 52 to use any of the protected information stored on disk 100, it must first decrypt corresponding keys within the encrypted key block 208—and then use the decrypted keys from the key block to decrypt the corresponding content.

In this example, the keys required to decrypt encrypted key block 208 may come from several different (possibly alternative) sources. In the example shown in Figure 3, disk 100 stores one or more decryption keys for decrypting key block 208 on the medium itself in the form of a hidden key(s) 210. Hidden key(s) 210 may

be stored, for example, in a location on disk 100 not normally accessible. This "not normally accessible" location could, for example, be physically enabled for drives 80 installed in players 52 and disabled for drives 80' installed in personal computers 62. Enablement could be provided by different firmware, a jumper on drive 80, etc. Hidden key(s) 210 could be arranged on disk 100 so that any attempt to physically copy the disk would result in a failure to copy the hidden key(s). In one example a hidden key(s) could be hidden in the bit stream coding sequences for one or more blocks as described by J. Hogan (Josh Hogan, "DVD Copy Protection," presentation to DVD copy protect technical meeting #4, 5/30/96, Burbank, CA.)

Alternatively, and/or in addition, keys required to decrypt encrypted key block 208 could be provided by disk drive 80. In this example, disk drive 80 might include a small decryption component such as, for example, an integrated circuit decryption engine including a small secure internal key store memory 212 having keys stored therein. Disk drive 80 could use this key store 212 in order to decrypt encrypted key block 208 without exposing either keys 212 or decrypted key block 208—and then use the

decrypted key from key block 208 to decrypt protected content 200, 202.

Disks May Store and/or Use Secure Containers

In yet another example, the key(s) required to decrypt protected content 200, 202 is provided within secure container 206. Figure 3A shows a possible example of a secure container 206 including information content 304 (properties 200 and metadata 202 may be external to the container—or alternatively, most or all of the data structures stored by video disk 100 may be included as part of a logical and/or actual protected container). The control set 204 shown in Figure 3 may comprise one or more permissions record 306, one or more budgets 308 and/or one or more methods 310 as shown in Figure 3A. Figure 3B shows an example control set 204 providing one or more encryption keys 208, one or more content identifiers 220, and one or more controls 222. In this example, different controls 222 may apply to different equipment and/or classes of equipment such as player 52 and/or computer equipment 62 depending upon the capabilities of the particular platform and/or class of platform. Additionally, controls 220 may apply to different ones of properties 200 and/or

different ones of metadata blocks 202. For example, a control 222(1) may allow property 200(1) to be copied only once for archival purposes by either player 52 or computer equipment 62. A control 222(2) (which may be completely ignored by player 52 because it has insufficient technical and/or security capabilities but which may be useable by computer equipment 62 with its secure node 72) may allow the user to request and permit a public performance of the same property 200(1) (e.g., for showing in a bar or other public place) and cause the user's credit or other account to be automatically debited by a certain amount of compensation for each showing. A third control 222(3) may, for example, allow secure node 72 (but not player 52) to permit certain classes of users (e.g., certified television advertisers and journalists) to extract or excerpt certain parts of protected property 200(1) for promotional uses. A further control 222(4) may, as another example, allow both video player 52 and secure node 72 to view certain still frames within property 200(1)—but might allow only secure node 72 to make copies of the still frames based on a certain compensation level.

Example Disks and/or System May Make Use of Trusted Infrastructure

Controls 222 may contain pointers to sources of additional control sets for one or more properties, controls, metadata, and/or other content on the optical disk. In one example, these additional controls may be obtained from a trusted third party, such as a rights and permissions clearinghouse and/or from any other value chain participant authorized by at least one rightsholder to provide at least one additional control set. This kind of rights and permissions clearinghouse is one of several distributed electronic administrative and support services that may be referred to as the "Distributed Commerce Utility," which, among other things, is an integrated, modular array of administrative and support services for electronic commerce and electronic rights and transaction management. These administrative and support services can be used to supply a secure foundation for conducting financial management, rights management, certificate authority, rules clearing, usage clearing, secure directory services, and other transaction related capabilities functioning over a vast electronic network such as the Internet and/or over organization internal Intranets, or even in-home networks of electronic appliances. Non-

limiting examples of these electronic appliances include at least occasionally connected optical media appliances, examples of which include read-only and/or writable DVD players and DVD drives in computers and convergent devices, including, for example, digital televisions and settop boxes incorporating DVD drives.

These administrative and support services can, for example, be adapted to the specific needs of electronic commerce value chains in any number of vertical markets, including a wide variety of entertainment applications. Electronic commerce participants can, for example, use these administrative and support services to support their interests, and/or they can shape and reuse these services in response to competitive business realities. Non-exhaustive examples of electronic commerce participants include individual creators, film and music studios, distributors, program aggregators, broadcasters, and cable and satellite operators.

The Distributed Commerce Utility can, for example, make optimally efficient use of commerce administration resources, and can, in at least some embodiments, scale in a practical fashion to

optimally accommodate the demands of electronic commerce growth.

The Distributed Commerce Utility may, for example, comprise a number of Commerce Utility Systems. These Commerce Utility Systems can provide a web of infrastructure support available to, and reusable by, the entire electronic community and/or many or all of its participants. Different support functions can, for example, be collected together in hierarchical and/or in networked relationships to suit various business models and/or other objectives. Modular support functions can, for example, be combined in different arrays to form different Commerce Utility Systems for different design implementations and purposes. These Commerce Utility Systems can, for example, be distributed across a large number of electronic appliances with varying degrees of distribution.

The "Distributed Commerce Utility" provides numerous additional capabilities and benefits that can be used in conjunction with the particular embodiments shown in the drawings of this application, non-exhaustive examples of which include:

- Enables practical and efficient electronic commerce and rights management.

- Provides services that securely administer and support electronic interactions and consequences.

- Provides infrastructure for electronic commerce and other forms of human electronic interaction and relationships.

- Optimally applies the efficiencies of modern distributed computing and networking.

- Provides electronic automation and distributed processing.

- Supports electronic commerce and communications infrastructure that is modular, programmable, distributed and optimally computerized.

- Provides a comprehensive array of capabilities that can be combined to support services that perform various administrative and support roles.

- Maximizes benefits from electronic automation and distributed processing to produce optimal allocation and use of resources across a system or network.

- Is efficient, flexible, cost effective, configurable, reusable, modifiable, and generalizable.

- Can economically reflect users' business and privacy requirements.

- Can optimally distribute processes -- allowing commerce models to be flexible, scaled to demand and to match user requirements.

- Can efficiently handle a full range of activities and service volumes.

- Can be fashioned and operated for each business model, as a mixture of distributed and centralized processes.

- Provides a blend of local, centralized and networked capabilities that can be uniquely shaped and reshaped to meet changing conditions.

- Supports general purpose resources and is reusable for many different models; in place infrastructure can be reused by different value chains having different requirements.

- Can support any number of commerce and communications models.

- Efficiently applies local, centralized and networked resources to match each value chain's requirements.

- Sharing of common resources spreads out costs and maximizes efficiency.

- Supports mixed, distributed, peer-to-peer and centralized networked capabilities.

- Can operate locally, remotely and/or centrally.

- Can operate synchronously, asynchronously, or support both modes of operation.

- Adapts easily and flexibly to the rapidly changing sea of commercial opportunities, relationships and constraints of "Cyberspace."

Any or all of these features may be used in combination with the inventions disclosed herein.

The Distributed Commerce Utility provides, among other advantages, comprehensive, integrated administrative and support services for secure electronic commerce and other forms of electronic interaction. These electronic interactions supported by the Distributed Commerce Utility may, in at least some embodiments, entail the broadest range of appliances and distribution media, non-limiting examples of which include networks and other communications channels, consumer appliances, computers, convergent devices such as WebTV, and optical media such as CD-ROM and DVD in all their current and future forms.

Example Access Techniques

Figures 3, 4A and 4B show example access techniques provided by player 52. In this example, upon disk 100 being loaded into player disk drive 80 (Figure 4A, block 400), the player controller 82 may direct drive 80 to fetch hidden keys 210 from disk 100 and use them to decrypt some or all of the encrypted key block 208 (Figure 4A, block 402). In this example, drive 80 may

store the keys so decrypted without exposing them to player controller 82 (e.g., by storing them within key store 212 within a secure decryption component such as an integrated circuit based decryption engine) (Figure 4A, block 404). The player 52 may control drive 80 to read the control set 204 (which may or may not be encrypted) from disk 100 (Figure 4A, block 406). The player microprocessor 82 may parse control set 204, ignore or discard those controls 222 that are beyond its capability, and maintain permissions and/or rights management information corresponding to the subset of controls that it can enforce (e.g., the "copy once" control 222(1)).

Player 52 may then wait for the user to provide a request via control inputs 58 and/or remote control unit 56. If the control input is a copy request ("yes" exit to Figure 4A, decision block 408), then player microprocessor 84 may query control 222(1) to determine whether copying is allowed, and if so, under what conditions (Figure 4A, decision block 410). Player 52 may refuse to copy the disk 100 if the corresponding control 222(1) forbids copying ("no" exit to Figure 4A, decision block 410), and may allow copying (e.g., by controlling drive 80 to sequentially access

all of the information on disk 100 and provide it to an output port not shown) if corresponding control 222(1) permits copying ("yes" exit to Figure 4A, decision block 410; block 412). In this example, player 52 may, upon making a copy, store an identifier associated with disk 100 within an internal, non-volatile memory (e.g., controller memory 86) or elsewhere if control 222(1) so requires. This stored disk identifier can be used by player 52 to enforce a "copy once" restriction (i.e., if the user tries to use the same player to copy the same disk more than once or otherwise as forbidden by control 222(1), the player can deny the request).

If the user requests one of properties 200 to be played or read ("yes" exit to Figure 4A, decision block 414), player controller 82 may control drive 80 to read the corresponding information from the selected property 200 (e.g., in a sequence as specified by metadata 202) and decrypt the read information as needed using the keys initially obtained from key block 208 and now stored within drive key storage 212 (Figure 4A, block 416).

Figure 4B is a variation on the Figure 4A process to accommodate a situation in which player 52 itself provides decryption keys for decrypting encrypted key block 208. In this

example, controller 82 may supply one or more decryption keys to drive 80 using a secure protocol such a Diffie-Hellman key agreement, or through use of a shared key known to both the drive and some other system or component to which the player 52 is or once was coupled (Figure 4B, block 403). The drive 80 may use these supplied keys to decrypt encrypted key block 208 as shown in Figure 4A, block 404, or it may use the supplied keys to directly decrypt content such as protected property 200 and/or protected metadata 202(2).

As a further example, the player 52 can be programmed to place a copy it makes of a digital property such as a film in encrypted form inside a tamper-resistant software container. The software container may carry with it a code indicating that the digital property is a copy rather than an original. The sending player 52 may also put its own unique identifier (or the unique identifier of an intended receiving device such as another player 52, a video cassette player or equipment 50) in the same secure container to enforce a requirement that the copy can be played only on the intended receiving device. Player 52 (or other receiving device) can be programmed to make no copies (or no

additional copies) upon detecting that the digital property is a copy rather than an original. If desired, a player 52 can be programmed to refuse to play a digital property that is not packaged with the player's unique ID.

Example Use of Analog Encoding Techniques

In another example, more comprehensive rights management information may be encoded by player 52 in the analog output using methods for watermarking and/or fingerprinting.Today, a substantial portion of the "real world" is analog rather than digital. Despite the pervasiveness of analog signals, existing methods for managing rights and protecting copyright in the analog realm are primitive or non-existent. For example:

- Quality degradation inherent in multigenerational analog copying has not prevented a multi-billion dollar pirating industry from flourishing.

- Some methods for video tape copy and pay per view protection attempt to prevent any copying at all of commercially released content, or allow only one

generation of copying. These methods can generally be easily circumvented.

- Not all existing devices respond appropriately to copy protection signals.

- Existing schemes are limited for example to "copy/no copy" controls.

- Copy protection for sound recordings has not been commercially implemented.

A related problem relates to the conversion of information between the analog and digital domains. Even if information is effectively protected and controlled initially using strong digital rights management techniques, an analog copy of the same information may no longer be securely protected.

For example, it is generally possible for someone to make an analog recording of program material initially delivered in digital form. Some analog recordings based on digital originals are of quite good quality. For example, a Digital Versatile Disk

("DVD") player may convert a movie from digital to analog format and provide the analog signal to a high quality analog home VCR. The home VCR records the analog signal. A consumer now has a high quality analog copy of the original digital property. A person could re-record the analog signal on a DVD-RAM. This recording will in many circumstances have substantial quality – and would no longer be subject to "pay per view" or other digital rights management controls associated with the digital form of the same content.

Since analog formats will be with us for a long time to come, rightsholders such as film studios, video rental and distribution companies, music studios and distributors, and other value chain participants would very much like to have significantly better rights management capabilities for analog film, video, sound recordings and other content. Solving this problem generally requires a way to securely associate rights management information with the content being protected.

In combination with other rights management capabilities, watermarking and/or fingerprinting, may provide "end to end"

secure rights management protection that allows content providers and rights holders to be sure their content will be adequately protected -- irrespective of the types of devices, signaling formats and nature of signal processing within the content distribution chain. This "end to end" protection also allows authorized analog appliances to be easily, seamlessly and cost-effectively integrated into a modern digital rights management architecture.

Watermarking and/or fingerprinting may carry, for example, control information that can be a basis for a Virtual Distribution Environment ("VDE") in which electronic rights management control information may be delivered over insecure (e.g., analog) communications channels. This Virtual Distribution Environment is highly flexible and convenient, accommodating existing and new business models while also providing an unprecedented degree of flexibility in facilitating ad hoc creation of new arrangements and relationships between electronic commerce and value chain participants -- regardless of whether content is distributed in digital and/or analog formats.

Watermarking together with distributed, peer-to-peer rights management technologies providers numerous advantages, including, but not limited to:

- An indelible and invisible, secure technique for providing rights management information.

- An indelible method of associating electronic commerce and/or rights management controls with analog content such as film, video, and sound recordings.

- Persistent association of the commerce and/or rights management controls with content from one end of a distribution system to the other -- regardless of the number and types of transformations between signaling formats (for example, analog to digital, and digital to analog).

- The ability to specify "no copy/ one copy/ many copies" rights management rules, and also more

complex rights and transaction pricing models (such as, for example, "pay per view" and others).

- The ability to fully and seamlessly integrate with comprehensive, general electronic rights management solutions.

- Secure control information delivery in conjunction with authorized analog and other non-digital and/or non-secure information signal delivery mechanisms.

- The ability to provide more complex and/or more flexible commerce and/or rights management rules as content moves from the analog to the digital realm and back.

- The flexible ability to communicate commerce and/or rights management rules implementing new, updated, or additional business models to authorized analog and/or digital devices.

Any or all of these features may be used in combination in and/or with the inventions disclosed in the present specification.

Briefly, watermarking and/or fingerprinting methods may, using "steganographical" techniques, substantially indelibly and substantially invisibly encode rights management and/or electronic commerce rules and controls within an information signal such as, for example, an analog signal or a digitized (for example, sampled) version of an analog signal, non-limiting examples of which may include video and/or audio data, that is then decoded and utilized by the local appliance. The analog information and stenographically encoded rights management information may be transmitted via many means, non-limiting examples of which may include broadcast, cable TV, and/or physical media, VCR tapes, to mention one non-limiting example. Any or all of these techniques may be used in combination in accordance with the inventions disclosed herein.

Watermarking and/or fingerprinting methods enable at least some rights management information to survive transformation of the video and/or other information from analog to digital and from

digital to analog format. Thus in one example, two or more analog and/or digital appliances may participate in an end-to-end fabric of trusted, secure rights management processes and/or events.

Example, More Capable Embodiments

As discussed above, the example control set shown in Figure 3B provides a comprehensive, flexible and extensible set of controls for use by both player 52 and computer equipment 62 (or other platform) depending upon the particular technical, security and other capabilities of the platform. In this example, player 52 has only limited technical and security capabilities in order to keep cost and complexity down in a mass-produced consumer item, and therefore may essentially ignore or fail to enable some or all of the controls 222 provided within control set 204. In another example, the cost of memory and/or processors may continue to decline and manufacturers may choose to expand the technical and security capabilities of player 52. A more capable player 52 will provide more powerful, robust, and flexible rights management capabilities.

Figure 5 shows an example arrangement permitting platform 60 including secure node 72 to have enhanced and/or different capabilities to use information and/or rights management information on disk 100, and Figure 6 shows an example access technique provided by the secure node. Referring to Figure 5, secure node 72 may be coupled to a network 150 whereas player 52 may not be—giving the secure node great additional flexibility in terms of communicating security related information such as audit trails, compensation related information such as payment requests or orders, etc. This connection of secure node 72 to network 150 (which may be replaced in any given application by some other communications technique such as insertion of a replaceable memory cartridge) allows secure node 72 to receive and securely maintain rights management control information such as an additional container 206' containing an additional control set 204'. Secure node 72 may use control set 204' in addition or in lieu of a control set 204 stored on disk 100. Secure node 72 may also maintain a secure cryptographic key store 212 that may provide cryptographic keys to be used in lieu of or in addition to any keys 208, 210 that may be stored on disk 100.

Because of its increased security and/or technical capabilities, secure node 72 may be able to use controls 222 within control set 204 that player 52 ignores or cannot use—and may be provided with further and/or enhanced rights and/or rights management capabilities based on control set 204' (which the user may, for example, order specially and which may apply to particular properties 200 stored on disk 100 and/or particular sets of disks).

Example Secure Node Access Techniques

The Figure 6 example access technique (which may be performed by platform 60 employing secure node 72, for example) involves, in this particular example, the secure node 72 fetching property identification information 220 from disk 100 (Figure 6, block 502), and then locating applicable control sets and/or rules 204 (which may be stored on disk 100, within secure node 72, within one or more repositories the secure node 72 accesses via network 150, and/or a combination of any or all of these techniques) (Figure 6, block 504). Secure node 72 then loads the necessary decryption keys and uses them to decrypt information as required (Figure 6, block 506). In one example, secure node 72 obtains the necessary keys from secure containers 206 and/or 206'

and maintains them within a protected processing environment such as SPU 164 or a software-emulated protected processing environment without exposing them externally of that environment. In another example, the secure node 72 may load the necessary keys (or a subset of them) into disk drive 82' using a secure key exchange protocol for use by the disk drive in decrypting information much in the same manner as would occur within player 52 in order to maintain complete compatibility in drive hardware.

Secure node 72 may monitor user inputs and perform requested actions based on the particular control set 204, 204'. For example, upon receiving a user request, secure node 72 may query the control set 204, 204' to determine whether it (they) permits the action the user has requested (Figure 6, block 508) and, if permitted, whether conditions for performing the requested operation have been satisfied (Figure 6, block 510). In this example, secure node 72 may effect the operations necessary to satisfy any such required conditions such as by, for example, debiting a user's locally-stored electronic cash wallet, securely requesting an account debit via network 150, obtaining and/or

checking user certificates to ensure that the user is within an appropriate class or is who he or she says he is, etc.—using network 150 as required (Figure 6, block 510). Upon all necessary conditions being satisfied, secure node 72 may perform the requested operation (and/or enable microprocessor 154 to perform the operation) (e.g., to release content) and may then generate secure audit records which can be maintained by the secure node and/or reported at the time or later via network 150 (Figure 6, block 512).

If the requested operation is to release content (e.g., make a copy of the content), platform 60 (or player 52 in the example above) may perform the requested operation based at least in part on the particular controls that enforce rights over the content. For example, the controls may prevent platform 60 from releasing content except to certain types of output devices that cannot be used to copy the content, or they may release the content in a way that discourages copying (e.g., by "fingerprinting" the copy with an embedded designation of who created the copy, by intentionally degrading the released content so that any copies made from it will be inferior, etc.). As one specific example, a

video cassette recorder (not shown) connected to platform 60 may be the output device used to make the copy. Because present generations of analog devices such as video cassette recorders are incapable of making multigenerational copies without significant loss in quality, the content provider may provide controls that permit content to be copied by such analog devices but not by digital devices (which can make an unlimited number of copies without quality loss). For example, platform 60 may, under control of digital controls maintained by secure node 72, release content to the video cassette recorder only after the video cassette recorder supplies the platform a digital ID that designates the output device as a video cassette recorder -- and may refuse to provide any output at all unless such a digital ID identifying the output device as a lower quality analog device is provided. Additionally or in the alternative, platform 60 may intentionally degrade the content it supplies to the video cassette recorder to ensure that no acceptable second -generation copies will be made. In another example, more comprehensive rights management information may be encoded by platform 60 in the analog output using watermarking and/or fingerprinting.

Additional Examples of Secure Container Usage

Figure 7 shows a basic example of a DVD medium 700 containing a kind of secure container 701 for use in DVDs in accordance with the present invention. As shown in this example, container 701 ("DigiBox for DVDs") could be a specialized version of a "standard" container tailored especially for use with DVD and/or other media, or it could, alternatively (in an arrangement shown later in Figure 8), be a fully "standard" container. As shown in this example, the specialized container 701 incorporates features that permit it to be used in conjunction with content information, metadata, and cryptographic and/or protection information that is stored on the DVD medium 700 in the same manner as would have been used had container 701 not been present. Thus, specialized container 701 provides compatibility with existing data formats and organizations used on DVDs and/or other media. In addition, a specialized container 701 can be tailored to support only those features necessary for use in support of DVD and/or other media, so that it can be processed and/or manipulated using less powerful or less expensive computing resources than would be required for complete support of a "standard" container object.

In this example, specialized "DVD only" container 701 includes a content object (a property) 703 which includes an "external reference" 705 to video title content 707, which may be stored on the DVD and/or other medium in the same manner as would have been used for a medium not including container 701. The video title content 707 may include MPEG-2 and/or AC-3 content 708, as well as scrambling (protection) information 710 and header, structure and/or meta data 711. External reference 705 contains information that "designates" (points to, identifies, and/or describes) specific external processes to be applied/executed in order to usecontent and other information not stored in container 701. In this example, external reference 705 designates video title content 707 and its components 708, 710, and 711. Alternatively, container 701 could store some or all of the video title content in the container itself, using a format and organization that is specific to container 701, rather than the standard format for the DVD and/or other medium 700.

In this example, container 701 also includes a control object (control set) 705 that specifies the rules that apply to use of video title content 707. As indicates by solid arrow 702, control object

705 "applies to" content object (property) 703. As shown in this example, rule 704 can specify that protection processes, for example CGMA or the Matsushita data scrambling process, be applied, and can designate, by external reference 709 contained in rule 704, data scrambling information 710 to be used in carrying out the protection scheme. The shorthand "do CGMA" description in rule 704 indicates that the rule requires that the standard CGMA protection scheme used for content on DVD media is to be used in conjunction with video title content 707, but a different example could specify arbitrary other rules in control object 705 in addition to or instead of the "do CGMA" rule, including other standard DVD protection mechanisms such as the Matsushita data scrambling scheme and/or other rights management mechanisms. External reference 709 permits rule 704 to be based on protection information 710 that is stored and manipulated in the same format and manner as for a DVD medium that does not incorporate container 701 and/or protection information that is meaningful only in the context of processing container 701.

Figure 8 shows a example of a DVD medium 800 containing a "standard" secure container 801. In this example, the

"standard" container provides all of the functionality (if desired) of the Figure 7 container, but may offer additional and/or more extensive rights management and/or content use capabilities than available on the "DVD only" container (e.g., the capacity to operate with various different platforms that use secure nodes).

Figure 9 shows a more complex example of DVD medium 800 having a standard container 901 that provides all of the functionality (if desired) of the Figure 7 container, and that can function in concert with other standard containers 902 located either on the same DVD medium or imported from another remote secure node or network. In this example, standard container 902 may include a supplementary control object 904 which applies to content object 903 of standard container 901. Also in this example, container 902 may provide an additional rule(s) such as, for example, a rule permitting/extending rights to allow up to a certain number (e.g., five) copies of the content available on DVD 900. This arrangement, for example, provides added flexibility in controlling rights management of DVD content between multiple platforms via access through "backchannels" such as via a set-top

box or other hardware having bi-directional communications capabilities with other networks or computers.

Additional Use of A DVD Disk With A Secure Container

Figure 10 illustrates the use of a "new" DVD disk—i.e., one that includes a special DVD secure container in the medium. This container may, in one example, be used or two possible use scenarios: a first situation in which the disk is used on an "old" player (DVD appliance, i.e., a DVD appliance that is not equipped with a secure node to provide rights management in accordance with the present invention; and a second situation in which the disk is used on a "new" player—i.e., a DVD appliance which is equipped with a secure node to provide rights management in accordance with the present invention. In this example, a secure node within the "new" player is configured with the necessary capabilities to process other copy protection information such as, for example, CGMA control codes and data scrambling formats developed and proposed principally by Matsushita.

For example, in the situation shown in Figure 10, the "new" player (which incorporates a secure node in accordance with the

present invention) can recognize the presence of a secure container on the disk. The player may then load the special DVD secure container from the disk into the resident secure node. The secure node opens the container, and implements and/or enforces appropriate rules and usage consequences associated with the content by applying rules from the control object. These rules are extremely flexible. In one example, the rules may, for example, call for use of other protection mechanisms (such as, for example, CGMA protection codes and Matsushita data scrambling) which can be found in the content (or property) portion of the container.

In another example shown in Figure 10, the special DVD container on the disk still allows the "old" player to use to a predetermined limited amount content material which may be used in accordance with conventional practices.

Example Use of A DVD Disk With No Secure Container

Referring now to Figure 11, a further scenario is discussed. Figure 11 illustrates use of an "old" DVD disk with two possible use examples: a first example in which the disk is used on an "old" player—i.e., a DVD appliance that is not equipped with a secure

node for providing rights management in accordance with the present invention—and a second example in which the disk is used on a "new" player (i.e., equipped with a secure node).

In the first case, the "old" player will play the DVD content in a conventional manner. In the second scenario, the "new" player will recognize that the disk does not have a container stored in the medium. It therefore constructs a "virtual" container in resident memory of the appliance. To do this, it constructs a container content object, and also constructs a control object containing the appropriate rules. In one particular example, the only applicable rule it need apply is to "do CGMA" -- but in other examples, additional and/or different rules could be employed. The virtual container is then provided to the secure node within the "new" player for implementing management of use rights in accordance with the present invention. Although not shown in Figures 10 and 11, use of "external references" may also be provided in both virtual and non-virtual containers used in the DVD context.

Example Illustrative Arrangements for Sharing, Brokering and Combining Rights When Operating in At Least Occasionally Connected Scenarios

As described above, the rights management resources of several different devices and/or other systems can be flexibly combined in diverse logical and/or physical relationships, resulting for example in greater and/or differing rights. Such rights management resource combinations can be effected through connection to one or more remote rights authorities. Figures 12-14 show some non-limiting examples of how rights authorities can be used in various contexts.

For example, Figure 12 shows a rights authority broker 1000 connected to a local area network (LAN) 1002. LAN 1002 may connect to wide area network if desired. LAN 1002 provides connectivity between rights authority broker 1000 and any number of appliances such as for example a player 50, a personal computer 60, a CD "tower" type server 1004. In the example shown, LAN 1002 includes a modem pool (and/or network

protocol server, not shown)1006 that allows a laptop computer 1008 to connect to the rights authority broker 1000 via dial-up lines 1010. Alternatively, laptop 1008 could communicate with rights authority broker 1000 using other network and/or communication means, such as the Internet and/or other Wide Area Networks (WANs). A disk player 50A may be coupled to laptop 1008 at the laptop location. In accordance with the teachings above, any or all of devices shown in Figure 12 may include one or more secure nodes 72.

Rights authority broker 1000 may act as an arbiter and/or negotiator of rights. For example, laptop 1008 and associated player 50A may have only limited usage rights when operating in a stand-alone configuration. However, when laptop 1008 connects to rights authority broker 1000 via modem pool 1006 and LAN 1002 and/or by other communication means, the laptop may acquire different and/or expanded rights to use disks 100 (e.g., availability of different content portions, different pricing, different extraction and/or redistribution rights, etc.) Similarly, player 50, equipment 60 and equipment 1004 may be provided with an enhanced and/or different set of disk usage rights through

communication with rights authority broker 1000 over LAN 1002. Communication to and from rights authority broker 1000 is preferably secured through use of containers of the type disclosed in the above-referenced Ginter et al. patent specification.

Figure 13 shows another example use of a rights authority broker 1000 within a home environment. In this example, the laptop computer 1008 may be connected to a home-based rights authority broker 1000 via a high speed serial IEEE 1394 bus and/or by other electronic communication means. In addition, rights authority broker 1000 can connect with any or all of:

- a high definition television 1100,
- one or more loudspeakers 1102 or other audio transducers,
- one or more personal computers 60,
- one or more set-top boxes 1030,
- one or more disk players 50,
- one or more other rights authority brokers 1000A-1000N and

- any other home or consumer equipment or appliances.

Any or all of the equipment listed above may include a secure node 72.

Figure 14 shows another example use of a rights authority broker 1000. In this example, rights authority broker 1000 is connected to a network 1020 such as a LAN, a WAN, the Internet, etc. Network 1020 may provide connectivity between rights authority broker 1000 and any or all of the following equipment:

- one or more connected or occasionally connected disk players 50A, 50B;
- one more networked computers 1022;
- one or more disk reader towers/servers 1004;
- one or more laptop computers 1008;
- one or more Commerce Utility Systems such as a rights and permissions clearinghouse 1024 (see Shear et al., "Trusted Infrastructure..." specification referenced above);

- one or more satellite or other communications uplinks 1026;
- one or more cable television head-ends 1028;
- one or more set-top boxes 1030 (which may be connected to satellite downlinks 1032 and/or disk players 50C);
- one or more personal computer equipment 60;
- one or more portable disk players 1034 (which may be connected through other equipment, directly, and/or occasionally unconnected;
- one or more other rights authority brokers 1000A-1000N; and
- any other desired equipment.

Any or all of the above-mentioned equipment may include one or more secure nodes 72. Rights authority broker 1000 can distribute and/or combine rights for use by any or all of the other components shown in Figure 14. For example, rights authority broker 100 can supply further

secure rights management resources to equipment connected to the broker via network 1020. Multiple equipment shown in Figure 14 can participate and work together in a permanently or temporarily connected network 1020 to share the rights management for a single node. Rights associated with parties and/or groups using and/or controlling such multiple devices and/or other systems can be employed according to underlying rights related rules and controls. As one example, rights available through a corporate executive's laptop computer 1008 might be combined with or substituted for, in some manner, the rights of one or more subordinate corporate employees when their computing or other devices 60 are coupled to network 1020 in a temporary networking relationship. In general, this aspect of the invention allows distributed rights management for DVD or otherwise packaged and delivered content that is protected by a distributed, peer-to-peer rights management. Such a distributed rights management can operate whether the DVD appliance or other content usage device is participating in a permanently or temporarily

connected network 1020, and whether or not the relationships among the devices and/or other systems participating in the distributed rights management arrangement are relating temporarily or have a more permanent operating relationship.

For example, laptop computer 1008 may have different rights available depending on the context in which that device is operating. For example, in a general corporate environment such as shown in Figure 12, the laptop 1008 may have one set of rights. However, the same laptop 1008 may be given a different set of rights when connected to a more general network 1020 in collaboration with specified individuals and/or groups in a corporation. The same laptop 1008 may be given a still different set of rights when connected in a general home environment such as shown by example in Figure 13. The same laptop 1008 could be given still different rights when connected in still other environments such as, by way of non-limiting example:

- a home environment in collaboration with specified individuals and/or groups,

- a retail environment,
- a classroom setting as a student,
- a classroom setting in collaboration with an instructor, in a library environment,
- on a factory floor,
- on a factory floor in collaboration with equipment enabled to perform proprietary functions, and so on.

As one more particular example, coupling a limited resource device arrangement such as a DVD appliance 50 shown in Figure 14 with an inexpensive network computer (NC) 1022 may allow an augmenting (or replacing) of rights management capabilities and/or specific rights of parties and/or devices by permitting rights management to be a result of a combination of some or all of the rights and/or rights management capabilities of the DVD appliance and those of an Network or Personal Computer (NC or PC). Such rights may be further augmented, or otherwise modified or replaced by the availability of rights management capabilities provided by a trusted (secure) remote network rights authority 1000.

The same device, in this example a DVD appliance 50, can thus support different arrays, e.g., degrees, of rights management capabilities, in disconnected and connected arrangements and may further allow available rights to result from the availability of rights and/or rights management capabilities resulting from the combination of rights management devices and/or other systems. This may include one or more combinations of some or all of the rights available through the use of a "less" secure and/or resource poor device or system which are augmented, replaced, or otherwise modified through connection with a device or system that is "more" or "differently" secure and/or resource rich and/or possesses differing or different rights, wherein such connection employs rights and/or management capabilities of either and/or both devices as defined by rights related rules and controls that describe a shared rights management arrangement.

In the latter case, connectivity to a logically and/or physically remote rights management capability can expand (by, for example, increasing the available secure rights management resources) and/or change the character of the rights available to the user of the DVD appliance 50 or a DVD appliance when such

device is coupled with an NC 1022, personal computer 60, and/or remote rights authority 1000. In this rights augmentation scenario, additional content portions may be available, pricing may change, redistribution rights may change (e.g., be expanded), content extraction rights may be increased, etc.

Such "networking rights management" can allow for a combination of rights management resources of plural devices and/or other systems in diverse logical and/or physical relationships, resulting in either greater or differing rights through the enhanced resources provided by connectivity with one or more "remote" rights authorities. Further, while providing for increased and/or differing rights management capability and/or rights, such a connectivity based rights management arrangement can support multi-locational content availability, by providing for seamless integration of remotely available content, for example, content stored in remote, Internet world wide web-based, database supported content repositories, with locally available content on one or more DVD discs 100.

In this instance, a user may experience not only increased or differing rights but may be able to use to both local DVD content

and supplementing content (i.e., content that is more current from a time standpoint, more costly, more diverse, or complementary in some other fashion, etc.). In such an instance, a DVD appliance 50 and/or a user of a DVD appliance (or other device or system connected to such appliance) may have the same rights, differing, and/or different rights applied to locally and remotely available content, and portions of local and remotely available content may themselves be subject to differing or different rights when used by a user and/or appliance. This arrangement can support an overall, profound increase in user content opportunities that are seamlessly integrated and efficiently available to users in a single content searching and/or usage activity.

Such a rights augmenting remote authority 1000 may be directly coupled to a DVD appliance 50 and/or other device by modem (see item 1006 in Figure 12) and/or directly or indirectly coupled through the use of an I/O interface, such as a serial 1394 compatible controller (e.g., by communicating between a 1394 enabled DVD appliance and a local personal computer that functions as a smart synchronous or asynchronous information communications interface to such one or more remote authorities,

including a local PC 60 or NC 1022 that serves as a local rights management authority augmenting and/or supplying the rights management in a DVD appliance) and/or by other digital communication means such as wired and/or wireless network connections.

Rights provided to, purchased, or otherwise acquired by a participant and/or participant DVD appliance 50 or other system can be exchanged among such peer-to-peer relating devices and/or other systems so long as they participate in a permanently or temporarily connected network. 1020. In such a case, rights may be bartered, sold, for currency, otherwise exchanged for value, and/or loaned so long as such devices and/or other systems participate in a rights management system, for example, such as the Virtual Distribution Environment described in Ginter, et al., and employ rights transfer and other rights management capabilities described therein. For example, this aspect of the present invention allows parties to exchange games or movies in which they have purchased rights. Continuing the example, an individual might buy some of a neighbor's usage rights to watch a movie, or transfer to another party credit received from a game

publisher for the successful superdistribution of the game to several acquaintances, where such credit is transferred (exchanged) to a friend to buy some of the friend's rights to play a different game a certain number of times, etc.

Example Virtual Rights Process

Figures 15A-15C shows an example of a process in which rights management components of two or more appliances or other devices establish a virtual rights machine environment associated with an event, operation and/or other action. The process may be initiated in a number of ways. In one example, an appliance user (and/or computer software acting on behalf of a user, group of users, and/or automated system for performing actions) performs an action with a first appliance (e.g., requesting the appliance to display the contents of a secure container, extract a portion of a content element, run a protected computer program, authorize a work flow process step, initiate an operation on a machine tool, play a song, etc.) that results in the activation of a rights management component associated with such first appliance (Figure 15A, block 1500). In other examples, the process may get started in response to an automatically generated event (e.g., based

on a time of day or the like), a random or pseudo-random event, and/or a combination of such events with a user-initiated event.

Once the process begins, a rights management component such as a secure node 72 (for example, an SPE and/or HPE as disclosed in Ginter et al.) determines which rights associated with such first appliance, if any, the user has available with respect to such an action (Figure 15A, block 1502). The rights management component also determines the coordinating and/or cooperating rights associated with such an action available to the user located in whole or in part on other appliances (Figure 15A, block 1502).

In one example, these steps may be performed by securely delivering a request to a rights authority server 1000 that identifies the first appliance, the nature of the proposed action, and other information required or desired by such a rights authority server. Such other information may include, for example:

- the date and time of the request,
- the identity of the user,
- the nature of the network connection,

- the acceptable latency of a response, etc.), and/or
- any other information.

In response to such a request, the rights authority server 1000 may return a list (or other appropriate structure) to the first appliance. This list may, for example, contain the identities of other appliances that do, or may, have rights and/or rights related information relevant to such a proposed action.

In another embodiment, the first appliance may communicate (e.g., poll) a network with requests to other appliances that do, or may, have rights and/or rights related information relevant to such proposed action. Polling may be desirable in cases where the number of appliances is relatively small and/or changes infrequently. Polling may also be useful, for example, in cases where functions of a rights authority server 1000 are distributed across several appliances.

The rights management component associated with the first appliance may then, in this example, check the security level(s) (and/or types) of devices and/or users of other appliances that do, or may, have rights and/or rights related information relevant to

such an action (Figure 15A, block 1506). This step may, for example, be performed in accordance with the security level(s) and/or device type management techniques disclosed in Sibert and Van Wie, and the user rights, secure name services and secure communications techniques disclosed in Ginter et al. Device and/or user security level determination may be based, for example, in whole or in part on device and/or user class.

The rights management component may then make a decision as to whether each of the other appliance devices and/or users have a sufficient security level to cooperate in forming the set of rights and/or rights related information associated with such an action (Figure 15A, block 1508). As each appliance is evaluated, some devices and/or users may have sufficient security levels, and others may not. In this example, if a sufficient security level is not available ("No" exit to decision block 1508), the rights management component may create an audit record (for example, an audit record of the form disclosed in Ginter et al.) (Figure 15A, block 1510), and may end the process (Figure 15A, block 1512). Such audit record may be for either immediate transmission to a responsible authority and/or for local storage and later

transmission, for example. The audit recording step may include, as one example, incrementing a counter that records security level failures (such as the counters associated with summary services in Ginter et al.)

If the devices and/or users provide the requisite security level ("Yes" exit to block 1508), the rights management component in this example may make a further determination based on the device and/or user class(es) and/or other configuration and/or characteristics (Figure 15B, block 1514). Such determination may be based on any number of factors such as for example:

- the device is accessible only through a network interface that has insufficient throughput;
- devices in such a class typically have insufficient resources to perform the action, or relevant portion of the action, at all or with acceptable performance, quality, or other characteristics;

- the user class is inappropriate due to various conditions (e.g., age, security clearance, citizenship, jurisdiction, or any other class-based or other user characteristic); and/or
- other factors.

In one example, decision block 1514 may be performed in part by presenting a choice to the user that the user declines.

If processes within the rights management component determines that such device and/or user class(es) are inappropriate("No" exit to block 1514), the rights management component may write an audit record if required or desired (Figure 15B, block 1516) and the process may end (Figure 15B, block 1518).

If, on the other hand, the rights management component determines that the device and/or user classes are appropriate to proceed ("Yes" exit to block 1514), the rights management component may determine the rights and resources available for performing the action on the first appliance and the other appliances acting together (Figure 15B, block 1520). This step may be performed, for example, using any or all of the method

processing techniques disclosed in Ginter et al. For example, method functions may include event processing capabilities that formulate a request to each relevant appliance that describes, in whole or in part, information related to the action, or portion of the action, potentially suitable for processing, in whole or in part, by such appliance. In this example, such requests, and associated responses, may be managed using the reciprocal method techniques disclosed in Ginter et al. If such interaction requires additional information, or results in ambiguity, the rights management component may, for example, communicate with the user and allow them to make a choice, such as making a choice among various available, functionally different options, and/or the rights management component may engage in a negotiation (for example, using the negotiation techniques disclosed in Ginter et al.) concerning resources, rights and/or rights related information.

The rights management component next determines whether there are sufficient rights and/or resources available to perform the requested action (Figure 15B, decision block 1522). If there are insufficient rights and/or resources available to perform the action ("No" exit to block 1522), the rights management component may

write an audit record (Figure 15B, block 1524), and end the process (Figure 15B, block 1526).

In this example, if sufficient rights and/or resources are available ("Yes" exit to block 1522), the rights management component may make a decision regarding whether additional events should be processed in order to complete the overall action (Figure 15B, block 1528). For example, it may not be desirable to perform only part of the overall action if the necessary rights and/or resources are not available to complete the action. If more events are necessary and/or desired ("Yes" exit to block 1528), the rights management component may repeat blocks 1520, 1522 (and potentially perform blocks 1524, 1526) for each such event.

If sufficient rights and/or resources are available for each of the events ("No" exit to block 1528), the rights management component may, if desired or required, present a user with a choice concerning the available alternatives for rights and/or resources for performing the action (Figure 15B, block 1530). Alternatively and/or in addition, the rights management component may rely on user preference information (and/or defaults) to "automatically" make such a determination on behalf

of the user (for example, based on the overall cost, performance, quality, etc.). In another embodiment, the user's class, or classes, may be used to filter or otherwise aid in selecting among available options. In still another embodiment, artificial intelligence (including, for example, expert systems techniques) may be used to aid in the selection among alternatives. In another embodiment, a mixture of any or all of the foregoing (and/or other) techniques may be used in the selection process.

If there are no acceptable alternatives for rights and/or resources, or because of other negative aspects of the selection process (e.g., a user presses a "Cancel" button in a graphical user interface, a user interaction process exceeds the available time to make such a selection, etc.), ("No" exit to block 1530) the rights management component may write an audit record (Figure 15B, block 1532), and end the process (Figure 15B, block 1534).

On the other hand, if a selection process identifies one or more acceptable sets of rights and/or resources for performing the action and the decision to proceed is affirmative ("Yes" exit to block 1530), the rights management component may perform the proposed action using the first appliance alone or in combination

with any additional appliances (e.g., a rights authority 1000, or any other connected appliance) based on the selected rights and/or resources (Figure 15C, block 1536). Such cooperative implementation of the proposed actions may include for example:

- performing some or all of the action with the first appliance;
- performing some or all of the action with one or more appliances other than the first appliance (e.g., a rights authority 1000 and/or some other appliance);
- performing part of the action with the first appliance and part of the action with one or more other appliances; or
- any combination of these.

For example, this step may be performed using the event processing techniques disclosed in Ginter et al.

As one illustrative example, the first appliance may have all of the resources necessary to perform a particular task (e.g., read certain information from an optical disk), but may lack the rights necessary to do so. In such an instance, the first appliance may

obtain the additional rights it requires to perform the task through the steps described above. In another illustrative example, the first appliance may have all of the rights required to perform a particular task, but it may not have the resources to do so. For example, the first appliance may not have sufficient hardware and/or software resources available to it for accessing, processing or otherwise using information in certain ways. In this example, step 1536 may be performed in whole or in part by some other appliance or appliances based in whole or in part on rights supplied by the first appliance. In still another example, the first appliance may lack both rights and resources necessary to perform a certain action, and may rely on one or more additional appliances to supply such resources and rights.

In this example, the rights management component may, upon completion of the action, write one or more audit records (Figure 15C, block 1538), and the process may end (Figure 15C, block 1540).

* * * * * *

An arrangement has been described which adequately satisfies current entertainment industry requirements for a low cost, mass-produceable digital video disk or other high capacity disc copy protection scheme but which also provides enhanced, extensible rights management capabilities for more advanced and/or secure platforms and for cooperative rights management between devices of lessor, greater, and/or differing rights resources. While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the invention.

We Claim:

1. An electronic appliance including:

a disk use arrangement for at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

a secure node coupled to the disk use arrangement, the secure node providing at least one rights management process.

2. An electronic appliance including:

a disk use arrangement for at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

at least one processing arrangement coupled to the disk use arrangement, the processing arrangement selecting at least some control information associated with information recorded on the storage medium based at least in part on the class of the appliance and/or the user of the appliance.

3. A system as in claim 2 wherein the processing arrangement selects a subset of control information used on another appliance and/or class of appliance.

4. A system as in claim 2 wherein the processing arrangement selects different control information from the information selected by another appliance and/or class of appliance.

5. A system as in claim 2 wherein at least some of the control information comprises an analog signal.

6. A system as in claim 2 wherein at least some of the control information comprises digitally encoded information.

7. In an appliance capable of using digital versatile disks, a method including the following steps:

at least one of (a) reading information from, and (b) writing information to, a digital versatile disk optical storage medium; and

selecting at least some control information associated with information recorded on the storage medium based at least in part on the class of the appliance and/or the user of the appliance.

8. A method as in claim 7 wherein the selecting step includes the step of selecting a subset of control information used on another appliance and/or class of appliance.

9. A method as in claim 7 wherein the selecting step includes the step of selecting, from control information stored on the storage medium, a different set of control information than the control information selected by another appliance and/or class of appliance.

10. An electronic appliance including:

a disk use arrangement for reading information from a digital versatile disk optical storage medium; and

at least one processing arrangement coupled to the disk use arrangement, the processing arrangement protecting information read from the storage medium.

11. An appliance as in claim 10 wherein the processing arrangement includes a rights management arrangement that applies at least one rights management technique to the read information.

12. An appliance as in claim 10 wherein the appliance further includes at least one port compliant at least in part with the IEEE 1394-1995 high speed serial bus standard, and the processing arrangement couples the protected information to the port.

13. In an electronic appliance, a method including the following steps:

reading information from a digital versatile disk optical storage medium; and

protecting the information read from the optical storage medium.

14. A method as in claim 13 wherein the protecting step includes the step of applying at least one rights management technique to the read information.

15. A method as in claim 13 further including the step of sending the protected information to an IEEE 1394 port.

16. An electronic appliance including:

a disk use arrangement for using information stored, or to be stored, on a digital versatile disk optical storage medium; and

at least one protecting arrangement coupled to the disk use arrangement and also coupled to receive at least one analog signal, the protecting arrangement creating protected digital information based at least in part on the analog signal.

17. In an electronic appliance, a method including the following steps:

receiving at least one analog signal; and

creating protected digital content based at least in part on the analog signal for storage on a digital versatile disk.

18. In an electronic appliance, a method including the following steps:

reading at least one analog signal from a digital versatile disk;

creating protected digital content based at least in part on the analog signal; and

outputting the protected digital content.

19. An electronic appliance including:

a disk use arrangement for using information stored, or to be stored, on a digital versatile disk optical storage medium; and

at least one rights management arrangement coupled to the disk use arrangement, the rights management arrangement treating the storage medium and/or information obtained from the storage medium differently depending on the geographical and/or jurisdictional context of the appliance.

20. In an electronic appliance, a method including the steps of:

reading information from at least one digital versatile disk; and

performing at least one rights management operation based at least in part on the geographical and/or jurisdictional context of the appliance.

21. An electronic appliance including:

a disk use arrangement for using at least one secure container stored on a digital versatile disk optical storage medium; and

at least one rights management arrangement coupled to the disk use arrangement, the rights management arrangement processing the secure container.

22. In an electronic appliance, a method including the following steps:

reading at least one secure container from at least one digital versatile disk; and

performing at least one rights management operation on the secure container.

23. An electronic appliance including:

at least one rights management arrangement for generating and/or modifying at least one secure container for storage onto a digital versatile disk optical storage medium.

24. In an electronic appliance, a method including the step of performing at least one rights management operation on at least one secure container for storage onto a digital versatile disk optical storage medium.

25. A digital versatile disk use system and/or method characterized in that the system and/or method uses at least one secure container.

26. A digital versatile disk use system and/or method characterized in that the system and/or method uses at least one

secure container of the type disclosed in PCT Publication No. WO 96/27155.

27. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium; and

a secure arrangement that securely manages information on the storage medium such that at least a first portion of the information may be used on at least a first class of appliance while at least a second portion of the information may be used on at least a second class of appliance

28. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium;

using at least a first portion of the information on at least a first class of appliance; and

using at least a second portion of the information on at least a second class of appliance.

29. A system including first and second classes of electronic appliances each including a secure processing arrangement, the first appliance class secure arrangement securely managing and/or using at least a first portion of the information, the second appliance class secure arrangement securely managing and/or using at least a second portion of the information.

30. A system as in claim 29 wherein the first and second information portions are different, and the second appliance class secure arrangement does not use the first information portion.

31. A system as in claim 29 wherein the first appliance class does not use the second information portion.

32. In a system including first and second classes of electronic appliances each including a secure arrangement, a method comprising:

(a) securely managing and/or using at least a first portion of the information with the first appliance class secure arrangement, and

(b) securely managing and/or using at least a second portion of the information with the second appliance class secure arrangement.

33. A method as in claim 32 wherein the first and second information portions are different, and step (b) does not use the first information portion.

34. A method as in claim 32 wherein step (a) does not use the second information portion.

35. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium; and

a secure arrangement that securely stores and/or transmits information associated with at least one of payment, auditing, controlling and/or otherwise managing content recorded on the storage medium.

36. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium; and

securely storing and/or transmitting information associated with at least one of payment, auditing, controlling and/or otherwise managing content recorded on the storage medium.

37. An electronic appliance including:

a disk use arrangement for writing information onto and/or reading information from a digital versatile disk optical storage medium;

a cryptographic engine coupled to the disk use arrangement, the engine using at least one cryptographic key; and

a secure arrangement that securely updates and/or replaces at least one cryptographic key used by the cryptographic engine to at least in part modify the scope of information usable by the appliance.

38. A method of operating an electronic appliance including:

writing information onto and/or reading information from a digital versatile disk optical storage medium;

using at least one cryptographic key in conjunction with said information; and

securely updating and/or replacing at least one cryptographic key used by the cryptographic engine to at least in part modify the scope of information useable by the appliance.

39. A digital versatile disk appliance characterized in that at least one cryptographic key used by the appliance is securely updated and/or replaced to at least in part modify the scope of information that can be used by the appliance.

40. An appliance as in claim 39 further characterized in that the key updating and/or replacing is based on class of appliance.

41. An electronic appliance having a class associated therewith, characterized in that at least one cryptographic key set used by the appliance class is selected to help ensure security of information released from at least one digital versatile disk.

42. A digital camera for generating at least one image to be written onto a digital versatile disk optical storage medium, characterized in that the camera includes at least one information protecting arrangement that at least in part protects the image so that the information is persistently protected through subsequent processes such as editing, production, writing onto a digital versatile disk, and/or reading from a digital versatile disk.

43. A digital camera for generating image information that can be written onto a digital versatile disk optical storage medium, a method comprising:

capturing at least one image with a digital camera; and

protecting information provided by the digital camera so that the information is selectively persistently protected through subsequent processes such as distribution, editing and/or production, writing onto the digital versatile disk optical storage medium, and/or reading from the digital versatile disk optical storage medium.

44. In an electronic appliance including a disk use arrangement, a method comprising:

reading information from at least one digital versatile disk optical storage medium; and

persistently protecting at least some of the read information through at least one subsequent editing and/or production process.

45. In an electronic appliance, a method including the following steps:

reading information from and/or writing information to at least one digital versatile disk optical storage medium; and

securely managing information on the storage medium, including the step of using at least a first portion of the information on at least a first class of appliance, and using at least a second portion of the information on at least a second class of appliance.

46. A method of providing copy protection and/or use rights management of at least one digital property content and/or secure container to be stored and/or distributed on a digital versatile disk medium, comprising the step(s) of:

providing a set of use control(s) within a cryptographically encapsulated data structure having a predetermined format, the data structure format defining at least one secure software container for providing use rights information for digital property content to be stored on the digital versatile disk medium.

47. A method as in claim 46 further including the step of using at least one digital property content stored on an optical disk in accordance with the use controls, including the step of using a prescribed secure cryptographic key or set of cryptographic keys for using rights information.

48. A method as in claim 46 further including the step of decrypting control rules and/or other selected encrypted

information content encapsulated in the software container using at least one set of cryptographic keys.

49. A method as in claim 46 further including the step of applying decrypted control rules to regulate use in accordance with control information contained within said control rules, so as to facilitate management of a diverse set of use and distribution rights which may be specific to different users and/or optical disk appliances.

50. A method of providing rights management of digital property stored on digital versatile disk according to claim 46 wherein said secure container data structure comprises:

one or more content objects comprising digital property content; and

one or more control objects comprising a set of control rules defining copy protection, use and distribution rights to digital property content stored on the optical disk.

51. A method of providing rights management of digital property stored on a digital versatile disk according to claim 46, wherein a content object further comprises one or more reference pointers to digital property content stored elsewhere on the digital versatile disk.

52. A method of providing rights management of digital property stored on a digital versatile disk according to claim 46, wherein a control object further comprises one or more reference pointers to control information stored elsewhere on the digital versatile disk.

53. A method of providing rights management of digital property stored on digital versatile disk according to claim 46, wherein digital information stored on said optical disk includes one or more metadata blocks comprising further information used in conjunction with the control rules to use digital property content stored elsewhere on the optical disk.

54. A method of providing rights management of digital property stored on digital versatile disk according to claim 46, wherein a metablock may be either of a protected type or of an unprotected type.

55. An arrangement for implementing a rights management system for controlling copy protection, use and/or distribution rights to multi-media digital property content stored or otherwise contained on a digital versatile disk, comprising:

an encrypted data structure defining a secure information container stored on an optical disk medium, the encrypted data structure including and/or referencing at least one content object and at least one control object associated with the content object, said content object comprising digital property content and said control object comprising rules defining use rights to the digital property content.

56. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a content object further comprises one or more reference pointers to digital property content stored elsewhere on the digital versatile disk.

57. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises one or more reference pointers to control information stored elsewhere on the digital versatile disk.

58. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein an control object further comprises information for controlling various operations of an optical disk appliance or computer.

59. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises information for controlling various operations of an optical disk appliance or computer.

60. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises a rule specifying decoding and/or enforcement of CGMA encoded copy protection rules associated with the digital content property.

61. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a control object further comprises a rule specifying at least one content scrambling system compatible encoding/decoding of digital property content.

62. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein said optical disk contains a block of stored information comprising encrypted keys used for decryption of said encrypted data structure.

63. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein said optical disk contains a block of stored information comprising hidden keys used for decryption of said encrypted keys.

64. An arrangement for implementing a rights management system for digital versatile disks according to claim 55, wherein a content object further comprises one or more reference pointers to digital property content stored on a separate storage medium.

65. A rights management system for providing copy protection, use and/or distribution rights management for multi-media digital property content stored or otherwise contained on a digital versatile disk for access by an optical disk player device that uses digital property content stored on said optical disk medium, wherein said appliance includes a microprocessor controller for decrypting and using control rules and other selected encrypted information content encapsulated in the secure container by using a prescribed cryptographic key and applying said decrypted control rules to regulate use in accordance with control information contained within said control rules, so as to facilitate management of a diverse set of use and/or distribution rights which may be specific to different users and/or optical disk appliances, the system including:

an optical disk medium having stored thereon an encrypted data structure defining a secure information container, the encrypted data structure comprising and/or referencing at least one content object and at least one control object, said content object comprising digital property content, said control object

comprising rules defining use rights associated with the digital property.

66. A method for providing copy protection, use and distribution rights management of multi-media digital property stored on and/or distributed via digital versatile disk, said optical disk medium having stored thereon an encrypted data structure defining a secure container for housing rights and/or copy protection information pertaining to digital property content stored on the optical disk, wherein an optical disk player appliance for using digital property content stored on an optical disk must utilize a prescribed secure cryptographic key or set of keys to use the secure container, said data structure comprising one or more content objects comprising digital property content and one or more control objects comprising a set of rules defining use rights to digital property, comprising the steps of:

(a) decrypting control rules and other selected encrypted information content encapsulated in the secure container using one or more cryptographic keys; and

(b) applying decrypted control rules to regulate use and/or distribution of digital property content stored on the optical disk in accordance with control information contained within the control rules, so as to provide customized use and/or distribution rights that are specific to different optical disk user platforms and/or optical disk appliances.

67. A rights management system for providing copy protection, use and/or distribution rights management of digital property stored or otherwise contained on a digital versatile disk, comprising:

a secure container means provided on an optical disk medium for cryptographically encapsulating digital property content stored on the optical disk, said container means comprising a content object means for containing digital property content and a control object means for containing control rules for regulating use and/or distribution of said digital property content stored on the optical disk.

68. The rights management system of claim 67 wherein an optical disk player appliance for using information stored on an optical disk comprises a secure node means for using said secure container means provided on an optical disk and implementing said control rules to control operation of said player appliance to regulate use of said digital property content.

69. In a system including plural electronic appliances at least temporarily connected to one another, a rights authority broker that determines what appliances are connected and specifies at least one rights management context depending on said determination.

70. An electronic appliance at least temporarily connected to a rights authority broker, the electronic appliance receiving at least one rights context from the rights authority broker when the device is connected to the rights authority broker.

71. A first electronic appliance at least temporarily connected to a second electronic appliance, the first

electronic appliance selecting between at least first and second rights management contexts depending at least in part on whether the first appliance is connected to the second electronic appliance.

72. In a system including first and second electronic appliances that may be selectively coupled to communicate with one another, an arrangement for defining at least one different rights management control based at least in part on whether the first and second electronic appliances are connected.

73. A method of defining at least one rights management context comprising:

(a) determining whether a first electronic appliance is present; and

(b) defining at least one rights management control set based at least in part on the determining step (a).

74. A method of defining at least one rights management context including:

(a) coupling an optical disk storing information to an electronic appliance that can be selectively connected to a rights management broker;

(b) determining whether the electronic appliance is currently coupled to a rights management broker; and

(c) conditioning at least one aspect of use of at least some of the information stored on the optical disk based on whether the electronic appliance is coupled to the rights management broker.

75. A method as in claim 74 wherein step (c) includes the step of obtaining at least one rights management context from the rights management broker.

76. A method as in claim 74 wherein step (c) includes the step of obtaining at least one combined control set from the rights management broker.

77. A method of defining at least one rights management context including:

(a) coupling an optical disk storing information to an electronic appliance;

(b) using at least some of the information stored on the optical disk based on a first rights management context;

(c) coupling the electronic appliance to a rights management broker; and

(d) concurrently with step (c), using at least some of the information stored on the optical disk based on a second rights management context different from the first rights management context

78. An electronic appliance include a secure node and an optical disk reader, the electronic appliance applying different rights management contexts to protected information stored on an optical disk coupled to the optical disk reader depending at least in part on whether the electronic appliance is coupled to at least one additional secure node.

79. An electronic appliance including:

an optical disk reading and/or writing arrangement;

a secure node coupled to the optical disk reading and/or writing arrangement, the secure node performing at least one rights management related function with respect to at least some information read by the optical disk reading and/or writing arrangement; and

at least one serial bus port coupled to the secure node, the serial bus port for providing any or all of the functions, structures, protocols and/or methods of IEEE 1394-1995.

80. A digital versatile disk appliance including:

means for watermarking content; and

serial bus means for communicating the watermarked content,

wherein the serial bus means complies with IEEE 1394-1995.

81. An optical disk reading and/or writing device including:

at least one secure node capable of watermarking content and/or processing watermarked content; and

an IEEE 1394-1995 serial bus port.

82. An optical disk using device comprising:

a secure processing unit; and

an IEEE 1394-1995 serial bus port.

83. A device as in claim 82 wherein the secure processing unit includes a channel manager.

84. A device as in claim 82 wherein the secure processing unit executes a rights operating system in whole or in part.

85. A device as in claim 82 wherein the secure processing unit includes a tamper-resistant barrier.

86. A device as in claim 82 wherein the secure processing unit includes an encryption/decryption engine.

87. A rights cooperation method comprising:

(a) connecting an appliance to at least one further appliance;

(b) determining whether the first and/or further appliance and/or user(s) of said first and/or further appliance have appropriate rights and/or resources for performing an action; and

(c) selectively performing the action based at least in part on the determination.

88. A rights cooperation method comprising:

(a) connecting an appliance to at least one further appliance;

(b) determining whether the first and/or further appliance and/or user(s) of said first and/or further appliance have appropriate security for performing an action; and

(c) cooperating between the first and further appliance to selectively perform the action.

89. A cooperative rights management arrangement comprising:

a communications arrangement that allows at least first and second appliances to communicate; and

an arrangement that processes at least one event based at least in part on assessing and/or pooling rights and/or resources between the first and second appliances.

90. An optical disk using system and/or method including at least some of the elements shown in Figure 1A.

91. An optical disk using system and/or method including at least some of the elements shown in Figure 1B.

92. An optical disk using system and/or method including at least some of the elements shown in Figure 1C.

93. An optical disk using system and/or method including at least some of the elements shown in Figure 2A.

94. An optical disk using system and/or method including at least some of the elements shown in Figure 2B.

95. An optical disk using system and/or method including at least some of the elements shown in Figure 3.

96. An optical disk using system and/or method using at least some of the elements shown in Figure 3A.

97. An optical disk using system and/or method using at least some of the control set elements shown in Figure 3B.

98. An optical disk using system and/or method using at least some of the elements shown in Figure 4A.

99. An optical disk using system and/or method using at least some of the elements shown in Figure 4B.

100. An optical disk using system and/or method using at least some of the elements shown in Figure 5.

101. An optical disk using system and/or method using at least some of the elements shown in Figure 6.

102. An optical disk using system and/or method using at least some of the elements shown in Figure 7.

103. An optical disk using system and/or method using at least some of the elements shown in Figure 8.

104. An optical disk using system and/or method using at least some of the elements shown in Figure 9.

105. An optical disk using system and/or method using at least some of the elements shown in Figure 10.

106. An optical disk using system and/or method using at least some of the elements shown in Figure 11.

107. An optical disk using system and/or method including at least some of the elements shown in Figure 12.

108. An optical disk using system and/or method including at least some of the elements shown in Figure 13.

109. An optical disk using system and/or method including at least some of the elements shown in Figure 14.

110. A system and/or method including some or all of the elements shown in Figures 15A-15C.

111. A system and/or method as in any one of the preceding claims, further including, in combination, any element described in any one of the following prior patent specifications:

PCT Publication No. WO 96/27155;

European Patent No. EP 329681;

PCT Application No. PCT/US96/14262;

U.S. Patent Application Serial No. 08/689,606; and/or

U.S. Patent Application Serial No. 08/689,754.

112. A system or process as in any of the preceding claims wherein the phrase "high capacity optical disk" is substituted for "digital versatile disk."

113. A method of clearing or otherwise processing information resulting at least in part from one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

114. A system and/or method for defining rules for use in one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

115. A system and/or method for defining rules and associated content for use in one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

116. A system and/or method for producing an optical disk for use with one or more digital versatile disk appliances and/or methods as defined in any of the preceding claims.

117. A system and/or method for clearing audit information from one or more appliances and/or methods as defined in any of the preceding claims.

118. In an network including at least one electronic appliance that reads information from and/or writes information to at least one digital versatile disk optical storage medium, and securely communicates information associated with at least one of

payment, auditing, usage, access, controlling and/or otherwise managing content recorded on the storage medium, a method of processing said communicated information including the step of generating at least one payment request and/or order based at least in part on the information.

119. A method of defining at least one control set for storage on a high capacity optical disk that can storage images, audio, text and/or other information, the high capacity optical disk for use by any of plural different electronic appliance types, the method including the step of specifying at least one control that provides different conditions and/or consequences depending upon at least one of the following:

electronic appliance class;

electronic appliance security;

electronic appliance user class;

electronic appliance connectivity;

electronic appliance resources;

electronic appliance access to resources; and

rights management cooperation between plural electronic appliances.

Fig. 1B

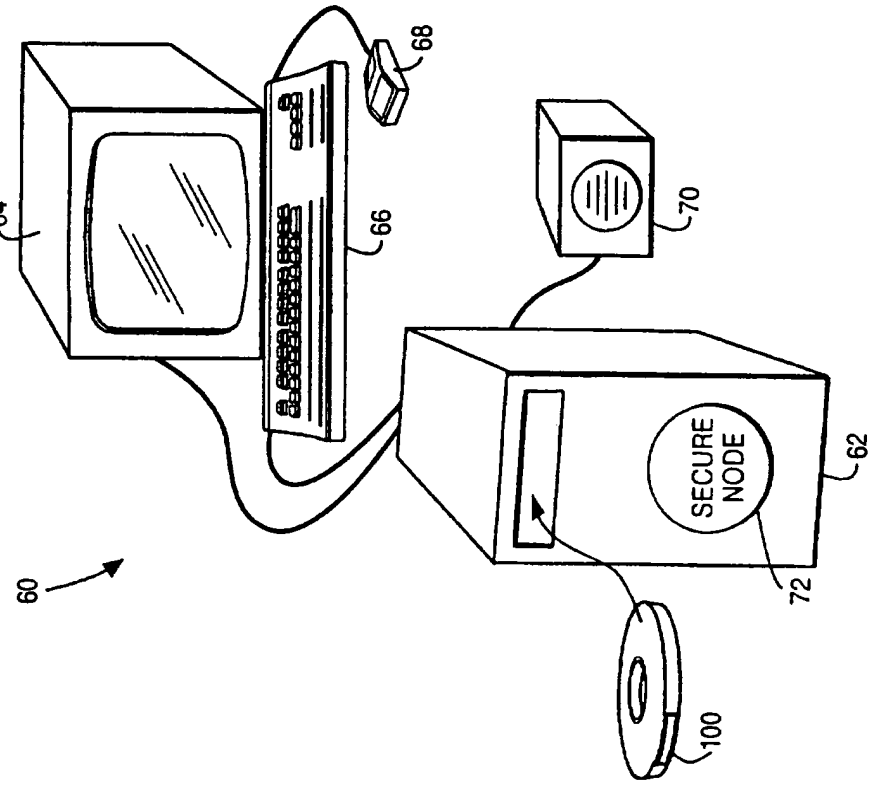
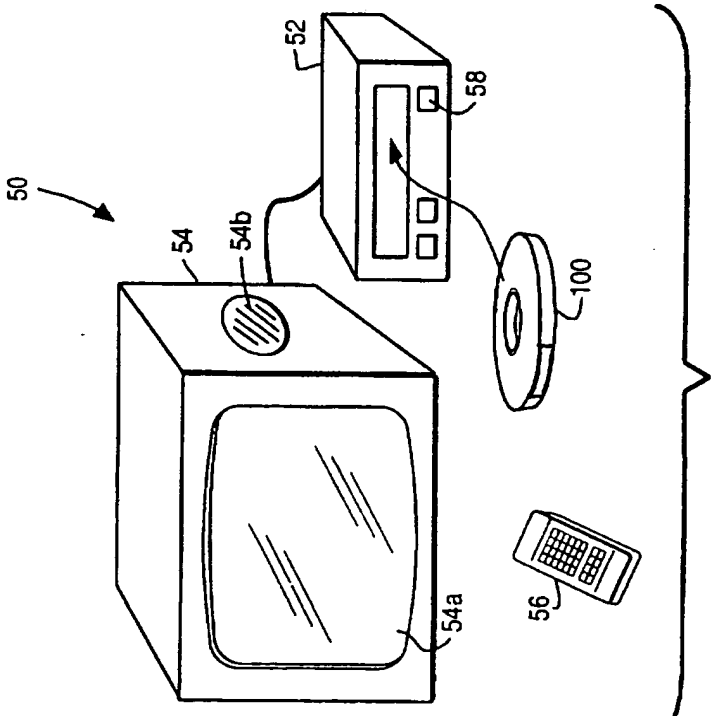


Fig. 1A



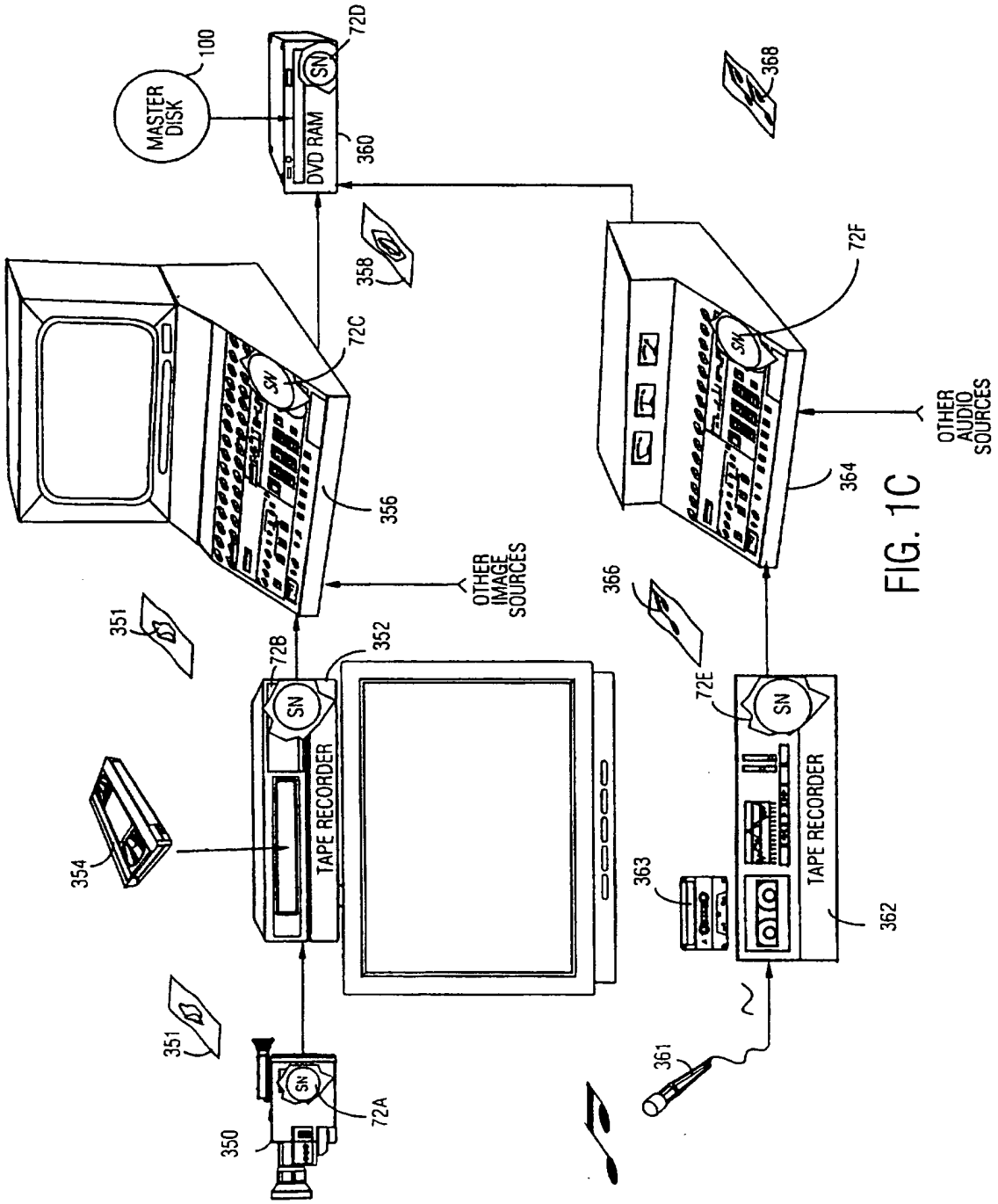


FIG. 1C

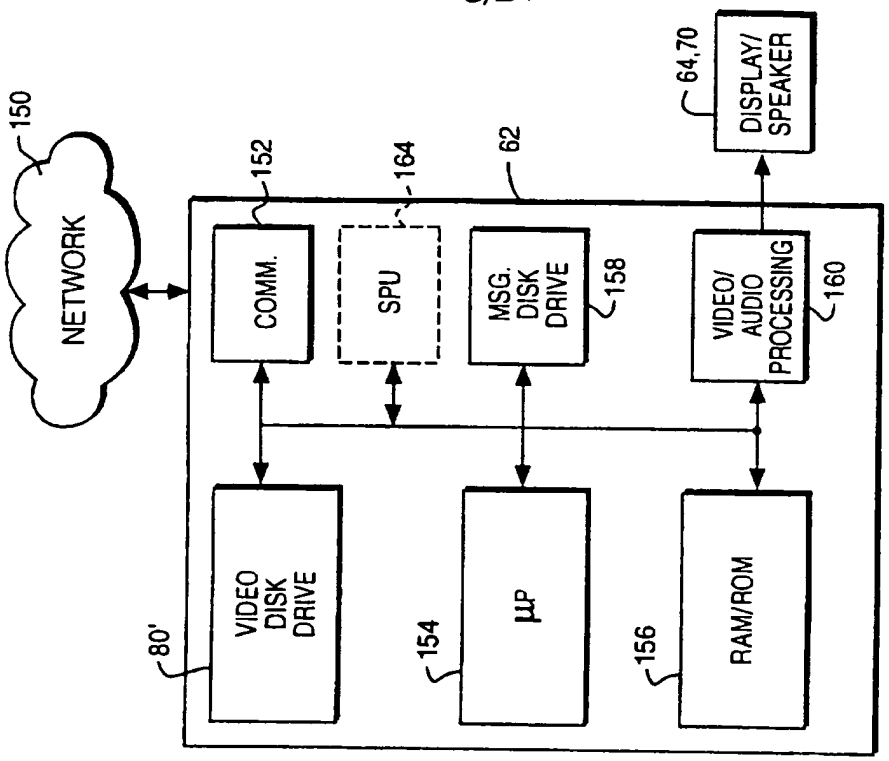


Fig.2B

EXAMPLE SECURE NODE ARCHITECTURE

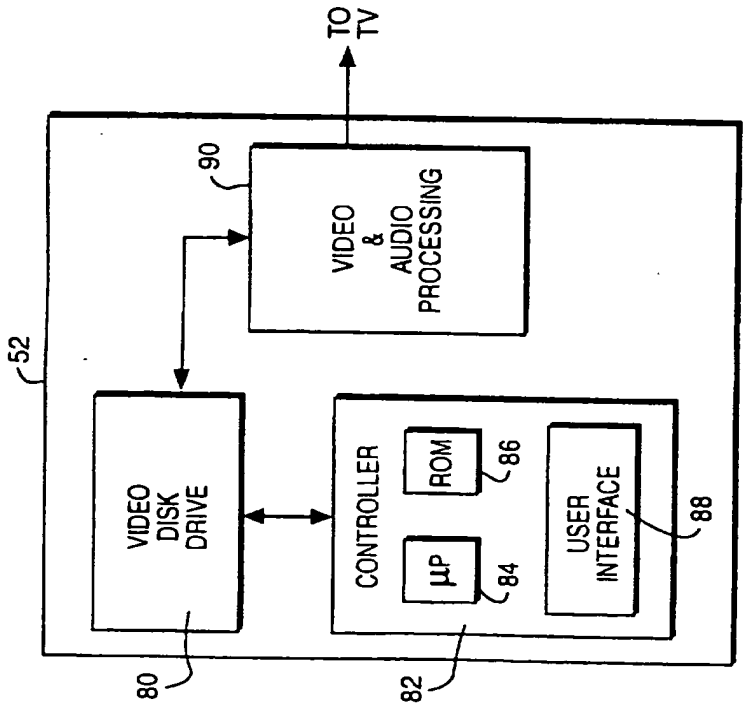


Fig.2A

EXAMPLE PLAYER ARCHITECTURE

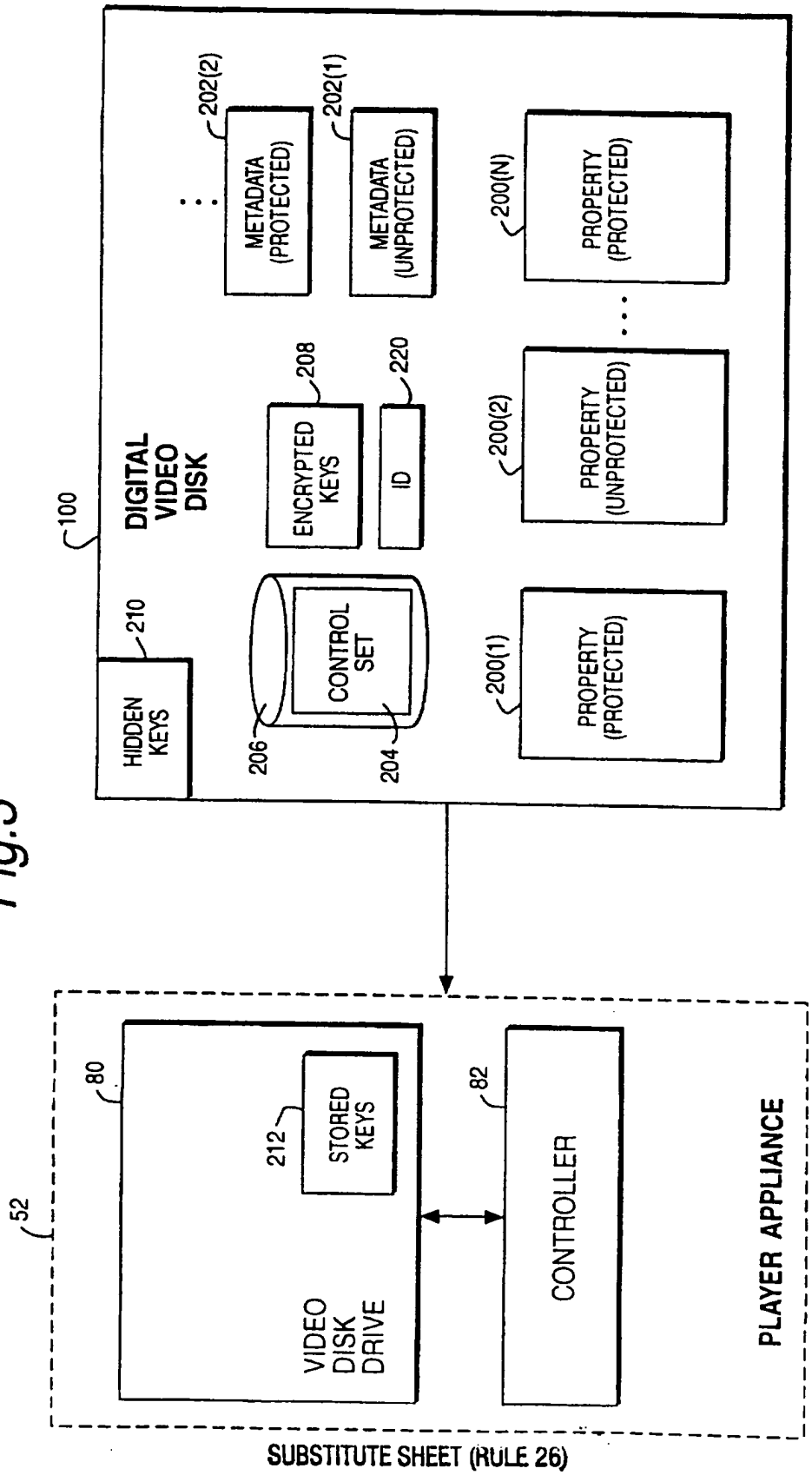


Fig.3

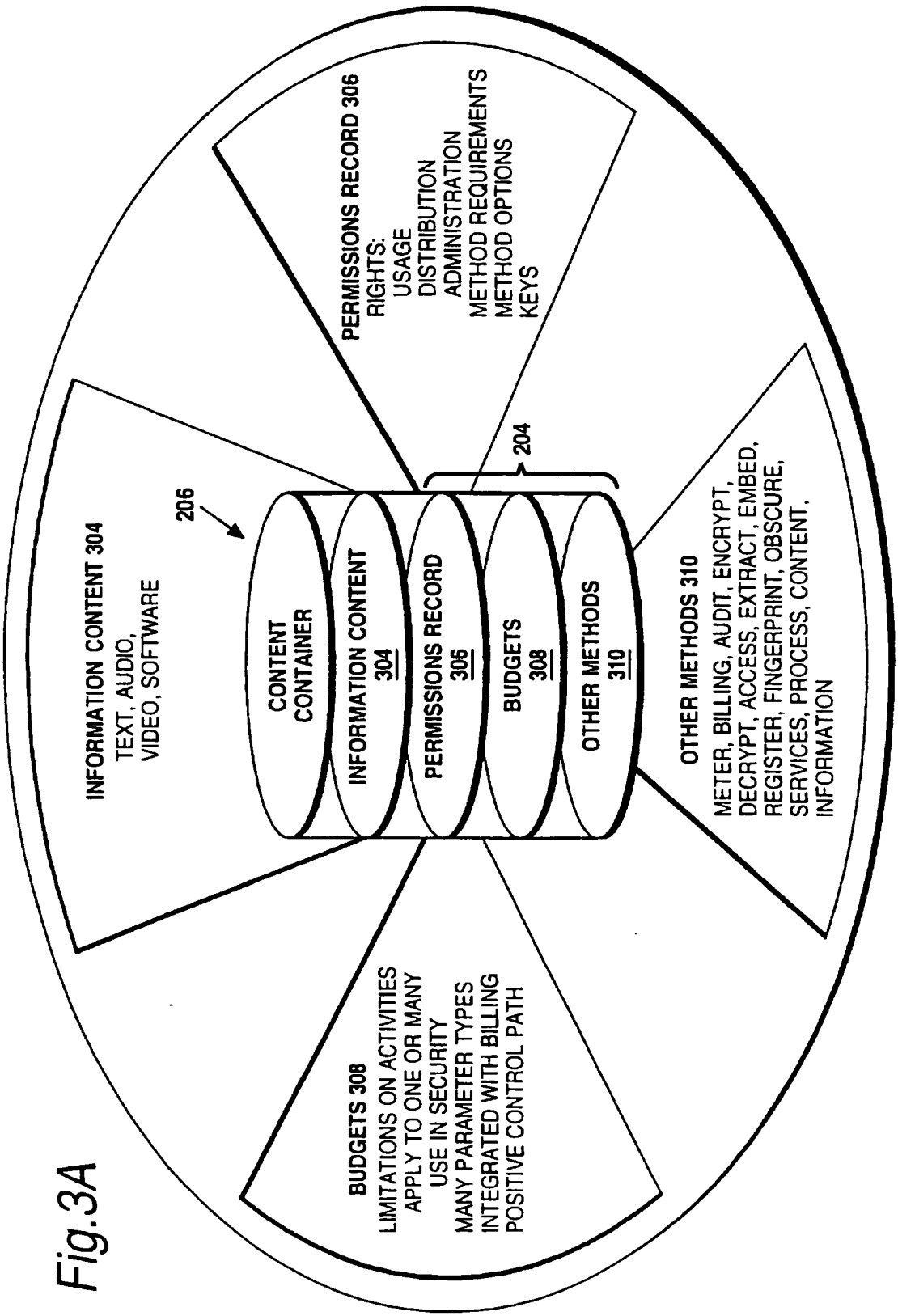


Fig.3A

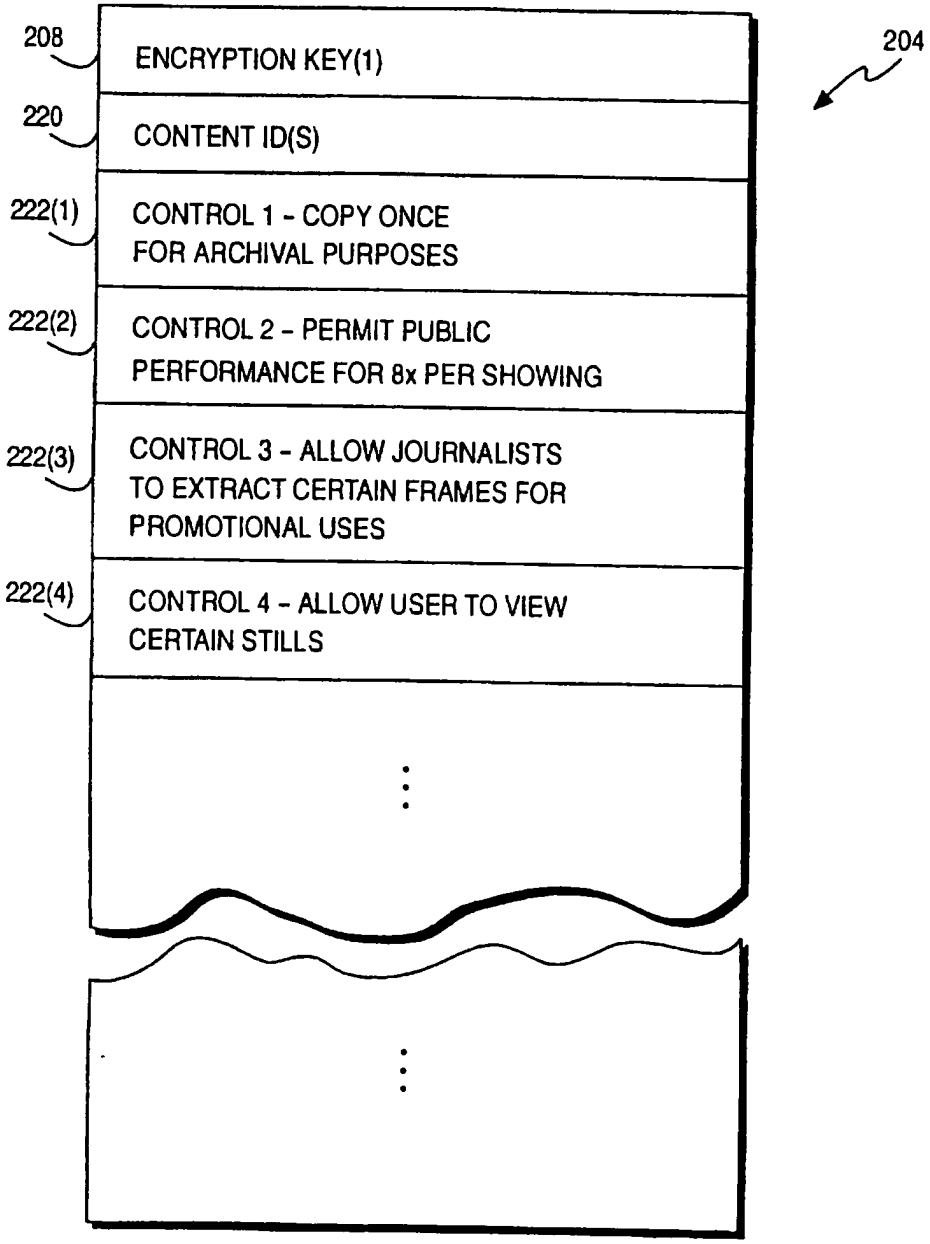


Fig.3B

EXAMPLE CONTROL SET

Fig.4A

EXAMPLE PLAYER STEPS

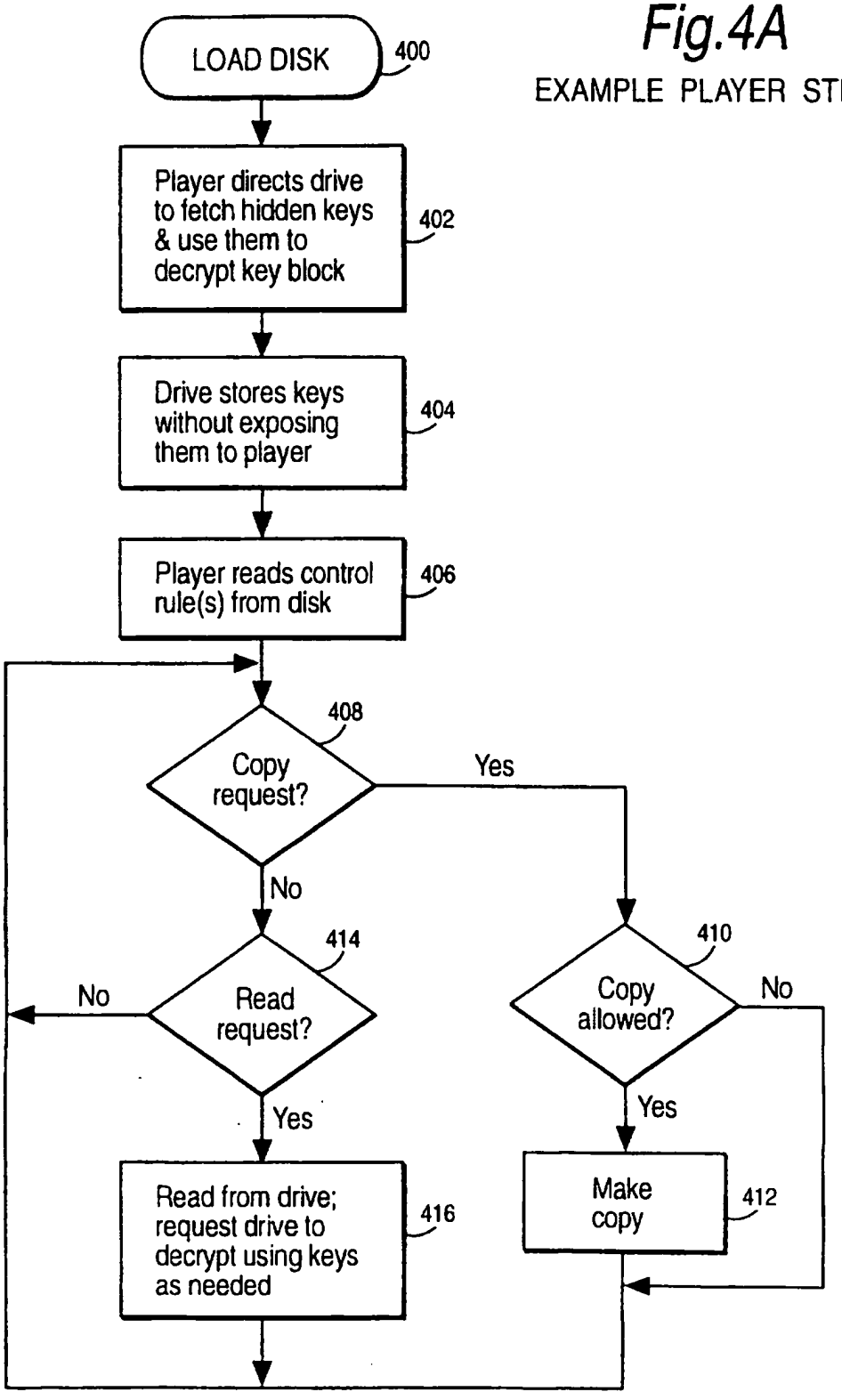
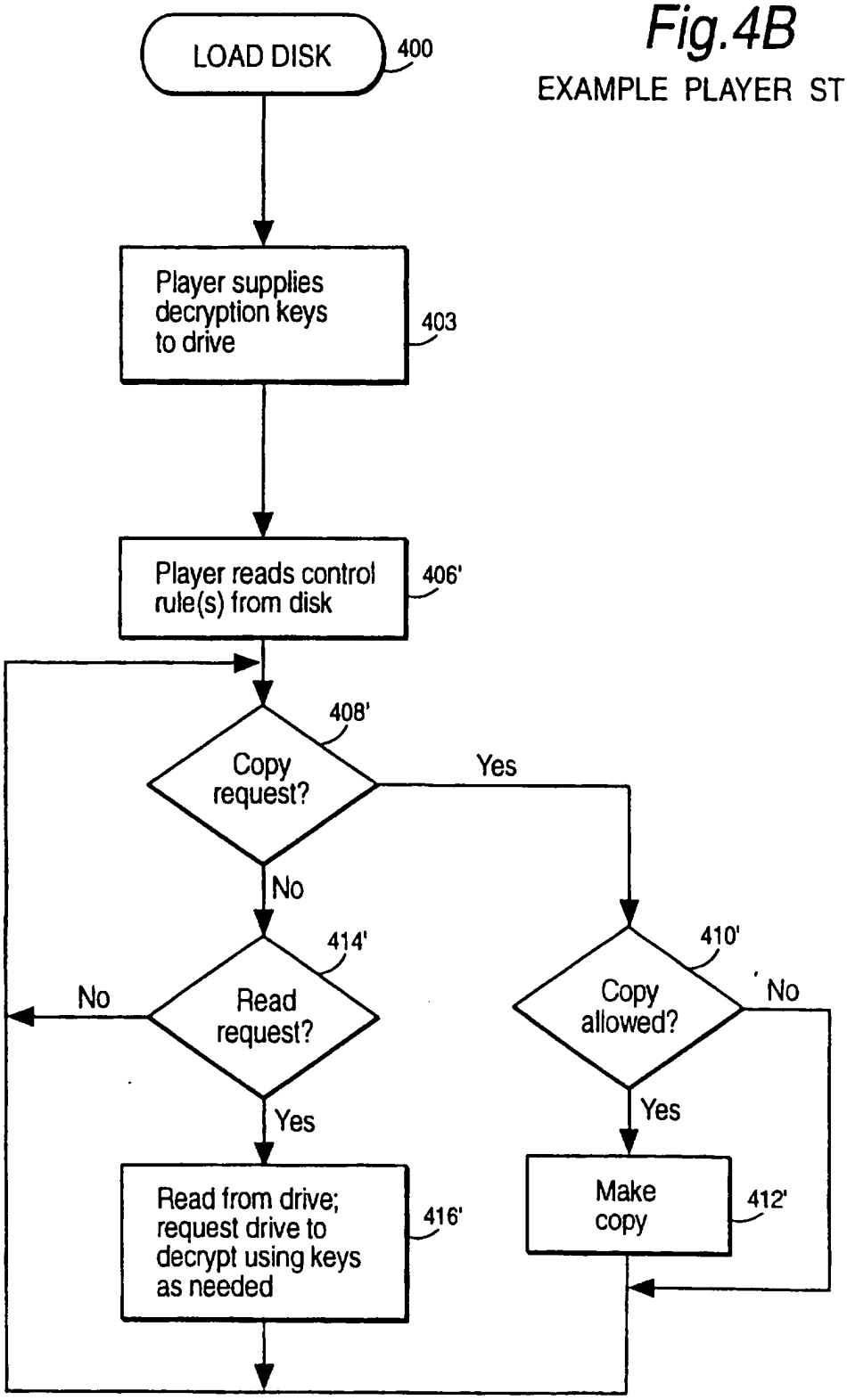


Fig.4B

EXAMPLE PLAYER STEPS



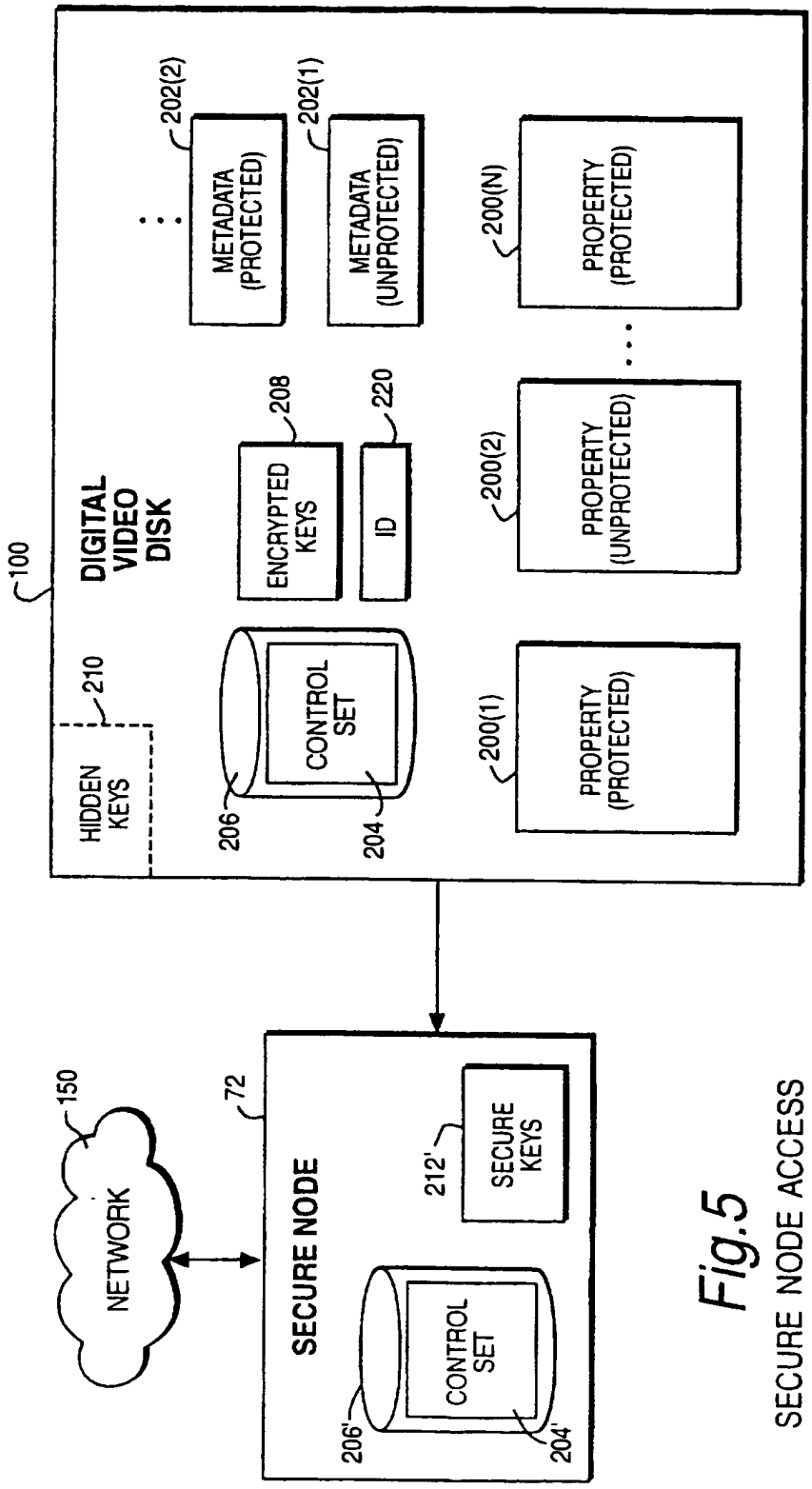


Fig.5

SECURE NODE ACCESS

Fig.6

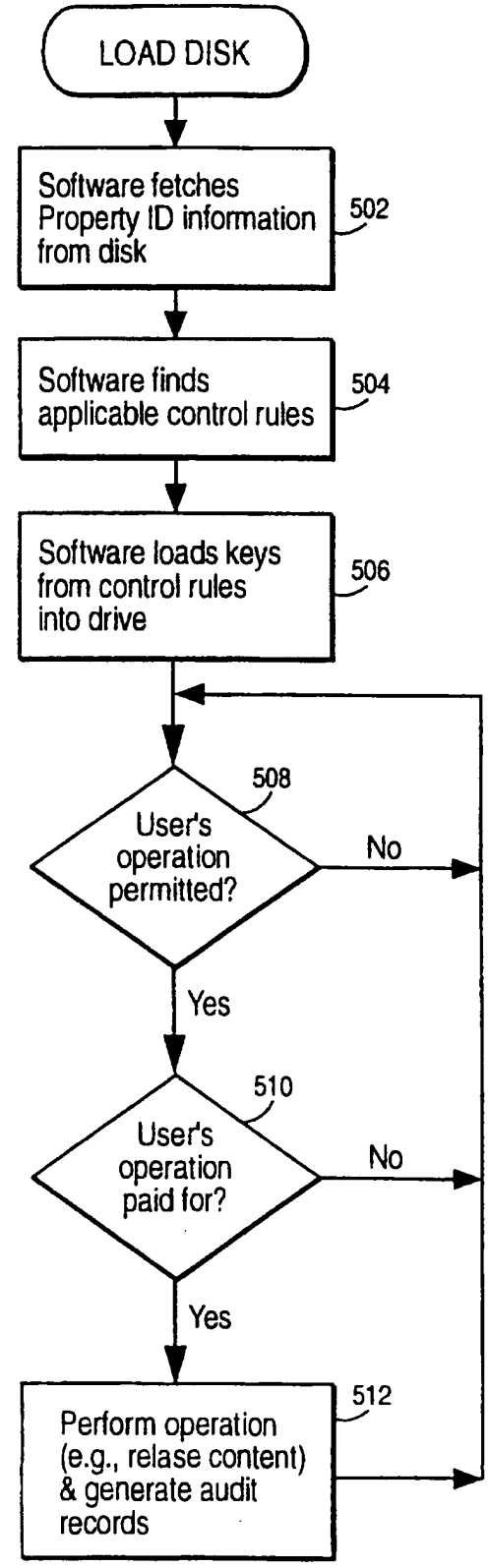


FIG. 7

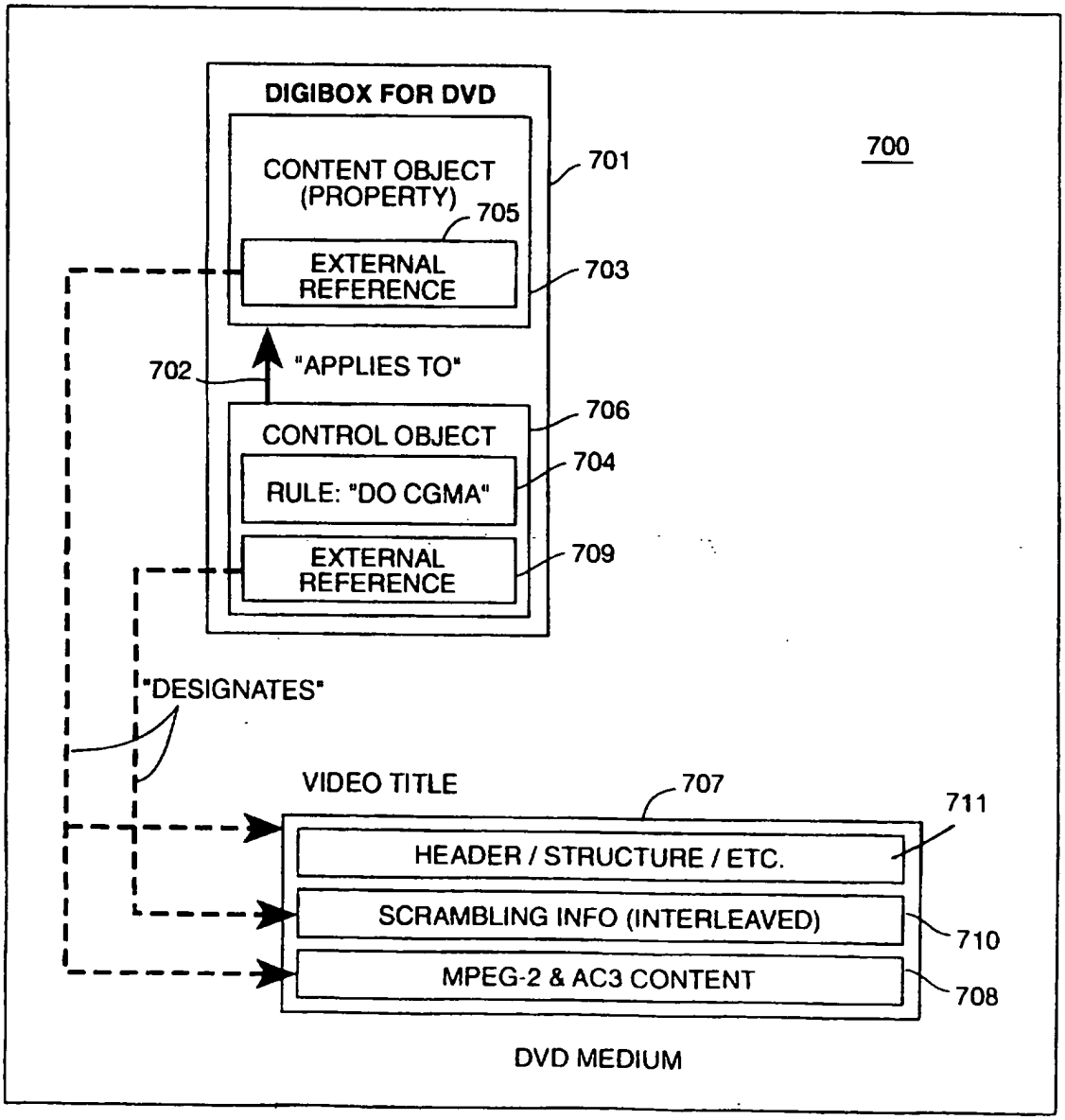


FIG. 8

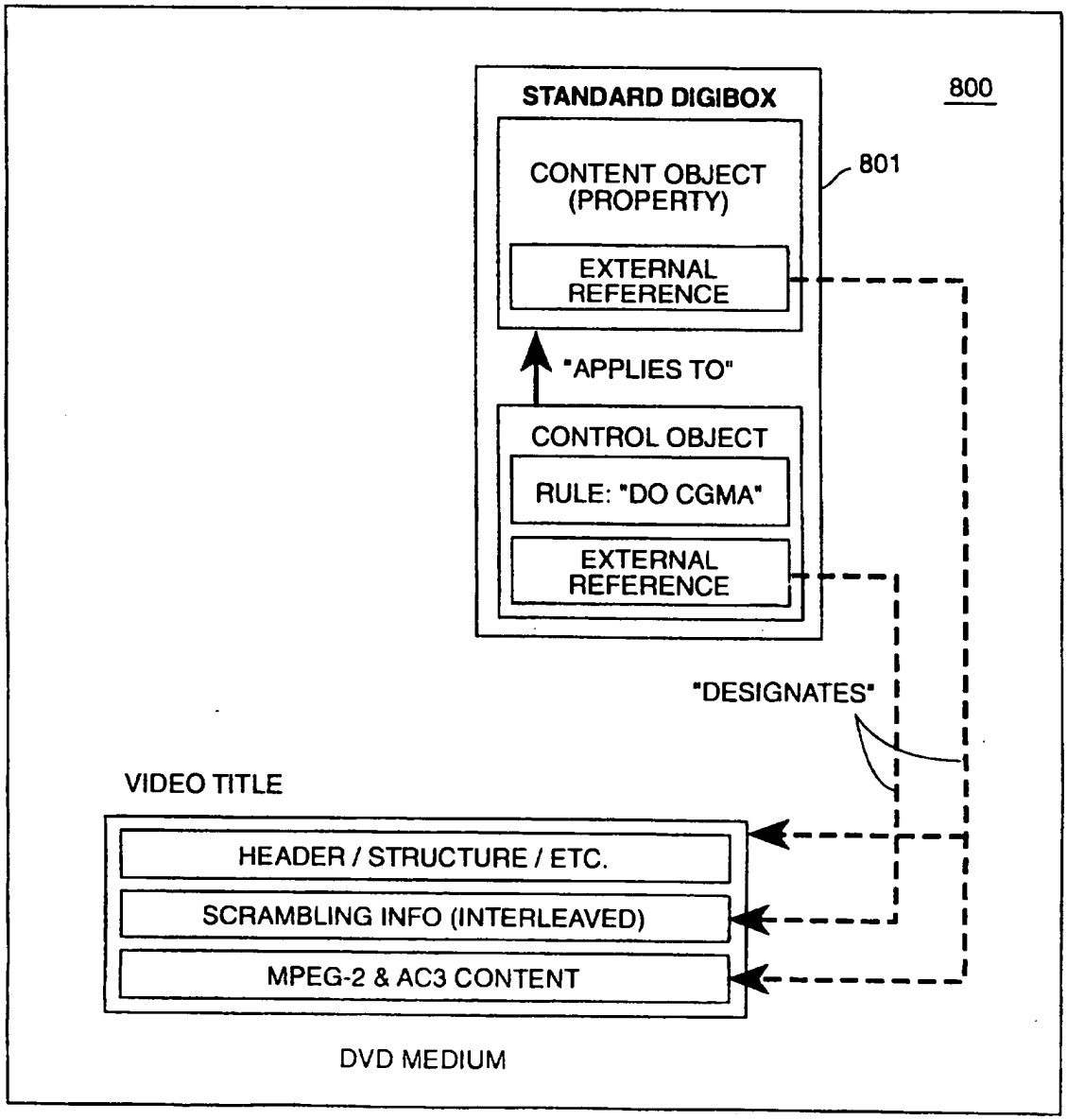


FIG. 9

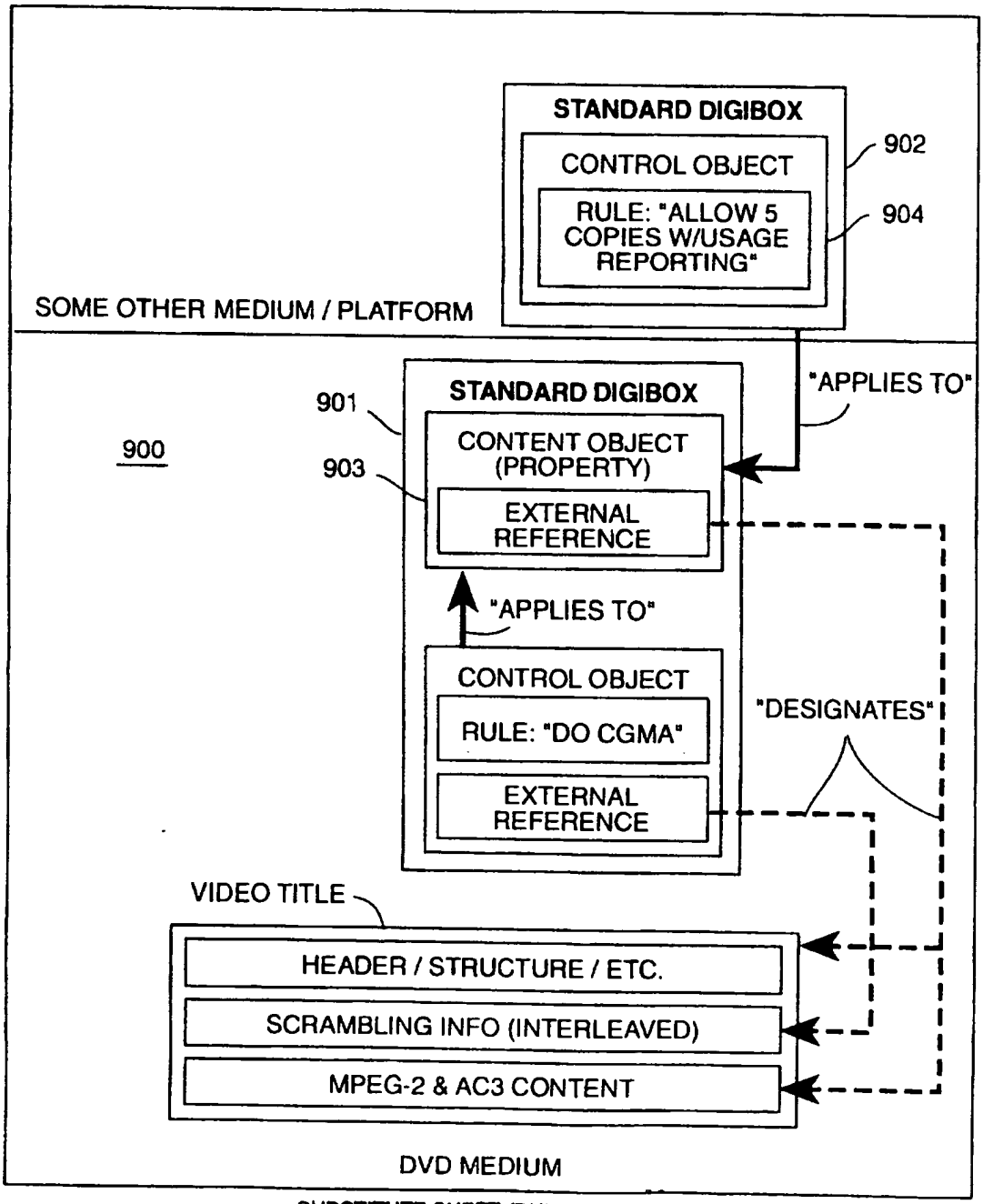


FIG. 10

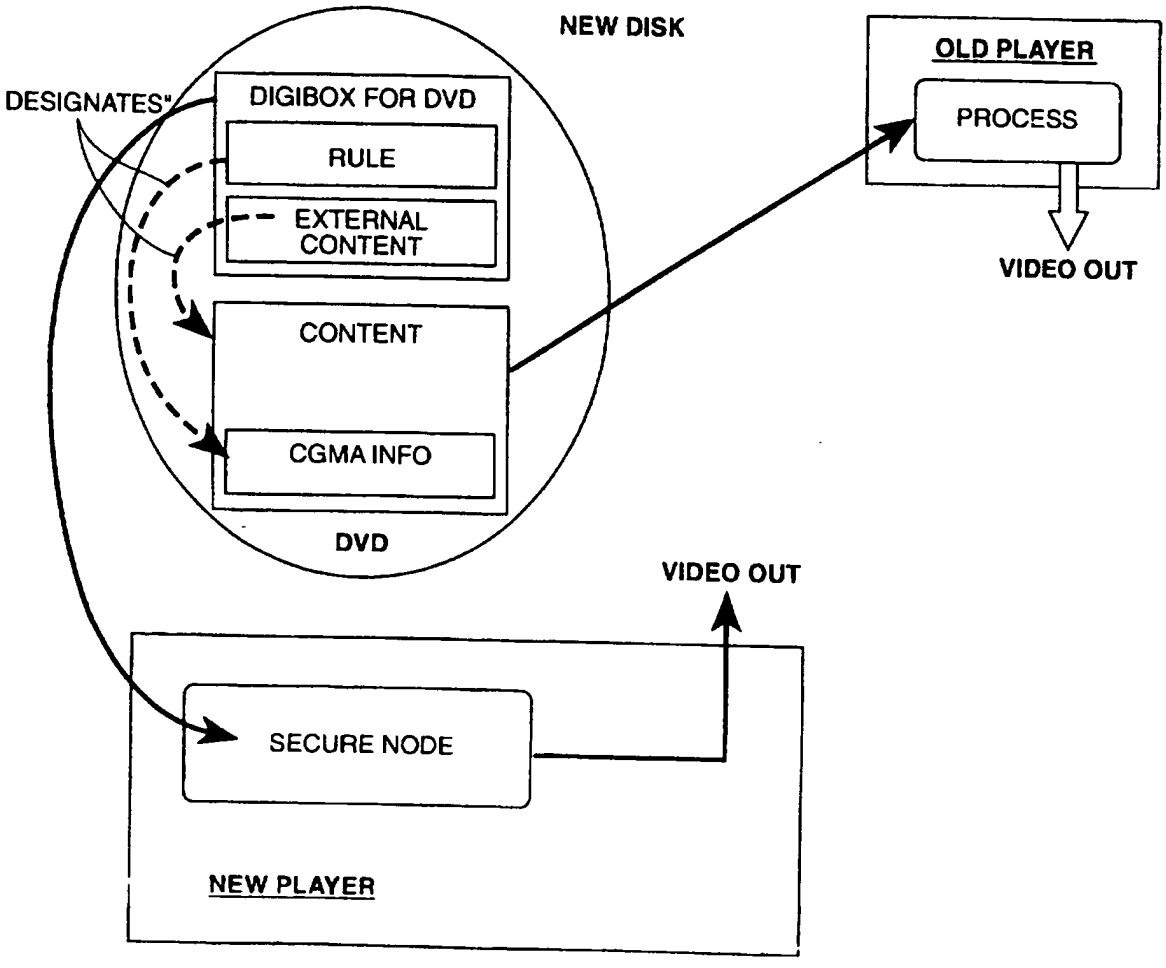
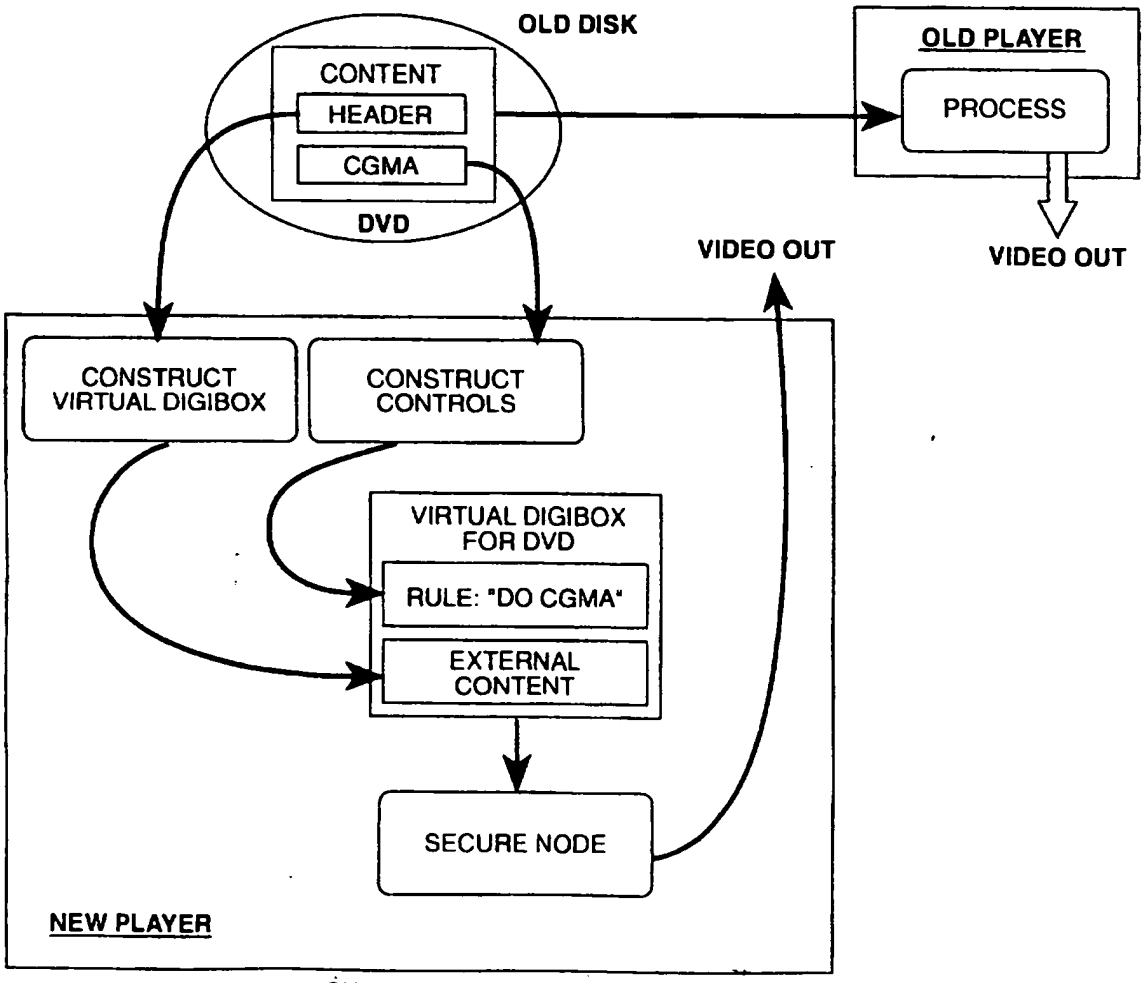


FIG. 11



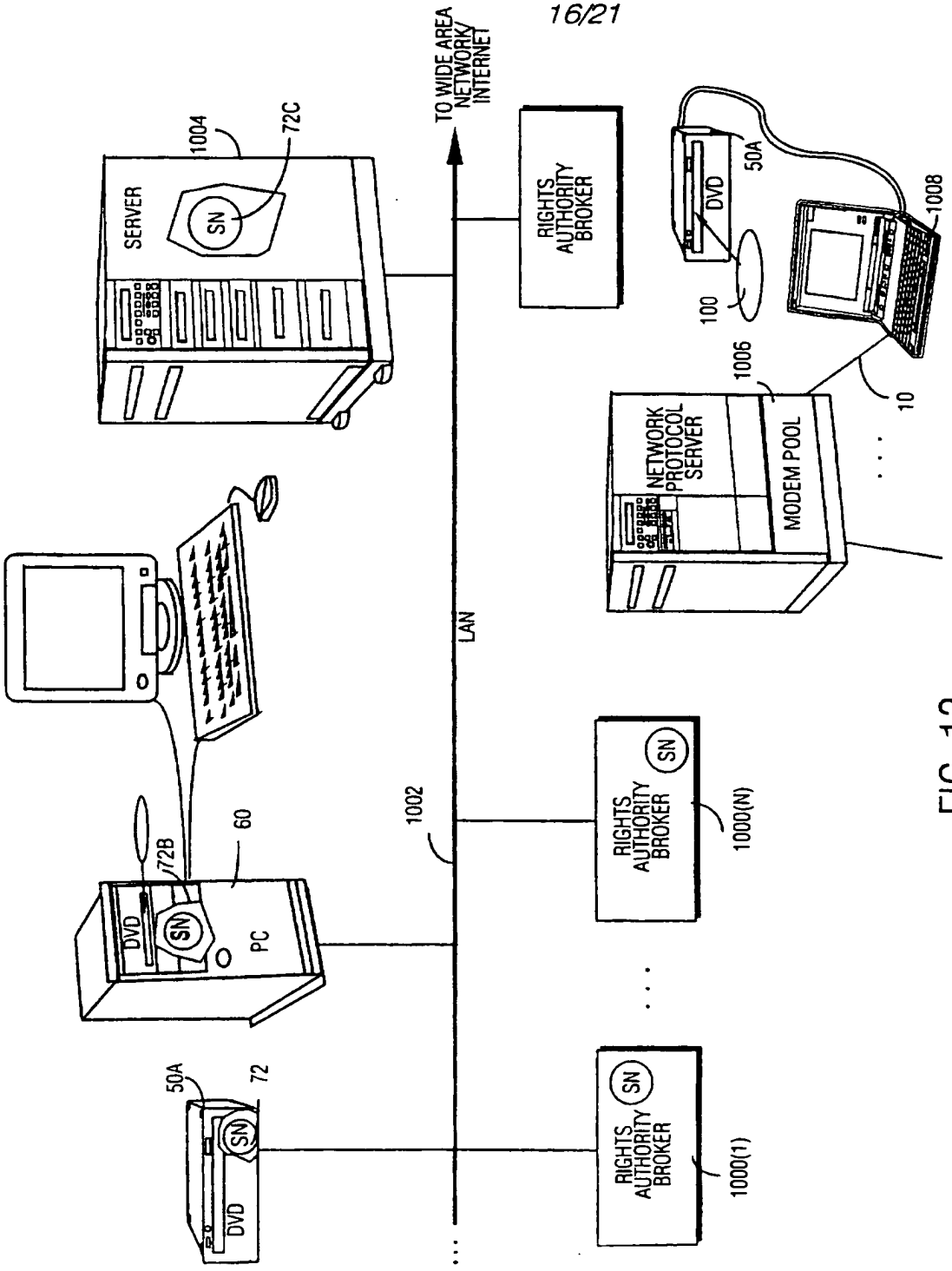


FIG. 12

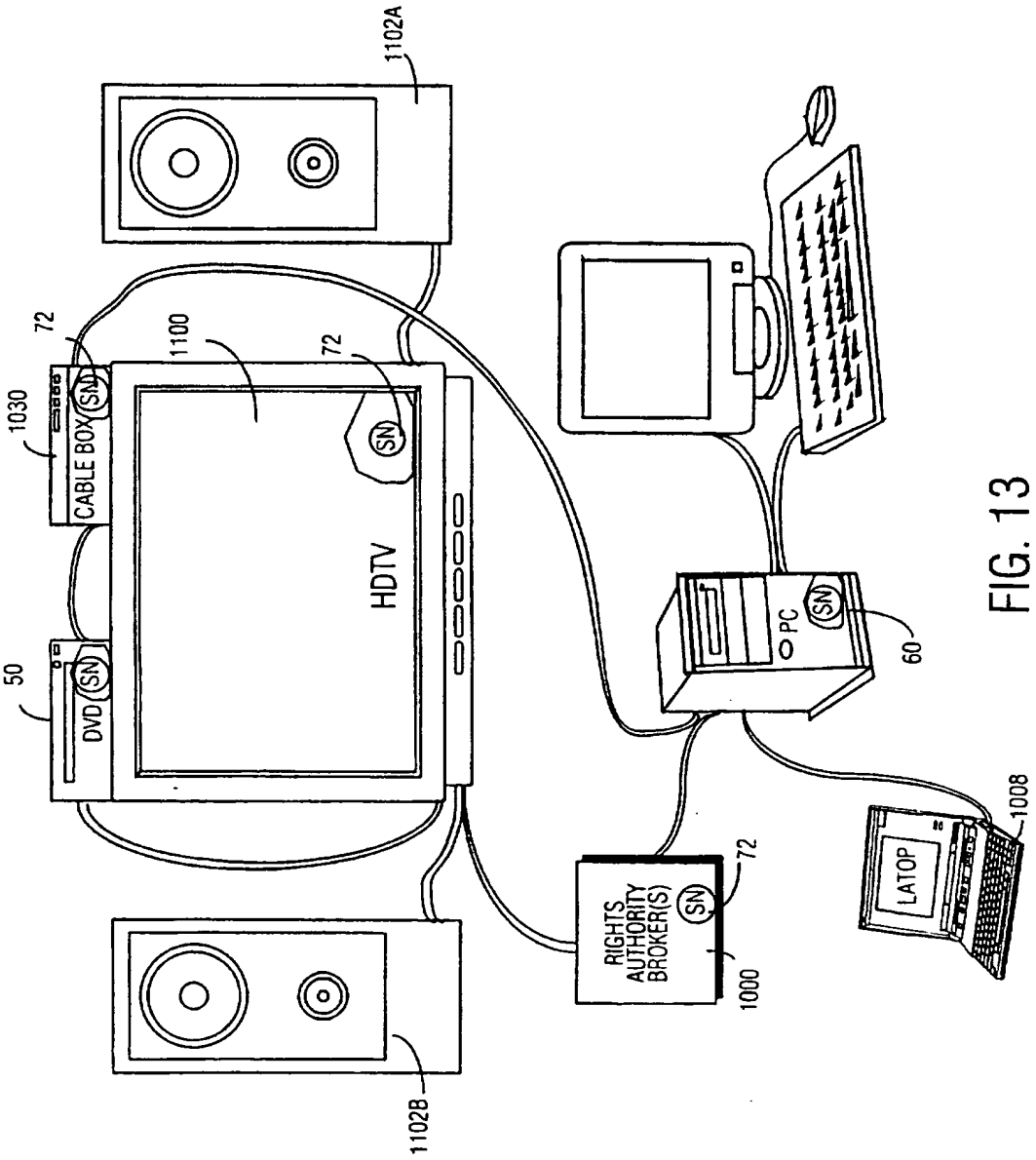


FIG. 13

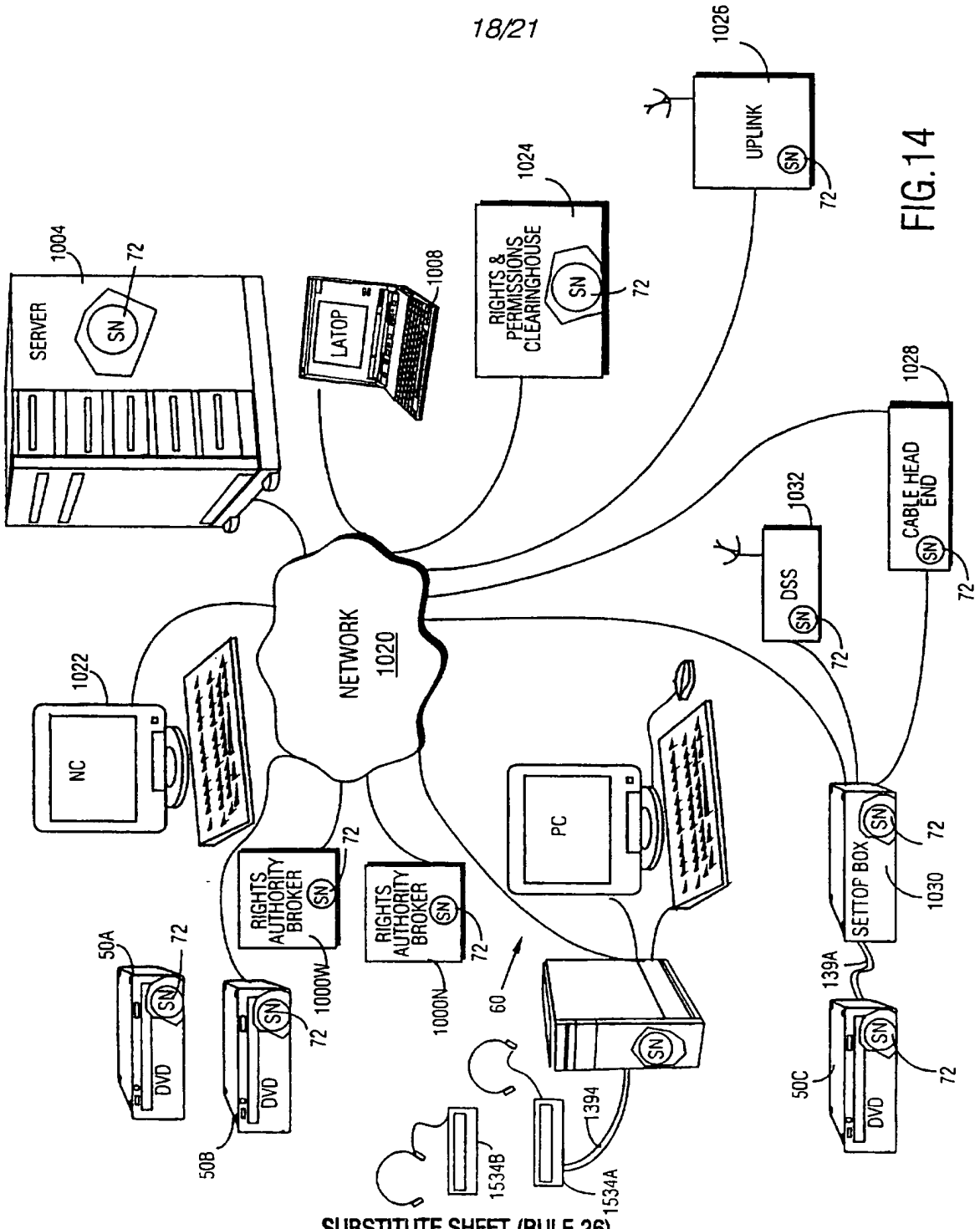


FIG.14

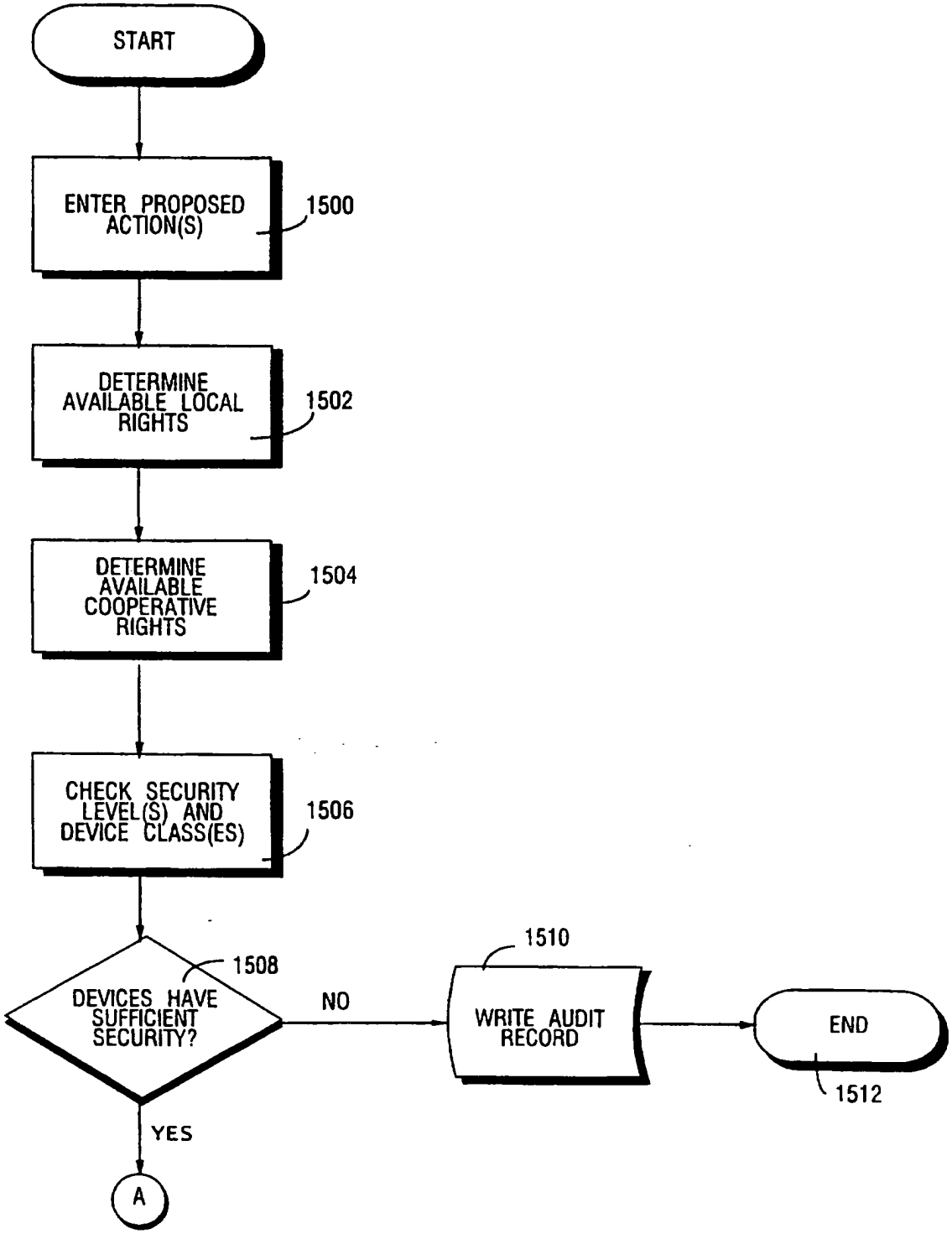


FIG.15A

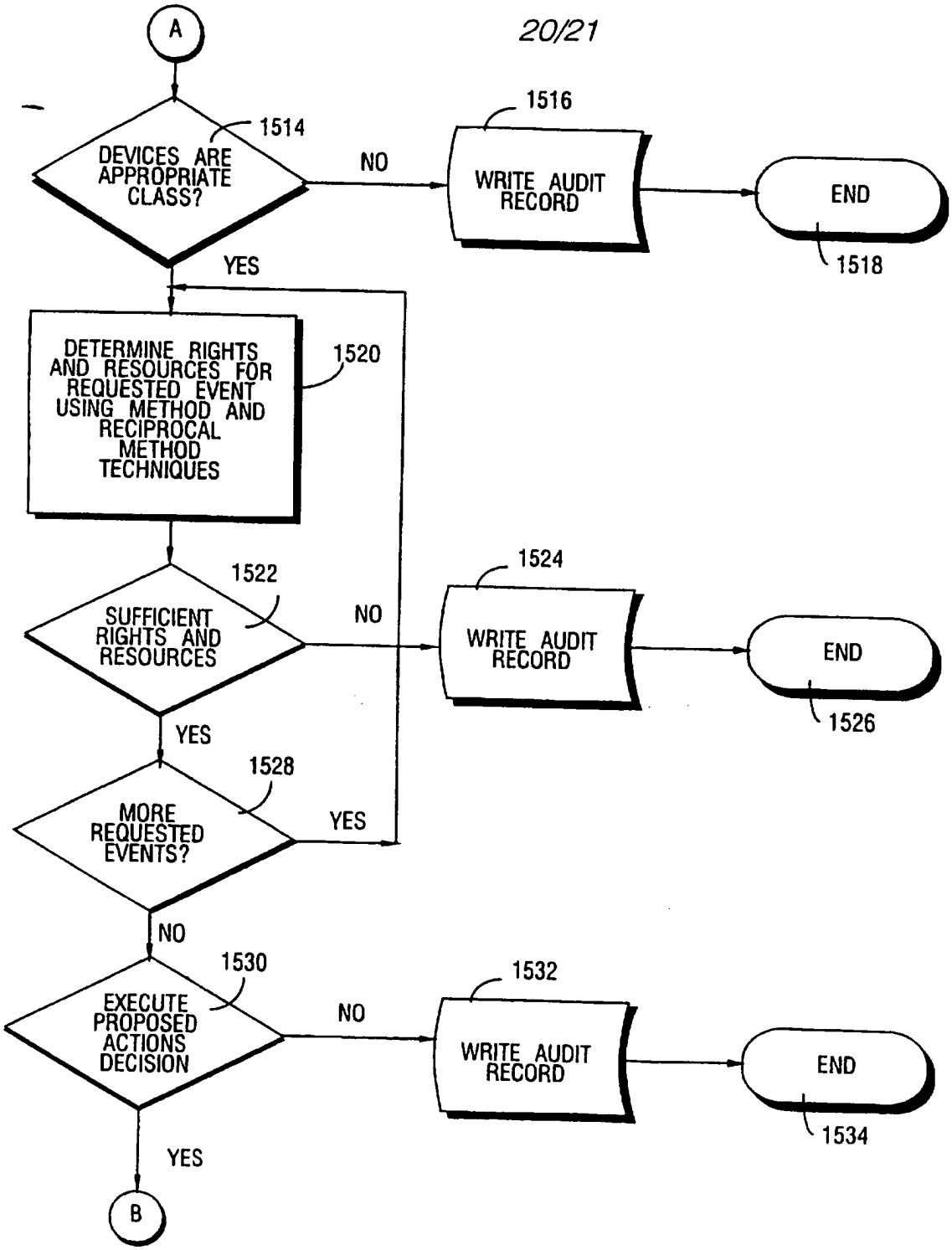
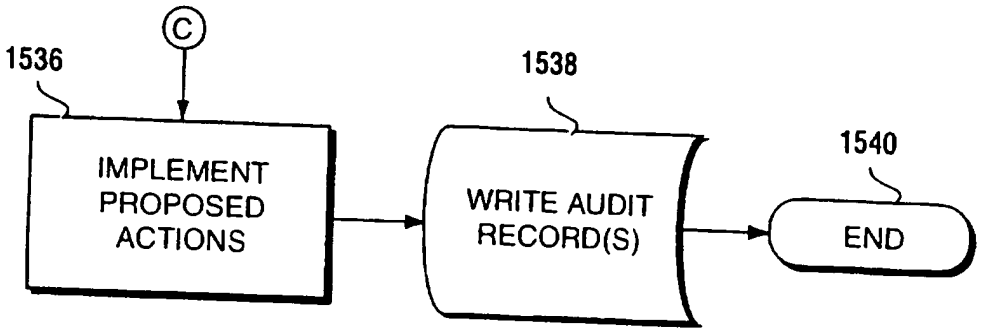


FIG.15B

FIG. 15C



PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 1/00, 12/14 — A2

(11) International Publication Number: **WO 96/24092**

(43) International Publication Date: 8 August 1996 (08.08.96)

(21) International Application Number: PCT/SE96/00115

(22) International Filing Date: 1 February 1996 (01.02.96)

(30) Priority Data:
9500355-4 1 February 1995 (01.02.95) SE

(71)(72) Applicant and Inventor: BENSON, Greg [US/SE]; Dalbackavägen 3, S-240 10 Dalby (SE).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: URICH, Gregory, H. [US/SE]; Warholmsvägen 8 B, S-224 65 Lund (SE).

(74) Agent: AWAPATENT AB; P.O. Box 5117, S-200 71 Malmö (SE).

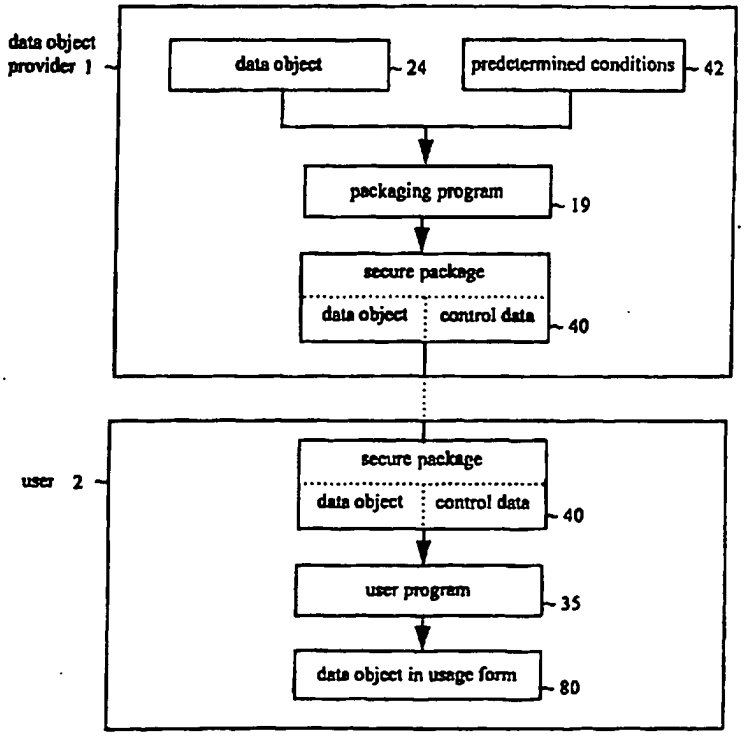
(81) Designated States: AL, AM, AT, AT (Utility model), AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, CZ (Utility model), DE, DE (Utility model), DK, DK (Utility model), EE, EE (Utility model), ES, FI, FI (Utility model), GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SK (Utility model), TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (KE, LS, MW, SD, SZ, UG), Eurasian patent (AZ, BY, KG, KZ, RU, TJ, TM), 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
Without international search report and to be republished upon receipt of that report.

(54) Title: A METHOD AND SYSTEM FOR MANAGING A DATA OBJECT SO AS TO COMPLY WITH PREDETERMINED CONDITIONS FOR USAGE

(57) Abstract

The present invention relates to a method and a system for managing a data object so as to comply with predetermined conditions for usage of the data object. To control the usage of the data object, a set of control data, defining usages of the data object which comply with the predetermined conditions, is created for the data object. The data object is concatenated with the user set of control data, encrypted and transferred to the user. When the user wants to use the data object, a special user program checks whether the usage complies with the control data. If so, the usage is enabled. Otherwise it is disabled.



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.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

A METHOD AND SYSTEM FOR MANAGING A DATA OBJECT SO AS TO COMPLY WITH PREDETERMINED CONDITIONS FOR USAGE

Technical Field

The present invention relates to data processing and more particularly to a method and a system for managing data objects so as to comply with predetermined conditions for usage.

Background

Much has been written recently regarding the puzzle of universal connectivity. A typical vision of the data highway has long distance high speed data carriers interconnecting regional networks which provide telecommunications services and a wide range of interactive on-line services to consumers. Many of the pieces are already in place, others are in development or testing. In fact, even though the data highway is under construction it is currently open to limited traffic. On-line services are springing up daily and video on demand services are currently being tested.

The potential to benefit society is immense. The scope of information available to consumers will become truly global as the traditional barriers to entry for distribution of, and access to, information are lowered dramatically. This means that more diverse and specialized information will be made available just as conveniently as generic sources from major vendors used to be. The end result is that organizations and individuals will be empowered in ways heretofore only imagined.

However, a fully functioning data highway will only be as valuable as the actual services which it provides. Services envisioned for the data highway that involve the delivery of data objects (e.g. books, films, video, news, music, software, games, etc.) will be and are currently limited by the availability of such objects. Library and educational services are similarly affected. Before owners will allow their data objects to be offered they

must be assured of royalty payments and protection from piracy.

Encryption is a key component of any solution to provide copy protection. But encryption alone is not enough. During transmission and storage the data objects will be protected by encryption, but as soon as anyone is given the key to decipher the content he will have unlimited control over it. Since the digital domain permits data objects to be reproduced in unlimited quantities with no loss of quality, each object will need to be protected from unlimited use and unauthorized reproduction and resale.

The protection problem must not be solved by a separate solution for each particular data format, because then the progress will indeed be slow. It is important to consider the effect of standardization on an industry. Consider how the VHS, the CD and the DAT formats, and the IBM PC compatibility standards have encouraged growth in their respective industries. However, if there is to be any type of standardization, the standard must provide universal adaptability to the needs of both data providers and data users.

The data object owner may want to have permanent secure control over how, when, where, and by whom his property is used. Furthermore, he may want to define different rules of engagement for different types of users and different types of security depending on the value of particular objects. The rules defined by him shall govern the automated operations enabled by data services and networking. The owner may also want to sell composite objects with different rules governing each constituent object. Thus, it is necessary to be able to implement variable and extensible control.

The user on his part wants to be able to search for and purchase data objects in a convenient manner. If desired, the user should be able to combine or edit purchased objects (i.e. for creating a presentation).

Furthermore, the user may want to protect his children from inappropriate material. A complete solution must enable these needs as well.

What is needed is a universally adaptable system and method for managing the exchange and usage of data objects while protecting the interests of data object owners and users.

Prior Art

A method for enforcing payment of royalties when copying softcopy books is described in the European patent application EP 0 567 800. This method protects a formatted text stream of a structured document which includes a royalty payment element having a special tag. When the formatted text stream is inputted in the user's data processor, the text stream is searched to identify the royalty payment element and a flag is stored in the memory of the data processor. When the user for instance requests to print the document, the data processor requests authorization for this operation from a second data processor. The second data processor charges the user the amount indicated in the royalty payment element and then transmits the authorization to the first data processor.

One serious limitation of this method is that it can only be applied to structured documents. The description of the above-mentioned European patent application defines a structured document as: a document prepared in accordance with an SGML-compliant type definition. In other words it can not be applied to documents which are not SGML compliant and it cannot be applied to any other types of data objects.

Furthermore, this method does not provide for variable and extensible control. Anyone can purchase a softcopy book on a CD, a floppy disc or the like, and the same royalty amount is indicated in the royalty payment element of all softcopy books of the same title.

Thus, the method described in EP 0 567 800 does not satisfy the above-mentioned requirements for universally adaptable protection of data objects.

Summary of the Invention

Accordingly, it is a first object of the invention to provide a method and a data processing system for managing a data object in a manner that is independent of the format and the structure thereof, so as to comply with predetermined conditions for usage control and royalty payment.

It is a further object of the invention to provide such a method and system which is universally adaptable to the needs of both the owner and the user of the data object.

A further object of the invention is to provide such a method and system which enables a data object provider to distribute his data object while maintaining control of the usage thereof.

Yet another object of the invention is to provide a method and system which allows a data object provider to select the level of security for his data object in a flexible way.

Yet another object of the invention is to provide such a method and system which makes it possible to establish an audit trail for the data object.

Yet another object is to provide such a method and system which makes it possible to sell and buy data objects in a secure way.

The above-mentioned objects are achieved by a method and a system having the features of claims 1, 16, 21, 24 and 27.

Particular embodiments of the inventions are recited in the subclaims.

More particularly, a data object provider, e.g. the owner of a data object or his agent (broker), stores the data object in a memory device, e.g. a bulk storage device, where it is accessible by means of the data

provider's data processor. The data object can consist of digital data, analog data or a combination or hybrid of analog and digital data.

A general set of control data, which is based on the predetermined conditions for usage of the data object, is created and stored in the same memory device as the data object or another memory device where it is accessible by the data provider's data processor. The predetermined conditions for usage may be defined by the data object owner, by the broker or by anyone else. They may differ between different data objects.

The general set of control data comprises at least one or more usage control elements, which define usages of the data object which comply with the predetermined conditions. These usages may encompass for instance the kind of user, a time limit for usage, a geographical area for usage, allowed operations, such as making a hard copy of the data object or viewing it, and/or claim to royalty payment. The general set of control data may comprise other kinds of control elements besides the usage control element. In a preferred embodiment, the general set of control data comprises a security control element which defines a security procedure which has to be carried out before usage of the data object. It also comprises an identifier, which uniquely identifies the general set of control data.

The general set of control data is concatenated with a copy of the data object. Thus, the control data does not reside in the data object, but outside it, which makes the control data independent of the format of and the kind of data object and which allows for usage control independently of the data object format.

At least the usage control element(s) and the data object are encrypted, so that the user is unable to use the data object without a user program which performs the usage control and which decrypts the data object. Alter-

natively, the whole set of control data and the copy of the data object may be encrypted.

A user may request authorization for usage of a data object residing at a data provider's processor via a data network or in any other appropriate way. The authorization may or may not require payment. When a request for authorization for usage is received, a user set of control data is created by the data provider's processor. The user set of control data comprises the general set of control data or a subset thereof including at least one of said usage control elements which is relevant for the actual user. It typically also includes a new identifier which uniquely identifies this set of control data. If relevant, the user set of control data also comprises an indication of the number of usages authorized. If more than one kind of usage is authorized, the number of each kind of usage may be specified. Finally, the user set of control data is concatenated with a copy of the data object, and at least the usage control elements and the copy of the data object are encrypted to create a secure data package ready for transfer to the user.

Before the data package is transferred to the user, it should be confirmed that the request for authorization for usage has been granted. The check is preferably carried out before the user set of control data is created. However, it can also be carried out in parallel with or after the creation of the user control data. In the latter case, the number of usages requested by the user is tentatively authorized and included in the user set, but if the request is refused the user set is cancelled or changed.

The data package may be transferred to the user by electronic means or stored on bulk storage media and transferred to the user by mail or by any suitable transportation means.

Once the data object has been packaged in the above-described manner, it can only be accessed by a user

program which has built-in usage control and means for derypting the data package. The user program will only permit usages defined as acceptable in the control data. Moreover, if the control data comprises a security control element, the security procedure prescribed therein has to be complied with. In one embodiment, the usage control may be performed as follows. If the user decides to use a data object, the user program checks the control data to see if this action is authorized. More particularly, it checks that the number of authorized usages of this kind is one or more. If so, the action is enabled and the number of authorized usages decremented by one. Otherwise, the action is interrupted by the user program and the user may or may not be given the opportunity to purchase the right to complete the action.

After the usage, the user program repackages the data object in the same manner as it was packaged before.

When a data object is redistributed by a user or a broker, new control elements are added in the control data to reflect the relation between the old user/broker and the new user/broker. In this way, an audit trail for the data object may be created.

According to another aspect of the invention at least two data packages are stored on a user's data processor, which examines the usage control elements of the data packages in order to find a match. If a match is found, the user's data processor carries out an action which is specified in the user set of control data. This method can be used for selling and buying data objects.

Brief Description of Drawings

Fig. 1 is a flow diagram showing the general data flow according to the invention.

Fig. 2 is a system block diagram of a data object provider's data processor.

Fig. 3 is a block diagram showing the different modules of a data packaging program according to the invention.

Fig. 4 is a data flow diagram of a data packaging process.

Fig. 5 is an example of a header file.

Fig. 6 is an example of a usage data file.

Fig. 7 is a data flow diagram of loading an object to the data object provider's data processor.

Figs 8a and 8b are examples of control data for a data object on the data object provider's data processor and for an object ready to be transferred to a user, respectively.

Fig. 9 is a data flow diagram of data packaging on the data object provider's data processor.

Fig. 10 is a flow diagram of a data packaging procedure.

Fig. 11 is a memory image of a data object and its control data.

Fig. 12a is a memory image of the concatenated control data and data object.

Fig. 12b is a memory image of the concatenated and encrypted control data and data object.

Fig. 13 is a system block diagram of a user's data processor.

Fig. 14 is a block diagram showing the different modules of a user program according to the invention.

Fig. 15 is a flow diagram of using a data object on the user's data processor.

Fig. 16 is a flow diagram of how the user program operates in a specific application example.

Fig. 17 is an example of various data package structures for composite objects.

Description of the Best Mode for Carrying Out the Invention

General Overview

Fig. 1 is a flow diagram showing the general data flow according to the invention. The flow diagram is divided into a data object provider part 1 and a user part 2.

In the data object provider part 1, a data object 24 is created by an author. The data object can consist of digital data, analog data or a combination or hybrid of analog and digital data. The primary difference between analog data objects and digital data objects is the means for storage, transfer and usage.

The author also determines the conditions 42 for the usage of the data object 24 by a user. The data object 24 and the usage conditions 42 are input to a data packaging program 19, which creates a secure data package 40 of the data object and of control data which are based on the input usage conditions 42. Once packaged in this way, the data object can only be accessed by a user program 35.

The data object may be packaged together with a general set of control data, which is the same for all users of the data object. This may be the case when the data object is sent to a retailer or a bulletin board, wherefrom a user may obtain it. The data object may also be packaged as a consequence of a request from a user for usage of the data object. In that case, the package may include control data which is specifically adapted to that user. This control data is called a user set of control data. It may for example comprise the number of usages purchased by the user. Typically, the user set of control data will be created on the basis of the general set of control data and include at least a subset thereof. A user set of control data need not always be adapted for a specific user. All sets of control data which are created on the basis of a general set of control data will be called a user set of control data. Thus, a set of control data can be a general set in one phase and a user set in another phase.

The above-mentioned data packaging can be carried out by the author himself by means of the data packaging program 19. As an alternative, the author may send his data object to a broker, who inputs the data object and the usage conditions determined by the author to the data

packaging program 19 in order to create a secure package 3. The author may also sell his data object to the broker. In that case, the broker probably wants to apply his own usage conditions to the data packaging program. The author may also provide the data object in a secure package to the broker, who repackages the data object and adds further control data which is relevant to his business activities. Various combinations of the above alternatives are also conceivable.

In the user part 2 of the flow diagram, the secure package 40 is received by a user, who must use the user program 35 in order to unpackage the secure package 40 and obtain the data object in a final form 80 for usage. After usage, the data object is repackaged into the secure package 40.

The different parts of the system and the different steps of the method according to the invention will now be described in more detail.

The data provider's data processor:

Fig. 2 is a system block diagram of a data object provider's data processor. As mentioned above, the data object provider may be an author of a data object, an owner of a data object, a broker of a data object or anyone else who wants to distribute a data object, while retaining the control of its usage. The data processor is a general or special purpose processor, preferably with network capabilities. It comprises a CPU 10, a memory 11 and a network adapter 12, which are interconnected by a bus 13. As shown in Fig. 2, other conventional means, such as a display 14, a keyboard 15, a printer 16, a bulk storage device 17, and a ROM 18, may also be connected to the bus 13. The memory 11 stores network and telecommunications programs 21 and an operating system (OS) 23. All the above-mentioned elements are well-known to the skilled person and commercially available. For the purpose of the present invention, the memory 11 also stores a data packaging program 19 and, preferably, a database

20 intended for control data. Depending upon the current operation, one or more data objects 24 can be stored in the memory 11 as shown or in the bulk storage 17. The data provider's data processor is considered secure.

The Data Packaging Program:

The data packaging program 19 is used for creating control data for controlling the usage of a data object and for packaging the data object and the control data into a secure package.

As shown in Fig. 3, it comprises a program control module 301, a user interface module 302, a packaging module 303, a control data creation module 304, an encryption module 305, one or more format modules 306, and one or more security modules 307.

The control module 301 controls the execution of the other modules. The user interface module 302 handles interaction with the data object provider. The packaging module 303 packages the control data and the data object. It uses the control data creation module 304, the format modules 306, the security modules 307 and the encryption module 305 as will be described more in detail below.

The format modules 306 comprise program code, which is required to handle the data objects in their native format. They can fulfill functions such as data compression and data conversion. They can be implemented by any appropriate, commercially available program, such as by means of a routine from the PKWARE Inc. Data Compression Library for Windows and the Image Alchemy package from Handmade Software Incorporated, respectively. They can also be implemented by custom designed programs.

The security modules 307 comprise program code required to implement security, such as more sophisticated encryption than what is provided by the encryption module 305, authorization algorithms, access control and usage control, above and beyond the basic security inherent in the data package.

The data packaging program 19 can contain many different types of both format and security modules. The program control module 301 applies the format and security modules which are requested by the data provider.

The encryption module 305 may be any appropriate, commercially available module, such as "FileCrypt" Visual Basic subprogram found in Crescent Software's QuickPak Professional for Windows - FILECRPT.BAS, or a custom designed encryption program.

The control data creation module 304 creates the control data for controlling the usage of the data object. An example of a control data structure will be described more in detail below.

The Control Data:

The control data can be stored in a header file and a usage data file. In a preferred embodiment, the header file comprises fields to store an object identifier, which uniquely identifies the control data and/or its associated data object, a title, a format code, and a security code. The format code may represent the format or position of fields in the usage data file. Alternatively, the format code may designate one or more format modules to be used by the data packaging program or the user program. The security code may represent the encryption method used by the encryption module 305 or any security module to be used by the data packaging program and the user program. The header file fields will be referred to as header elements.

The usage data file comprises at least one field for storing data which controls usage of the data object. One or more usage data fields which represent one condition for the usage of the data object will be referred to as a usage element. In a preferred embodiment, each usage element is defined by an identifier field, e.g. a serial number, a size field, which specifies the size of the usage element in bytes or in any other appropriate way, and a data field.

The header elements and the usage elements are control elements which control all operations relating to the usage of the object. The number of control elements is unlimited. The data provider may define any number of control elements to represent his predetermined conditions of usage of the data object. The only restriction is that the data packaging program 19 and the user program 35 must have compatible program code to handle all the control elements. This program code resides in the packaging module and the usage manager module, to be described below.

Control elements can contain data, script or program code which is executed by the user program 35 to control usage of the related data object. Script and program code can contain conditional statements and the like which are processed with the relevant object and system parameters on the user's data processor. It would also be possible to use a control element to specify a specific proprietary user program which can only be obtained from a particular broker.

It is evident that the control data structure described above is but one example. The control data structure may be defined in many different ways with different control elements. For example, the partitioning of the control data in header data and usage data is not mandatory. Furthermore, the control elements mentioned above are but examples. The control data format may be unique, e.g. different for different data providers, or defined according to a standard.

The operation of the data packaging program

The operation of a first embodiment of the data packaging program will now be described with reference to the block diagram of Fig. 3 and the flow diagram of Fig. 4.

First a data provider creates a data object and saves it to a file, step 401. When the data packaging program is started, step 402, the user interface module

302 prompts the data object provider to input, step 403, the header information consisting of e.g. an object identifier, a title of the data object, a format code specifying any format module to be used for converting the format of the data object, and a security code specifying any security module to be used for adding further security to the data object. Furthermore, the user interface module 302 prompts the data object provider to input usage information, e.g. his conditions for the usage of the data object. The usage information may comprise the kind of user who is authorized to use the data object, the price for different usages of the object etc. The header information and the usage information, which may be entered in the form of predetermined codes, is then passed to the control module 301, which calls the packaging module 303 and passes the information to it.

The packaging module 303 calls the control data creation module 304, which first creates a header file, then creates header data on the basis of the header information entered by the data object provider and finally stores the header data, step 404-405. Then a usage data file is created, usage data created on the basis of the usage information entered by the data provider, and finally the usage data is stored in the usage data file, step 406-407.

The packaging module 303 then applies any format and security modules 306, 307 specified in the header file, steps 408-413, to the data object.

Next, the packaging module 303 concatenates the usage data file and the data object and stores the result as a temporary file, step 414. The packaging module 303 calls the encryption module 305, which encrypts the temporary file, step 415. The level of security will depend somewhat on the quality of the encryption and key methods used.

Finally, the packaging module 303 concatenates the header file and the encrypted temporary file and saves

the result as a single file, step 416. This final file is the data package which may now be distributed by file transfer over a network, or on storage media such as CD-ROM or diskette, or by some other means.

Example 1

An example of how the data packaging program 19 can be used will now be described with reference to Figs 5 and 6. In this example the data object provider is a computer graphics artist, who wants to distribute an image that can be used as clip art, but only in a document or file which is packaged according to the method of the invention and which has usage conditions which do not permit further cutting or pasting. The artist wants to provide a free preview of the image, but also wants to be paid on a per use basis unless the user is willing to pay a rather substantial fee for unlimited use. The artist will handle payment and usage authorization on a dial-up line to his data processor.

The artist uses some image creation application, such as Adobe's Photoshop to create his image. The artist then saves the image to file in an appropriate format for distribution, such as the Graphical Interchange Format (GIF). The artist then starts his data packaging program and enters an object identifier, a title, a format code and a security code, which in this example are "123456789", "image","a", and "b", respectively. In this example, the format code "a" indicates that no format code need be applied, and this code is selected since the GIF format is appropriate and already compressed. Furthermore, the security code "b" indicates that no security module need be applied and this code is selected since the security achieved by the encryption performed by means of the encryption module 305 is considered appropriate by the artist.

Then the artist enters his dial-up phone number, his price for a single use of the image and for unlimited use of the data object, a code for usage types approved, and

for number of usages approved. For this purpose, the user interface module 302 may display a data entry form.

The data packaging program 19 creates control data on the basis of the information entered by the artist and stores the data in the header file and in the usage data file as shown in Figs 5 and 6, respectively. This data consitutes a general set of control data which is not specifically adapted to a single user, but which indicates the conditions of usage determined by the artist for all future users.

Then the package program 19 concatenates the data object and the control data in accordance with steps 414-416 of Fig. 4 to achieve the secure package. No format module or security module is applied to the data object, since they are not needed according to the data in the header file.

When the secure package has been obtained, the artist sends it to a bulletin board, from where it can be retrieved by a user.

Example 2

Below, another embodiment of the data packaging program 19 will be described with reference to Figs 7-12b. In this example, the data object consists of a video film, which is created by a film company and sent to a broker together with the predetermined conditions 42 for usage of the video. The broker loads the video 24 to the bulk storage 17 of his data processor. Then, he uses his data packaging program 19 to create a general set of control data 50 based on the predetermined conditions 42 for usage indicated by the film company. Furthermore, the address to the video in the bulk storage 17 is stored in an address table in the control database 20 or somewhere else in the memory 11. It could also be stored in the general set of control data 50. Finally, the general set of control data 50 is stored in the control database 20. It could also be stored somewhere else in the memory 11.

After these operations, which correspond to steps 401-407 of Fig. 4, the data packaging program is exited.

Fig. 8a shows the general set of control data for the video according to this example. Here the control data includes an identifier, a format code, a security code, the number of usage elements, the size of the data object, the size of the usage elements and two usage elements, each comprising an identifier field, a size field and a data field. The identifier may be a unique number in a series registered for the particular broker. In this example, the identifier is "123456789", the format code "0010", which, in this example, indicates the format of a AVI video and the security code is "0010". Furthermore, the first usage element defines the acceptable users for the video and the second usage element data defines the number of viewings of the video purchased by a user. The first usage element data is 1 which, for the purposes of this example will signify that only education oriented users are acceptable to the film company. The data field of the second usage element data is empty, since at this stage no viewings of the video has been purchased.

Managing Object Transfer:

The broker wants to transfer data objects to users and enable controlled usage in return for payment of usage fees or royalties. Managing the broker-user business relationship and negotiating the transaction between the broker and the user can both be automated, and the control data structure can provide unlimited support to these operations. The payment can be handled by transmitting credit card information, or the user can have a debit or credit account with the broker which is password activated. Preferably, payment should be confirmed before the data object is transferred to the user.

Data packaging:

When a user wants to use a data object, he contacts the broker and requests authorization for usage of the data object. When the request for authorization is recei-

ved in the broker's data processor, a data program compares the usage for which authorization is requested with the usage control elements of the control data of the data object to see if it complies with the predetermined conditions for usage indicated therein. The comparison may include comparing the user type, the usage type, the number of usages, the price etc. If the requested usage complies with the predetermined conditions the authorization is granted, otherwise it is rejected.

Fig. 9 is a data flow diagram of the data packaging on the broker's data processor, which occurs in response to a granted request from a user for authorization for usage of the video, e.g. a granted request for the purchase of two viewings.

In response to a granted request, the broker again applies the data packaging program 19. The general set of control data 50 and the data object 24 are input to the program from the control database 20 and the bulk storage 17, respectively. The program creates a user set of control data 60 on the basis of the general set of control data 50 and concatenates the user set 60 and the data object 24 to create a secure data package 40, which may then be transferred to the user by any suitable means. A copy of the user set of control data is preferably stored in the broker's control database. This gives the broker a record with which to compare subsequent use, e.g. when a dial-up is required for usage.

Fig. 10 is a flow diagram of an exemplary procedure used for creating a user set of control data and for packaging the user set of control data and the video into a secure package. Here, the procedure will be described with reference to the general set of control data shown in Fig. 8a.

The user set of control data 60, i.e. a set of control data which is adapted to the specific user of this example, is created in steps 1001-1003 of Fig. 11. First, the general set of control data 50 stored in the control

database is copied to create new control data, step 1001. Second, a new identifier, here "123456790", which uniquely identifies the user set of control data, is stored in the identifier field of the new control data 60, step 1002. Third, the data field of the second usage element is updated with the usage purchased, i.e. in this example with two, since two viewings of the video were purchased, step 1003.

The thus-created user set of control data, which corresponds to the general set of control data of Fig. 8a is shown in Fig. 8b.

The user set of control data is stored in the control database 20, step 1004. Then, the video, which is stored in the bulk storage 17, is copied, step 1005. The copy of the video is concatenated with the user set of control data, step 1006. The security code 0010 specifies that the entire data package 40 is to be encrypted and that the user program 35 must contain a key which can be applied. Accordingly, the whole data package is encrypted, step 1007. Finally, the encrypted data package is stored on a storage media or passed to a network program, step 1008, for further transfer to the user.

Fig. 11 is a memory image of the video 24 and the user control data 60. The user control data and a copy of the video 24 are concatenated as shown in Fig. 12a. The encrypted data package 40 is shown in Fig. 12b.

The procedure of Fig. 10 can be implemented by the data packaging program of Fig. 3. As an alternative to the procedure of Fig. 10, the user set of control data can be created as in steps 1001-1003 and saved in a header file and in a usage data file, whereafter steps 408-416 of the data packaging program of Fig. 4 can be performed to create the secure package.

The above-described process for creating a user-adapted set of control data may also be used by a user who wants to redistribute a data object or by a broker who wants to distribute the data object to other brokers.

Obviously, redistribution of the data object requires that redistribution is a usage approved of in the control data of the data object. If so, the user or the broker creates a user set of control data by adding new control elements and possibly changing the data fields of old control element to reflect the relation between the author and the current user/broker and between the current user/broker and the future user/broker. In this way, an audit trail is created.

The user's data processor:

The user's data processor, which is shown in Fig. 13, is a general or special purpose processor, preferably with network capabilities. It comprises a CPU 25, a memory 26, and a network adapter 27, which are interconnected by a bus 28. As shown in Fig. 13, other conventional means, such as a display 29, a keyboard 30, a printer 31, a sound system 32, a ROM 33, and a bulk storage device 34, may also be connected to the bus 28. The memory 26 stores network and telecommunications programs 37 and an operating system (OS) 39. All the above-mentioned elements are well-known to the skilled person and commercially available. For the purpose of the present invention, the memory 26 also stores a user program 35 and, preferably, a database 36 intended for the control data. Depending upon the current operation, a data package 40 can be stored in the memory 26, as shown, or in the bulk storage 34.

The user program:

The user program 35 controls the usage of a data object in accordance with the control data, which is included in the data package together with the data object.

As shown in Fig. 14, the user program 35 comprises a program control module 1401 a user interface module 1402, a usage manager module 1403, a control data parser module 1404, a decryption module 1405, one or more format modules 1406, one or more security modules 1407, and a file transfer program 1409.

The control module 1401 controls the execution of the other modules. The user interface module 1402 handles interactions with the user. The usage manager module 1403 unpackages the secure package 40. It uses the control data parser module 1404, the decryption module 1405, the format modules 1406, and the security modules 1407.

The format modules 1406 comprise program code, which is necessary to handle the data objects in their native format, such as decompression and data format procedures. The security modules 1407 comprises program code required to implement security above the lowest level, such as access control, usage control and more sophisticated decryption than what is provided by the basic decryption module 1405.

The user program 35 can contain many different types of both format and security modules. However, they should be complementary with the format and security modules used in the corresponding data packaging program. The usage manager module 1401 applies the format and security modules which are necessary to use a data object and which are specified in its control data. If the proper format and security modules are not available for a particular data object, the usage manager module 1401 will not permit any usage.

The decryption module 1405 can be the above-mentioned FileCrypt Visual Basic subprogram or some other commercially available decryption program. It can also be a custom designed decryption module. The only restriction is that the decryption module used in the user program is complementary with the encryption module of the data packaging program.

The control data parser module 1403 performs the reverse process of the control data creation module 304 in Fig. 3.

The user program 35 can have code which controls use of the program by password or by any other suitable method. A password may be added in a password control

element during packaging of the data object. The password is tranferred to the user by registered mail or in any other appropriate way. In response to the presence of the password control element in the control data structure, the user program prompts the user to input the password. The input password is compared with the password in the control data, and if they match, the user program continues, otherwise it is disabled.

The user program 35 can also have procedures which alter the behavior of the program (e.g. provide filters for children) according to the control data of the user object 41. It is important to mention that the user program 35 never stores the object in native format in user accessible storage and that during display of the data object the print screen key is trapped.

The file transfer program 1409 can transfer and receive files via network to and from other data processor.

Since the data object is repackaged into the secure package after the usage, the user program should also include program code for repackaging the data object. The program code could be the same as that used in the corresponding data packaging program 19. It could also be a separate program which is called from the user program.

Operation of the user program:

The operation of an embodiment of the user program 35 will now be described with reference to the block diagram of Fig. 14 and the flow diagram of Fig. 15.

First the user receives a data package 40 via file transfer over a network, or on a storage media such as CD-ROM or diskette, or by any other appropriate means, step 1501. He then stores the data package as a file on his data processor, step 1502.

When the user wants to use the data object, he starts the user program 35, step 1503. Then he requests usage of the data object, step 1504. The request is received by the user interface module 1402, which noti-

fies the control module 1401 of the usage request. The control module 1401 calls the usage manager module 1403 and passes the usage request.

The usage manager module 1403 reads the format code from the data package to determine the control data format. Then it calls the decryption module 1405 to decrypt and extract the control data from the data package. The usage manager module 1403 applies the decryption module 1405 incrementally to decrypt only the control data. Finally, it stores the control data in memory, step 1505.

The usage manager module 1403 then calls the control data parser module 1404 to extract the data fields from the usage elements.

The usage manager module 1403 then compares the user request for usage with the corresponding control data, steps 1506-1507. If the requested usage is not permitted in the control data, the requested usage is disabled, step 1508. However, if the requested usage is approved of in the control data, the usage manager module 1403 applies any format and security modules 1406, 1407 specified in the header data or usage data, steps 1509-1514, to the data package.

Then the usage manager module 1403 calls the decryption module 1405, which decrypts the object data, step 1515, whereafter the requested usage is enabled, step 1516. In connection with the enabling of the usage, the control data may need to be updated, step 1517. The control data may for instance comprise a data field indicating a limited number of usages. If so, this data field is decremented by one in response to the enabling of the usage. When the user has finished usage of the data object, the user program 35 restores the data package in the secure form by repackaging it, step 1518. More particularly, the data object and the usage elements are reconcatenated and reencrypted. Then the header elements are added and the thus-created package is stored in the user's data processor.

Example 1 contd.

A specific example of how the user program operates will now be described with reference to Figs 6 and 15. The example is a continuation of Example 1 above, where an artist created an image and sent it to a bulletin board.

Assume that a user has found the image at an electronic bulletin board (BBS) and is intrested in using it. He then loads the data package 40 containing the image to his data processor and stores it as a file in the bulk storage. The user then executes the user program 35 and requests to preview the image. The user program then performs steps 1505-1507 of the flow diagram in Fig. 15. The request for a preview of the image is compared with the data field of the usage element "code for usage type approved". In this example, the code "9" designates that previews are permitted. Thus, the requested preview is OK. Then, the user program 35 performs step 1509-1515 of Fig. 15. Since the format code "a" and the security code "b" of the header data indicate that neither conversion, nor decompression, nor security treatment is required, the user program only decrypts the object data. The usage manager module 1403 then displays the preview on the user's data processor and passes control back to the user interface 1402.

When the user is finished previewing the image, the user interface module 1402 displays the costs for usage of the image in accordance with the price usage data of the control data ("price for single use" and "price for unlimited use" in Fig. 6) and prompts the user to enter a purchase request. The user decides to buy unlimited use of the image, and the user interface module 1402 inputs purchase information, such as an identification, billing, and address for that request and passes the request to the control module 1401. The control module calls the file transfer program 1409, which dials the artist's dial-up number as indicated in the usage data ("control

element for artist's phone number" in Fig. 6) and transfers the request and purchase information to a broker program on the artist's data processor. Upon approval of the purchase, the broker program returns a file containing an update for "usage type approved" control elements. The update is "10" for the usage type approved, which in this example indicates that unlimited use by that user is permitted. The file transfer program 1409 passes this update to the usage manager module 1403 which updates the control data with the "usage type approved" code. The user interface module 1402 then displays a confirmation message to the user.

Subsequently, the user interface module inputs a request to copy the image to a file packaged according to this invention, on the user's machine. The usage manager module then compares the user request control data. The usage manager module examines the data filed for "usage type approved", which now is "10". The usage manager module copies the image to the file.

When the user is finished with the image, the usage manager module 1403 repackages the image as before except with updated control data. This repackaging process is exactly like that shown in Fig. 4, except that the header and usage data already exist, so the process starts after step 406 where control data is created.

Improved security

If the data object provider wants to improve the security of a data package containing a data object, a security module 307 containing a sophisticated encryption algorithm, such as RSA, could be used. In that case the packaging module 303 calls the security module 307 in step 412 of the flow diagram of Fig. 4. The security module encrypts the image and passes a security algorithm code to the control data creation module 302, which adds a control element for the security module code, which will be detected by the user program 35. Then the data packaging continues with step 414. When the data package

is sent to the user, the public key is mailed to the user by registered mail. When the user program is executed in response to a request for usage of this data object, the usage manager module will detect the security module code in the control data and call the security module. This module passes control to the user interface module 1402, which requests the user to input the public key. If the key is correct, the user security module applies complementary decryption using that key and passes a usage approved message to the usage manager module, which enables the usage.

As another example of improved security, a security module may implement an authorization process, according to which each usage of the data object requires a dial-up to the data processor of the data object provider. When the corresponding security module code is detected by the user program 35, the relevant security module is called. This module passes a request for authorization to the control module 1401, which calls the file transfer program 1409, which dial the data object provider's dial-up number, which is indicated in a usage element and transfers the request for authorization of usage. Upon a granted authorization, the data provider's data processor returns a usage approved message to the user security module, which forwards the approval tl the usage control module, which enables one usage. If the user requests further usages of the data object, the authorization process is repeated. This procedures results in a permanent data object security.

Example 2 contd.

A further specific example of how the user program 35 operates will now be described with reference to Fig. 16. The example is a continuation of Example 2 above, where a user purchased two viewings of a video film from a broker.

The user wants to play the video which was purchased and transferred from the broker. The user applies the

user program 35, step 1601, and requests to play the video, step 1602. The user program 35 first examines the user set of control data 60, step 1603. In this example, the user program 35 contains only those format and security modules for objects with format code of 0010 and with a security code of 0010. Consequently, only those types of data objects may be used. If the program encounters other codes it will not enable the usage action, step 1604-1605.

Next, the user program 35 compares the first control element data which is 1, for educational users only, to user information entered by the user on request of the user program. Since the user type entered by the user is the same as that inicated in the first usage element the process continues, steps 1606-1607. Then the user program checks the second control element data which specifies that the number of plays purchased is 2. Consequently, the usage is enabled, step 1609. The user program applies the decryption module with the universal key and the AVI format video is displayed on the display unit 29. Then, the second control element data is decremented by one, step 1610. Finally, the video is repackaged, step 1611

Implementation of Variable and Extensible Object Control:

Object control is achieved through the interaction of the data packaging program 19 and the usage program 35 with the control data. Variation of object control can be applied to a particular object by creating a control data format with control elements defining the control variation and the circumstances in which the variation is applied. Program procedures should then be added to program modules to process the control elements. For example, suppose a broker wants to allow students to print a particular article for free but require business users to pay for it. He defines control elements to represent the user types student and business and the associated costs for each. He then adds program logic to examine the user type and calculate costs accordingly. Object control is

extensible in the sense that the control data format can have as many elements as there are parameters defining the rules for object control.

Implementation of Variable and Extensible Object Security:

Object security is also achieved through the interaction of the data packaging program 19 and the user program 35 with the control data. Security process and encryption/decryption algorithms can be added as program modules. Variation of object security can be applied to a particular object by creating a control data format with control elements defining the security variation and the circumstances in which the variation is applied. Program procedures should be added to program modules to process the control elements. For example, suppose a broker wants to apply minimal security to his collection of current news articles but to apply tight security to his encyclopedia and text books. He defines a control element for security type. He then adds program logic to apply the security algorithms accordingly. Object security is extensible in the sense that multiple levels of security can be applied. The level of security will of course be dependent on the encryption/key method which is implemented in the security modules. One level of security may be to require online confirmation when loading a data object to the user's data processor. This can be implemented in program code in a security module. This permits the broker to check that the object has not already been loaded as well as double check all other parameters.

It is also important to have version control with time stamping between the usage program and the user's control database. Otherwise the database can be duplicated and reapplied to the user program. The user program can place a time stamp in the control database and in a hidden system file each time the control database is accessed. If the time stamps are not identical, the control database has been tampered with and all usage is

disabled. Program code for handling time stamps can reside in a security module.

Handling Composite Objects:

A composite object can be handled by defining a control data format with control elements defining relationships between constituent objects and by defining a parent/child element and a related object id element. For example, suppose a broker wants to include a video and a text book in an educational package. He creates a parent object with control elements referring to the video and textbook objects. He also includes control elements in the control data for the video object and the textbook object referring to the parent object. Finally, he adds program procedures to program modules to process the control elements.

In other words, when the data object is a composite data object including at least two constituent data objects, a respective general set of control data is created for each of the constituent data object and the composite data object. In response to a request from a user, a respective user set of control data is created for each of the constituent data objects as well as for the composite data object.

Examples of various data package structures for composite objects are given in Fig. 17.

Another side of composite objects is when the user wants to combine data objects for some particular use. Combination is a usage action that must be permitted in each constituent data object. A new data object is created with control data linking the constituent data objects. Each constituent data object retains its original control data which continues to control its subsequent usage.

When a user requests authorization for usage of one constituent data object in a composite data object, a user set of control data is created only for that consti-

tuent data object and concatenated only with a copy of that constituent data object.

Scaleable Implementation:

The flexible control data structure and modular program structure permit almost boundless extensibility with regard to implementation of the owner's requirements for usage control and royalty payment. The control data structure can include control elements for complex user types, usage types, multiple billing schemes, artistic or ownership credit requirements and others. Security modules can be included which interact with any variation of the control data structure and the control data. Security modules could require a dial up to the brokers data processor to approve loading or usage actions and to implement approval authentication mechanisms.

User acting as a broker:

A limited or full implementation of the broker's data packaging program can be implemented on the user's machine to permit further distribution or reselling. However, only those data objects with control data permitting further distribution or reselling are enabled in that way.

Rebrokering

An author of a data object may want to allow his original broker to distribute his data objec to other brokers whom will also distribute his image. He then includes a control element which enables rebrokering in tne control data before distributing the data object with its associated control data to the origingal broker. Upon request for rebrokering, the original broker copies the general set of control data and updates the copy to create a user set of control data which will function as the general set of control data on the subsequent brokers data processor. The original broker packages the data object with the user set of control data and transfers the package to the subsequent broker. The subsequent broker then proceeds as if he were an original broker.

Automated transaction negotiation

This is an example of how the predetermined conditions for usage included in the control data can be used for achieving automated transaction negotiation.

Suppose some company wnats to provide a computer automated stock trading. Buy and sell orders could be implemented in the form of data packages and a user program could process the data packages and execute transactions. Data packages could carry digital cash and manage payment based on conditions defined in the control data.

In this example, the buy order is created using a data packaging program according to the invention on the buyer's data processor. The sell order is created using the data packaging program on the seller's data processor. Both orders are used by the the user program on the stock trader's data processor. The usages would take the form of using a sell order data package to sell stock and a buy order data package to buy stock. The rules or conditions for buying and selling stocks could be indicated in the control data of the packages. The data object consists of digital money. In this context it is important to remember that digital money is merely data which references real money or vitual money that is issued and maintained for the purpose of digital transactions.

In this example the buyer starts with a digital money data file. He uses the data packaging program to create control data, e.g. kind of stock, price, quantity, for the purchase, and he then packages the digital money data file and the control data into a secure package as described above.

The seller starts with an empty data file. This empty file is analogous to the digital money data file except it is empty. The seller creates control data, e.g. kind of stock, price, quantity, and packages the empty file and the control data into a secure package.

Both the sell order package and the buy order package are transferred to the data processor of the stock trading company, where they are received and stored in the memory. The user program of the stock trading company examines the control data of the buy and sell order packages in the same way as has been described above and looks for a match. Upon identifying matched buy and sell orders the user program executes a transaction, whereby the digital money is extracted from the buy order data package and transferred to the sell order package. Then the control data of the data packages is updated to provide an audit trail. Both packages are repackaged in the same manner as they were previously packaged and then transferred back to their authors.

The above described technique could be used for selling and buying any object as well as for automated negotiations. Payment may be carried out in other ways than by digital money.

In the general case, the data processor of the user decrypts the usage control elements of the user sets of control data and examines the usage control elements to find a match. In response to the finding of a match, the user's data processor carries out an action which is specified in the user set of control data.

CLAIMS

1. A method for managing a data object so as to comply with predetermined conditions for usage of the data object, comprising the steps of:

- storing the data object in a memory device, where it is accessible by means of a data object provider's data processor;

- creating, by said data processor, a general set of control data for the data object based on said predetermined conditions for usage, said general set of control data comprising at least one or more usage control elements defining usages of the data object which comply with said predetermined conditions;

- storing said general set of control data in a memory device, where it is accessible by said data processor;

- concatenating the general set of control data with a copy of the data object; and

- encrypting at least the copy of the data object and said one or more usage control elements to create a secure data package which is ready for transfer to a user.

2. A method as set forth in claim 1, wherein the step of encrypting comprises encrypting the data object and the general set of control data.

3. A method as set forth in claims 1 or 2, wherein the step of creating control data comprises creating an identifier which uniquely identifies the general set of control data.

4. A method as set forth in claims 1, 2 or 3, wherein the step of creating a general set of control data comprises creating a security control element which identifies a security process to be applied before usage of the data object is allowed.

5. A method as set forth in any of the preceding claims, wherein the step of creating a general set of

control data comprises creating a format control element which identifies the format of the control data.

6. A method as set forth in any of the preceding claims, comprising the further steps of:

- creating, in response to a request for authorization for usage of the data object by a user, a user set of control data, which comprises at least a subset of the general set of control data, including at least one of said usage control elements;
- using the user set of control data instead of the general set of control data in said concatenating step;
- using the at least one usage control element of the user set of control data instead of the one or more usage control elements of the general set of control data in the encrypting step;
- checking, before allowing transfer of the data package to the user, that said request for authorization for usage of the data object has been granted.

7. A method as set forth in any of the preceding claims, further comprising the steps of receiving in said data processor the request for authorization for usage by a user; comparing the usage for which authorization is requested with said one or more usage control elements of the general set of control data and granting the authorization if the usage for which authorization is requested complies with the usages defined by said one or more usage control elements.

8. A method as set forth in claim 7, further comprising the step of securing payment for the requested authorization for usage before granting the authorization.

9. A method as set forth in any one of claims 6-8, wherein the data object is composed of at least two constituent data objects and wherein the user set of control data, in response to a request for authorization for usage of one of said constituent data objects by a user, is created only for that constituent data object and

concatenated only with a copy of that constituent data object.

10. A method as set forth in any one of claims 6-9, wherein the data provider's data processor is connected to a data network and the request for authorization is received from a data processor of the user, which is also connected to the data network, further comprising the step of transferring the data package through the data network to the user's data processor.

11. A method as set forth in any one of claims 6-8 or 10, wherein the data object is a composite data object including at least two constituent data objects and wherein the step of creating a general set of control data comprises the step of creating a respective general set of control data for each of the constituent data objects and the composite data object and wherein the step of creating a user set of control data comprises the step of creating a respective user set of control data for each of the constituent data objects and the composite data object.

12. A method as set forth in any one of claims 6-11, comprising the further step of storing a copy of the user set of control data in the data object provider's processor.

13. A method as set forth in any of the preceding claims, comprising the further steps of:

- receiving the data package in a user's data processor;

- storing the data package in a memory device where it is accessible by means of the user's data processor;

- decrypting said one or more usage control elements;

- checking, in response to a request by the user for usage of the data object, whether the requested usage complies with the usage defined by the at least one usage control element of the general set of control data;

- decrypting, in response to the requested usage complying with the usage defined by the at least one usage control element of the general set of control data, the data object and enabling the requested usage, otherwise disabling it.

14. A method as set forth in any one of claims 6-12, comprising the further steps of:

- receiving the data package in a user's data processor;

- storing the data package in a memory device where it is accessible by means of the user's data processor;

- decrypting the at least one usage control element of the user set of control data;

- checking, in response to a request by the user for usage of the data object, whether the requested usage complies with the usage defined by the at least one usage control element of the user set of control data;

- decrypting, in response to the requested usage complying with the usage defined by the at least one usage control element of the user set of control data, the data object and enabling the requested usage, otherwise disabling it.

15. A method as set forth in claims 13 or 14, comprising the further steps of reconcatenating, after the usage of the data object, the data object and the one or more usage control elements, reencrypting at least the data object and the one or more usage control elements, and storing the thus-repackaged data package in the memory of the user's data processor.

16. A method for controlling the usage by a user of a data object so as to comply with predetermined conditions for usage of the data object, comprising the steps of:

- storing a data package in a memory device, where it is accessible by means of a data processor of the user, said data package comprising the data object and control data, which comprises at least one usage control

element defining a usage of the data object which complies with the predetermined conditions, the data object and said at least one usage control element being encrypted;

- receiving a request by the user for usage of the data object;

- decrypting the control data;

- checking, in response to the request by the user for usage of the data object, whether the requested usage complies with the usage defined by the at least one usage control element of the control data;

- decrypting, in response to the requested usage complying with the usage defined by the at least one usage control element of the control data, the data object and enabling the requested usage, otherwise disabling it.

17. A method as set forth in claim 16, wherein the usage control element is updated after the usage of the data object.

18. A method as set forth in claims 16 or 17, wherein said control data comprises an indication of the number of times the user is authorized to use the data object in accordance with said at least one user control element; wherein the requested usage of the data object is only enabled when said number of times is one or more; and wherein said number of times is decremented by one when the requested usage is enabled.

19. A method as set forth in any one of claims 16-18, wherein the control data comprise a security control element, and further comprising the step of carrying out, before each usage of the data object, a security procedure defined in the security control element.

20. A method as set forth in any one of claims 16-19, wherein the step of checking whether the requested usage complies with the usage defined by the at least one usage control element comprises the step of checking that the user's data processor is capable of carrying out the

security procedure specified in the security control element of the user set of control data, and if not, disabling the usage.

21. A method as set forth in any one of claims 16-20, comprising the further steps of reconcatenating, after the usage of the data object, the data object and the one or more usage control elements, reencrypting at least the data object and the one or more usage control elements, and storing the thus-repackaged data package in the memory of the user's data processor.

22. A system for managing a data object so as to comply with predetermined conditions for usage of the data object, comprising

- first means in the data object provider's data processor for creating a general set of control data for the data object based on the predetermined conditions for usage, said general set of control data comprising at least one or more usage control elements defining usages of the data object which comply with the predetermined conditions;

- storing means, which are accessible by means of said data processor, for storing the data object and the general set of control data;

- concatenating means for concatenating the general set of control data with a copy of the data object; and

- encrypting means for encrypting the copy of the data object and at least said one or more usage control elements to create a secure data package, which is ready for transfer to a user.

23. A system as set forth in claim 22, further comprising

- second means in said data processor for creating, in response to a request for authorization for usage of the data object by a user, a user set of control data, which comprises at least a subset of the general set of control data, which subset comprises at least one of said usage control elements; and

- checking means in said data processor for checking that said request for authorization for usage of the data object has been granted before allowing transfer of the data package to the user.

24. A system as set forth in claims 22 or 23, wherein the general set of control data comprises a control data element which defines the right to further distribution of the data object by the user.

25. A system for controlling the usage by a user of a data object so as to comply with predetermined conditions for usage of the data object, comprising

- storing means for storing a data package which comprises a data object and a control data comprising at least one usage control element defining a usage of the data object which complies with the predetermined conditions;

- means for decrypting the at least one usage control element and the data object;

- checking means for checking whether a usage requested by the user complies with the usage defined by said at least one usage control element;

- enabling means for enabling the usage requested by the user when the usage complies with the usage defined by said at least one usage control element; and

- disabling means for disabling the usage requested by the user when the usage does not comply with the usage defined by said at least one usage control element.

26. A system as set forth in claim 25, further comprising means for repackaging the data object after usage thereof.

27. A method for controlling the usage by a user of data objects so as to comply with predetermined conditions for usage of the data objects, comprising the steps of:

- storing at least two data packages in a memory device, where they are accessible by a data processor of the user, each said data package comprising a data object

and a user set of control data, which comprises at least one usage control element defining a usage of the data object which complies with the predetermined conditions, the data object and said at least one usage control elements being encrypted;

- decrypting the usage control elements of the user sets of control data;

- examining the usage control elements of said at least two data packages to find a match;

- using, in response to the finding of a match, the data processor to carry out an action, which is specified in the user sets of control data.

28. A method as set forth in claim 27, comprising the further steps of updating the usage control element of each data package, reconcatenating after the usage of the data objects, each of the data object and its usage control element, reencrypting each of the concatenated data objects and its usage control element and transferring the repackaged data objects to their creators.

Fig 1

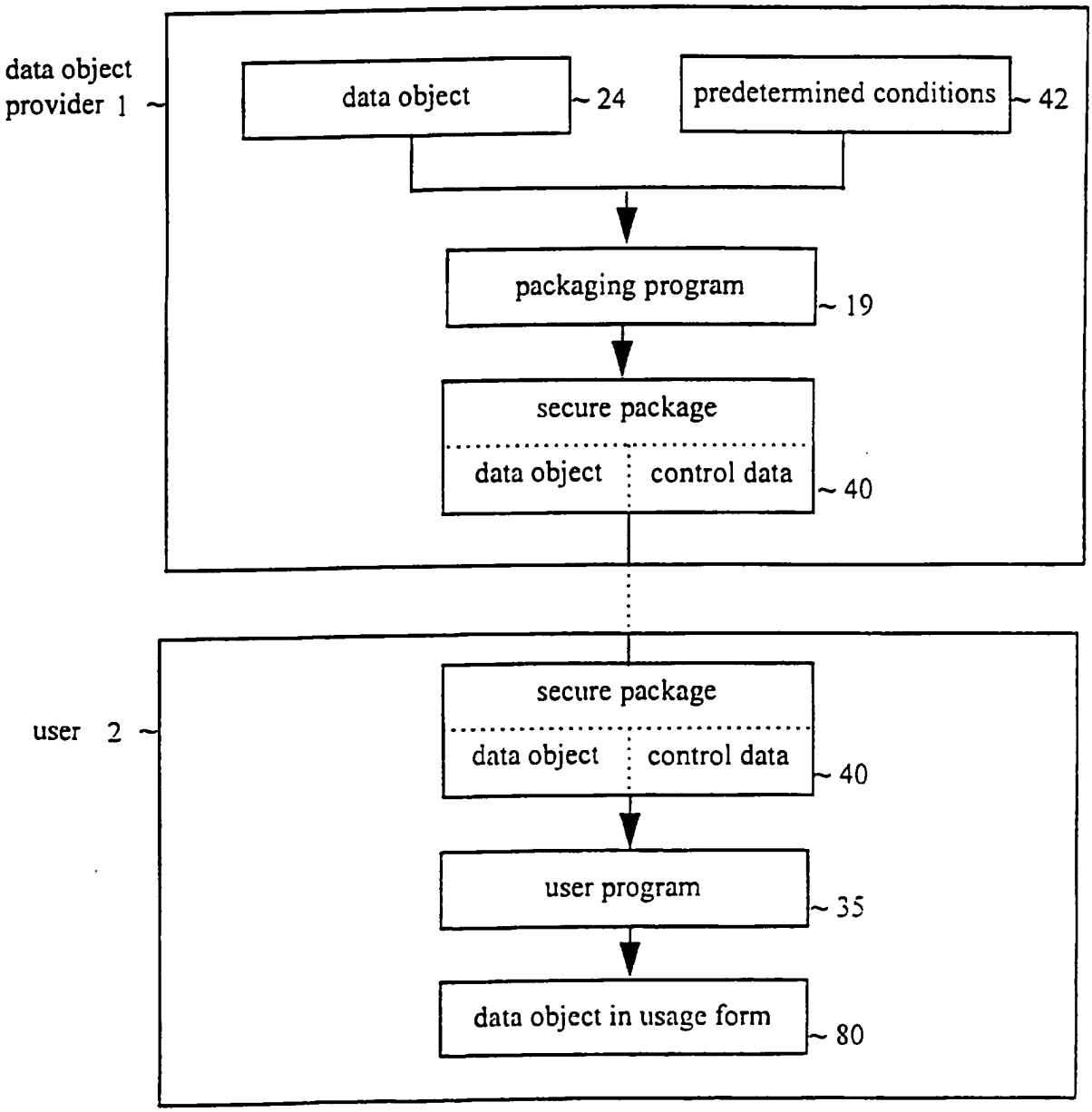


Fig 2

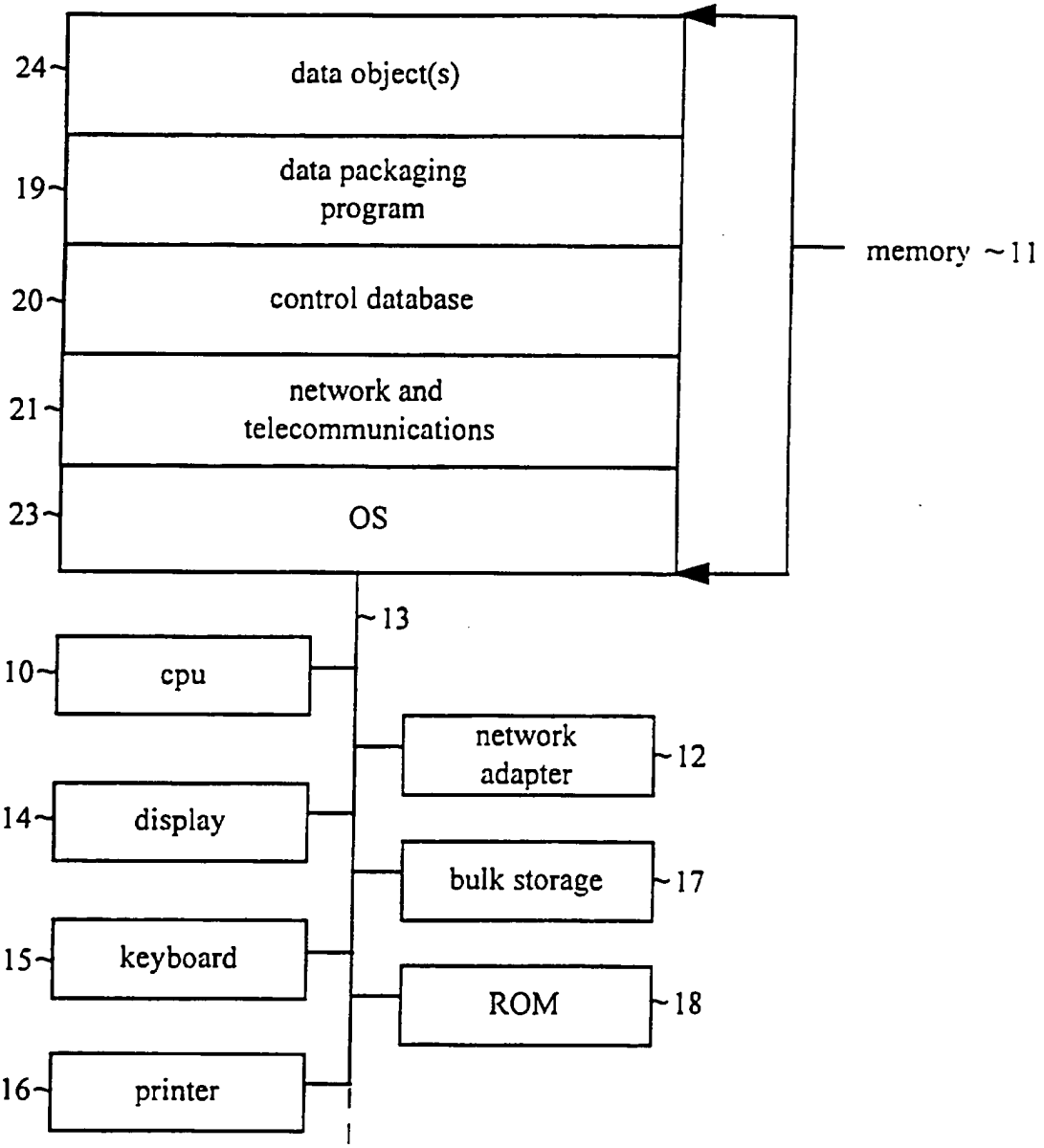


Fig 3

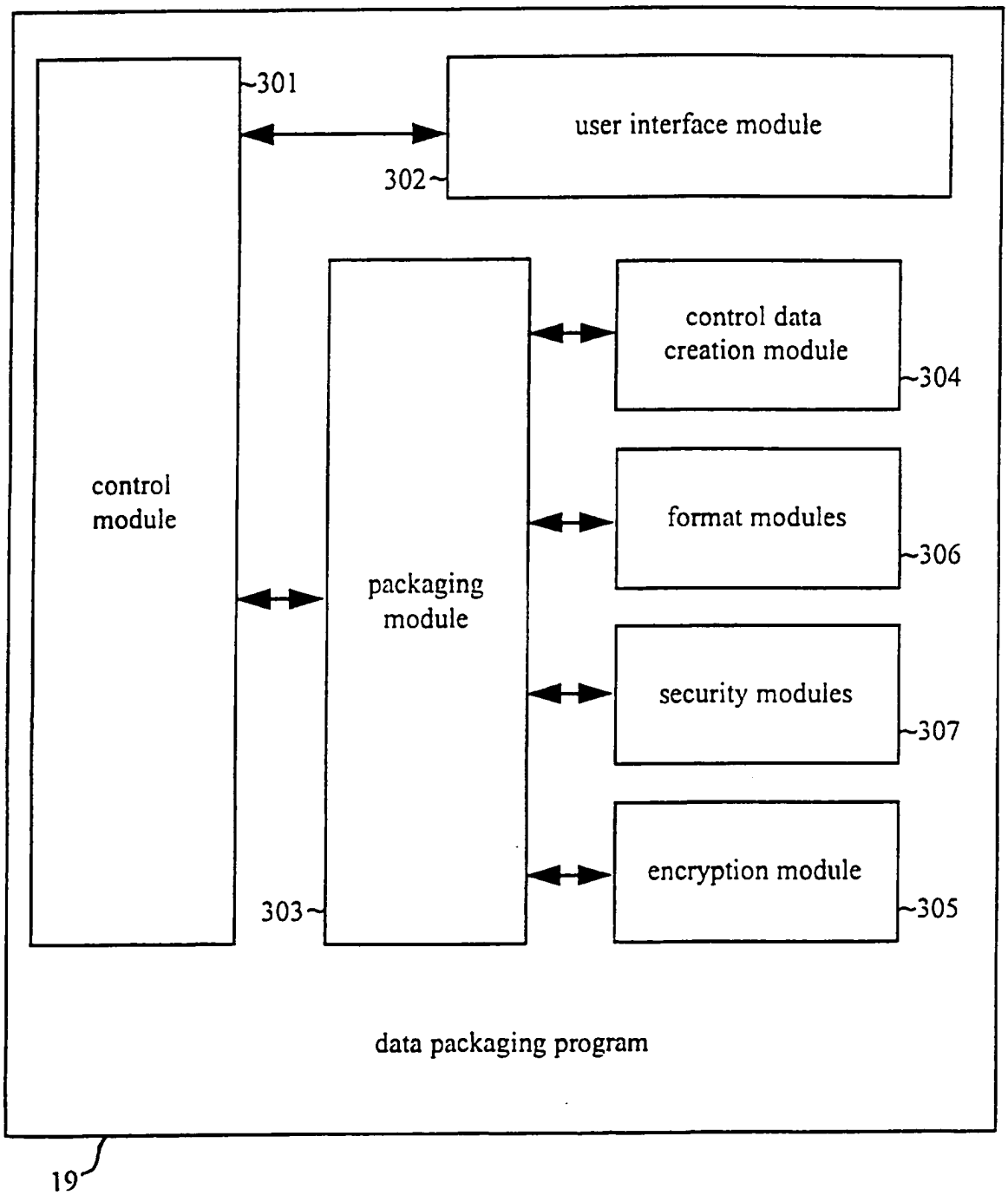


Fig 4

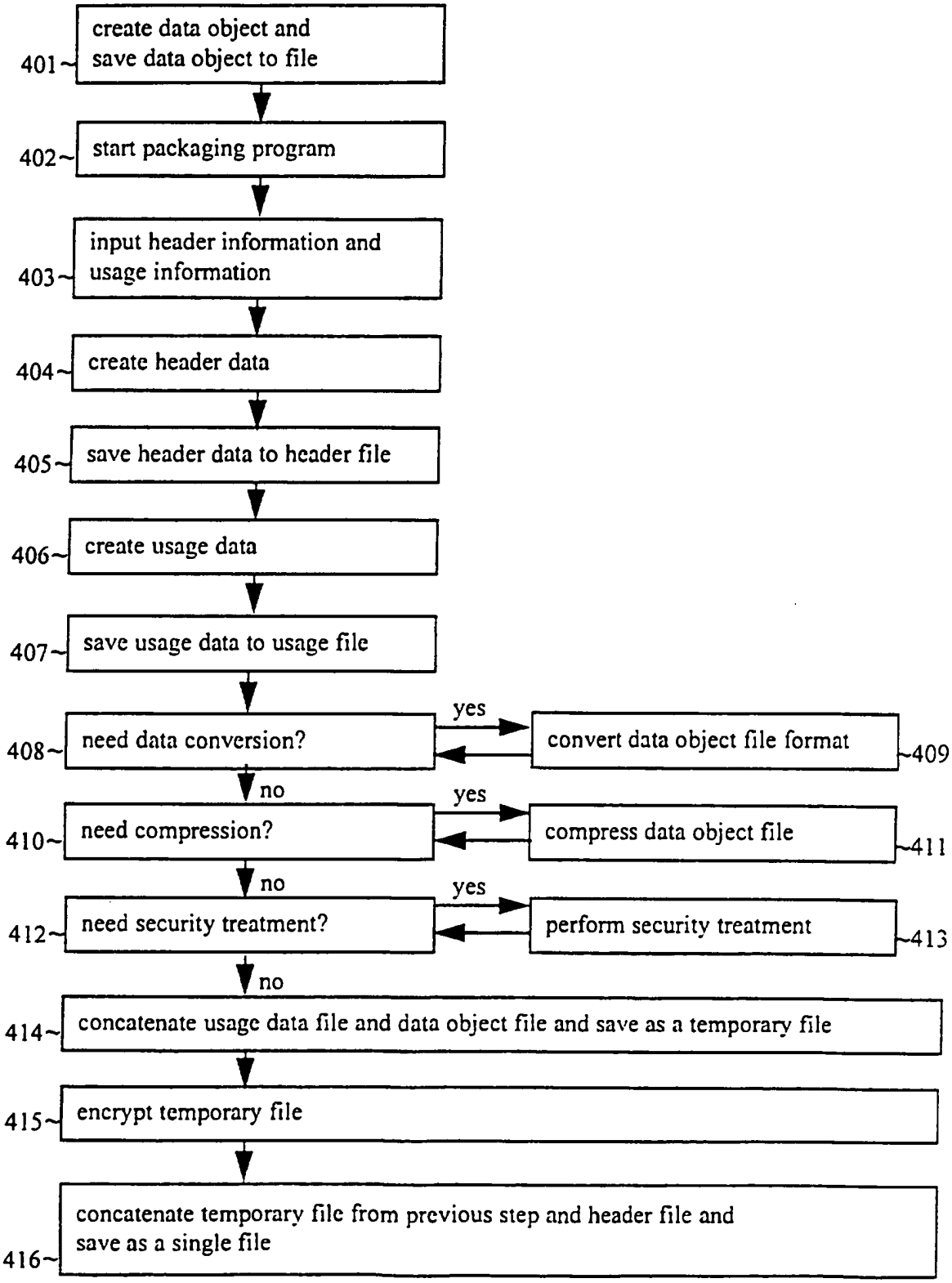


Fig 5

file identifier	123456789
title	image
format code	a
security code	b

Fig 6

usage element for author's phone number	identifer	1
	size	13
	data	716 381 5356
...price for single use	identifer	2
	size	4
	data	.50
...price for unlimited use	identifer	3
	size	4
	data	50.00
...code for usage type approved	identifer	4
	size	2
	data	9
...code for number of usages approved	identifer	5
	size	2
	data	1

Fig 7

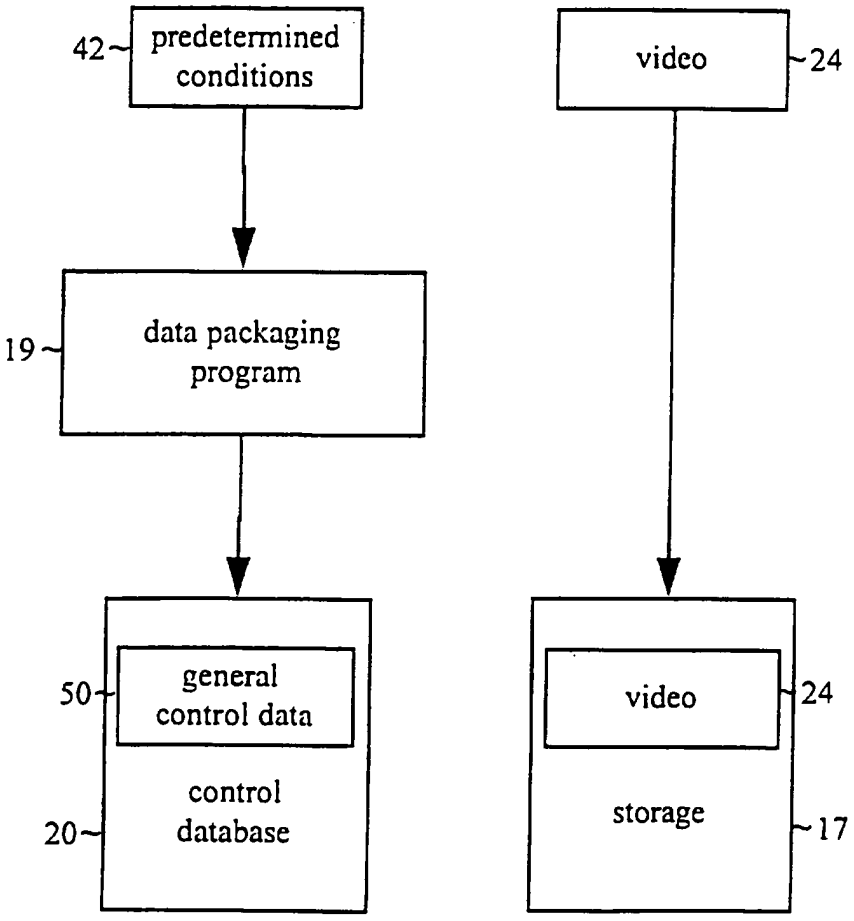


Fig 8a

header	object identifier	123456789
	format code	0010
	security code	0010
	number of usage elements	2
	size of usage data	17
	size of data object	273
	1st usage element id	001
	1st usage element size	6
	1st usage element data	1
	2nd usage element id	002
	2nd usage element size	3
	2nd usage element data	

Fig 8b

header	object identifier	123456790
	format code	0010
	security code	0010
	number of usage elements	2
	size of usage data	17
	size of data object	273
	1st usage element id	001
	1st usage element size	6
	1st usage element data	1
	2nd usage element id	002
	2nd usage element size	3
	2nd usage element data	2

Fig 9

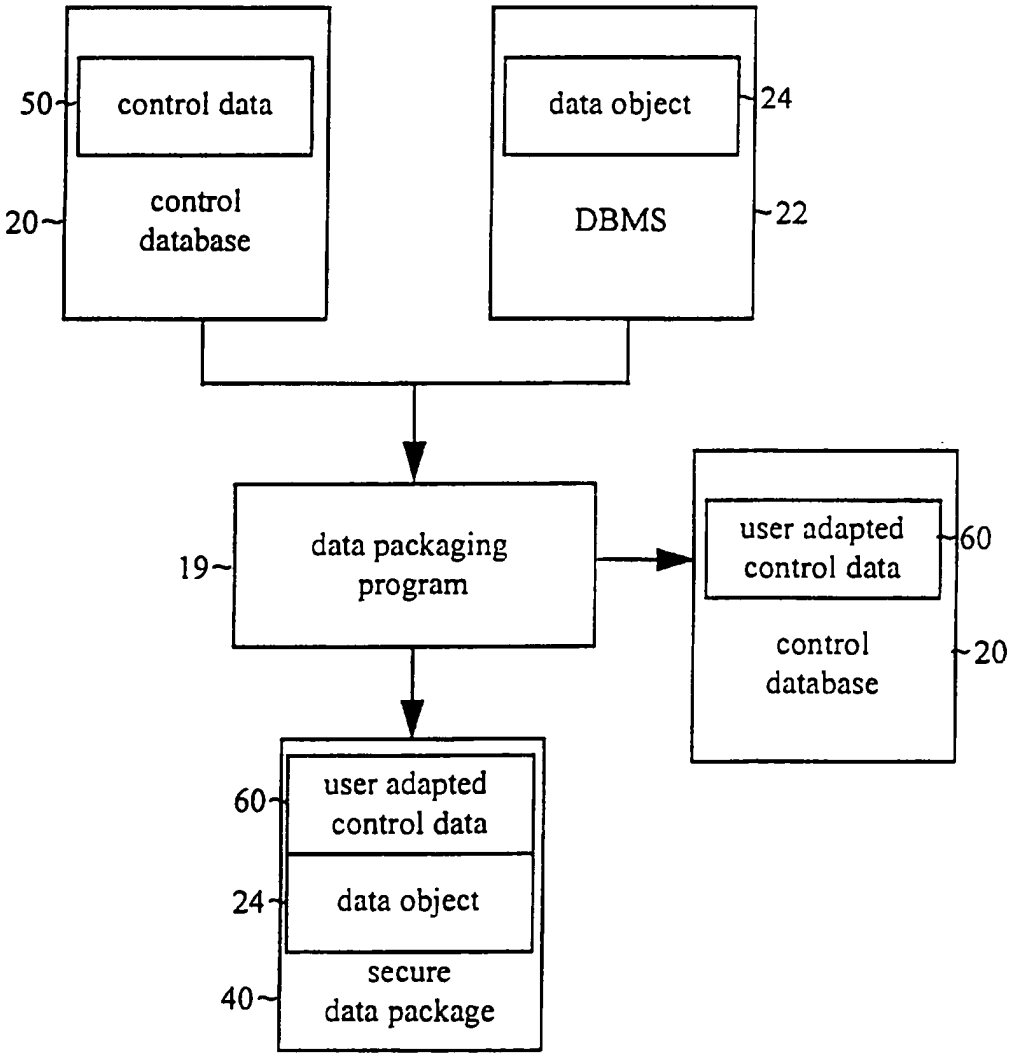


Fig 10

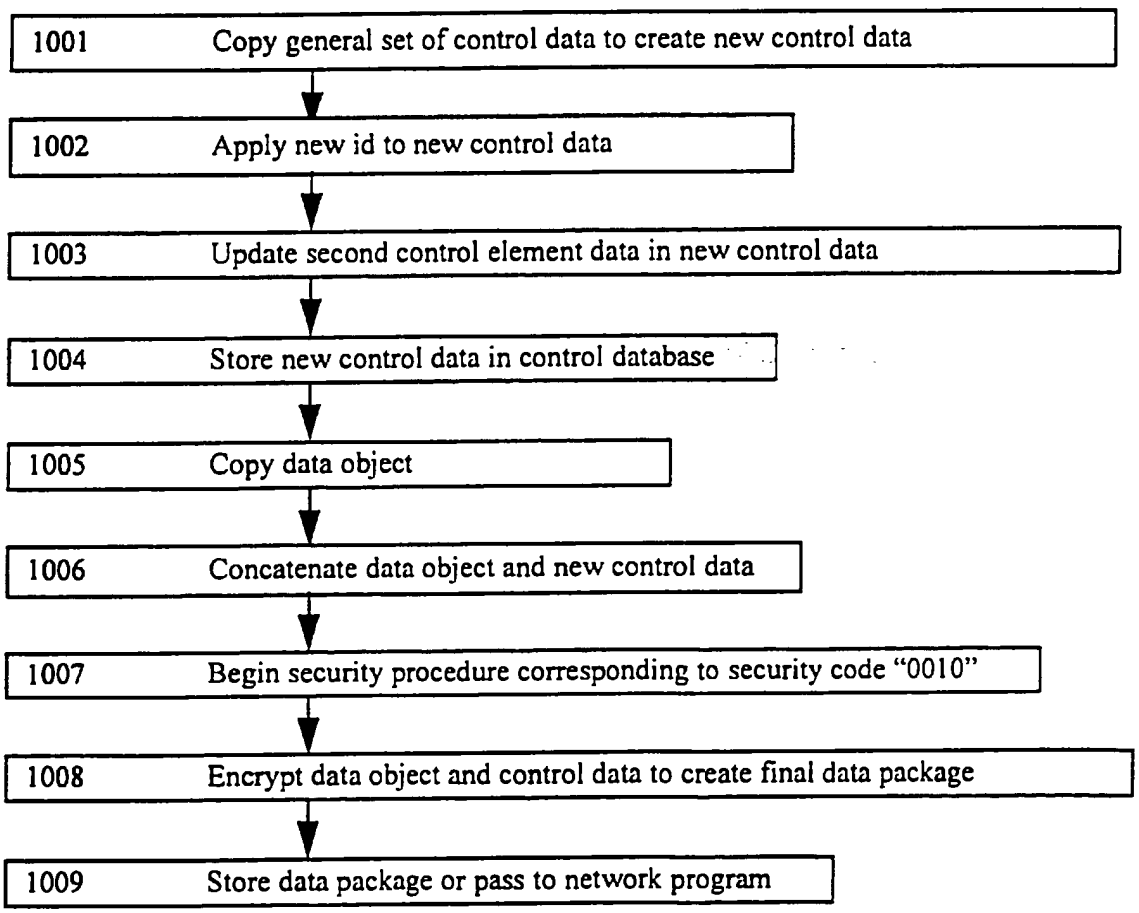
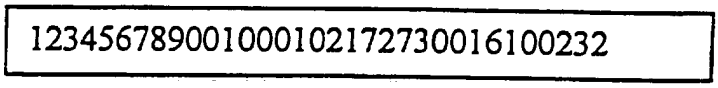


Fig 11

control data in memory



AVI file in memory

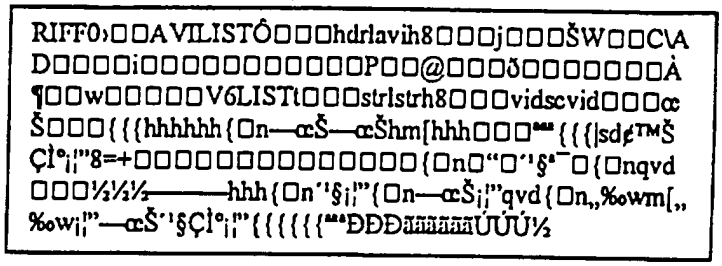


Fig 12a

concatenated control data and AVI file in memory

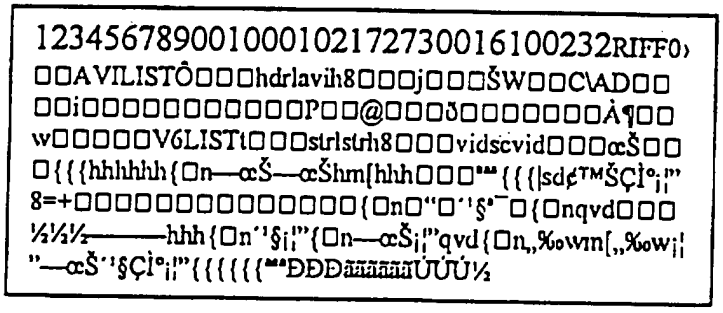


Fig 12b

concatenated and encrypted control data and AVI file in memory

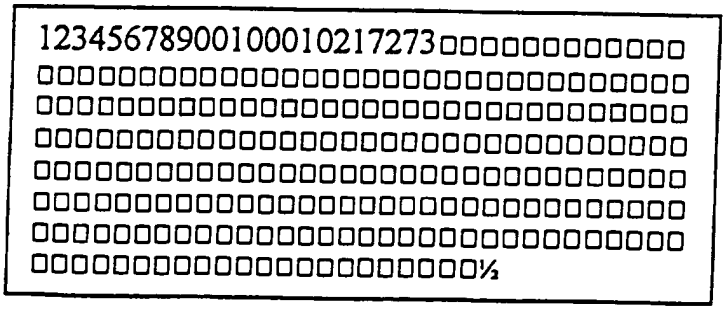


Fig 13

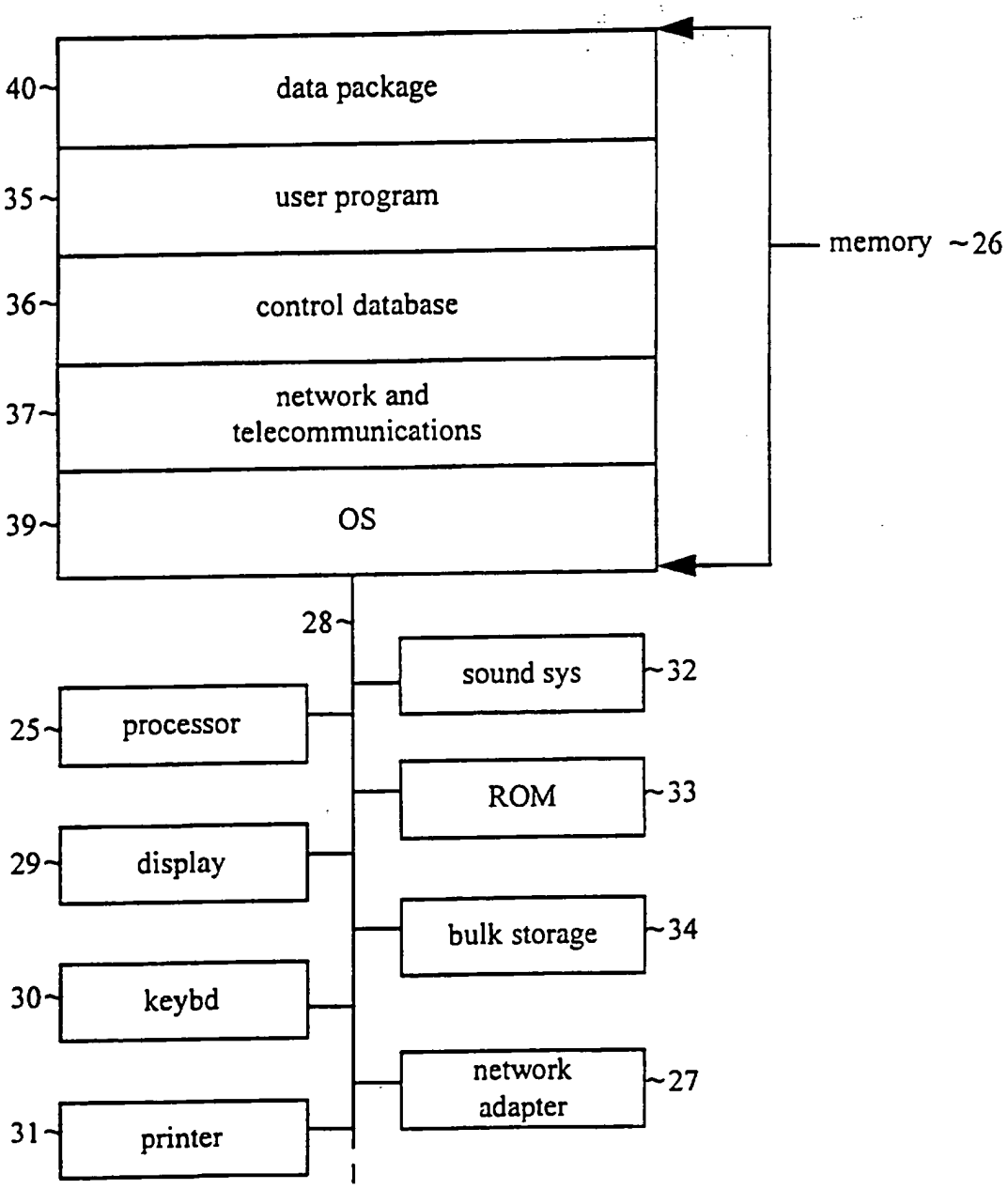


Fig 14

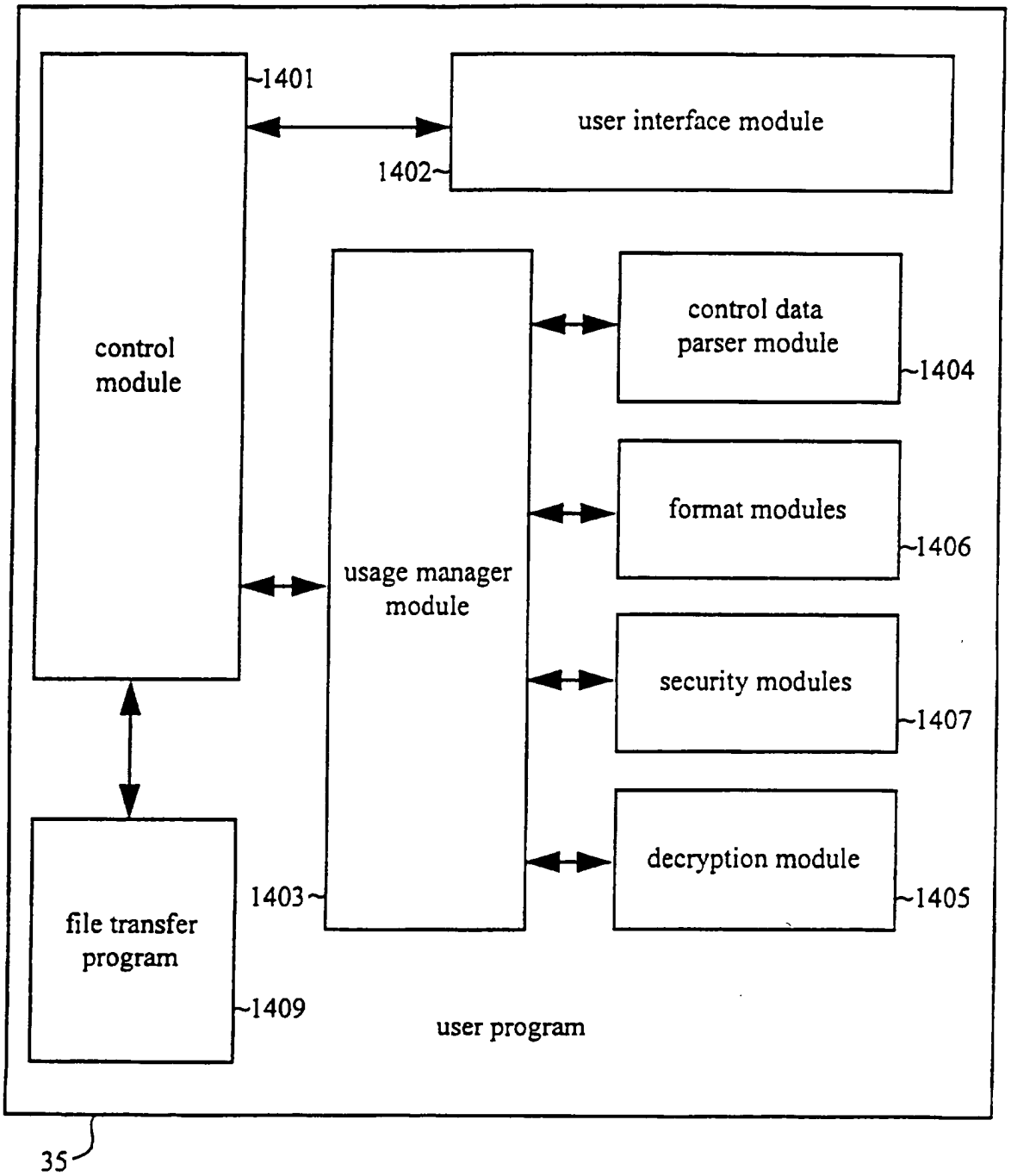


Fig 15

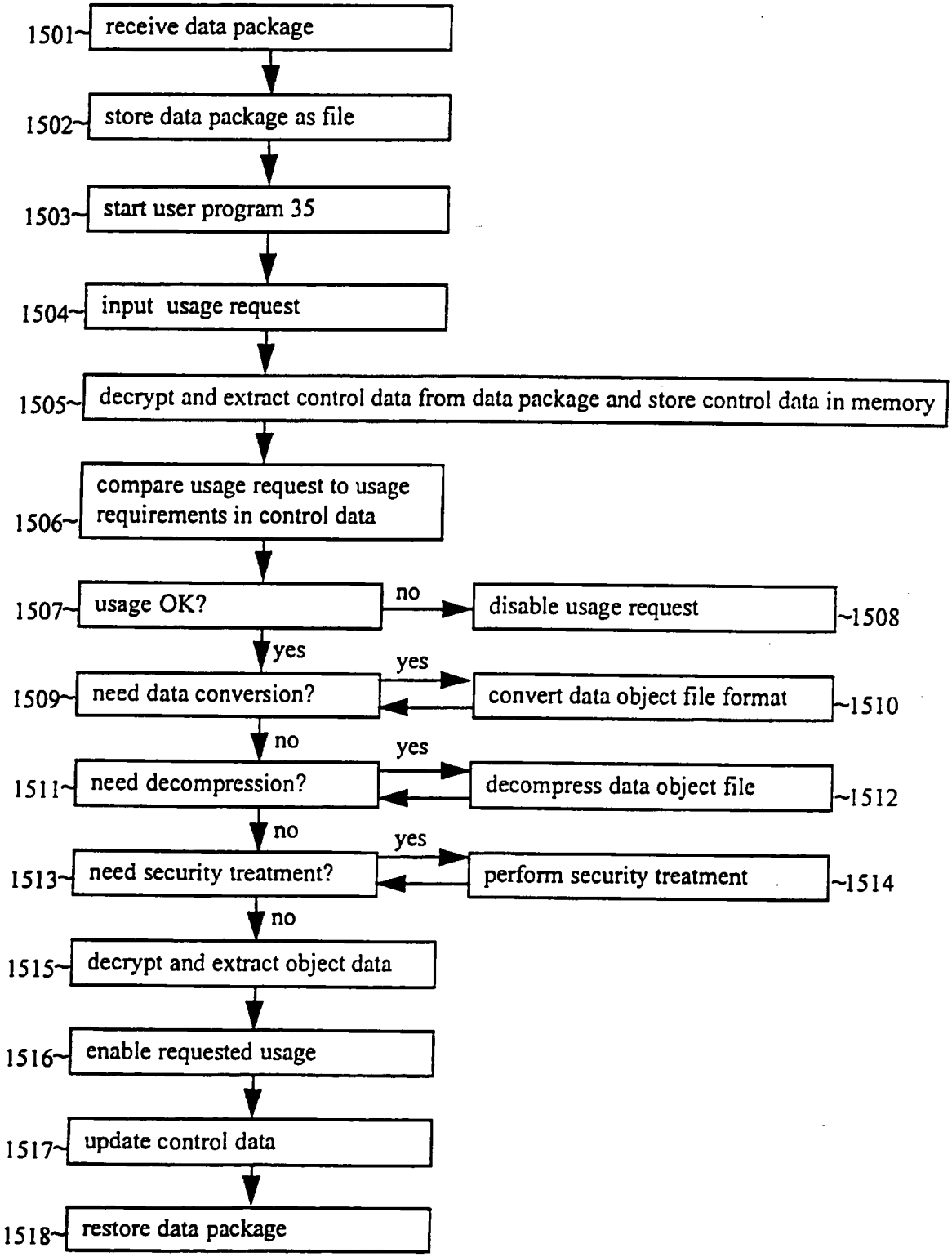
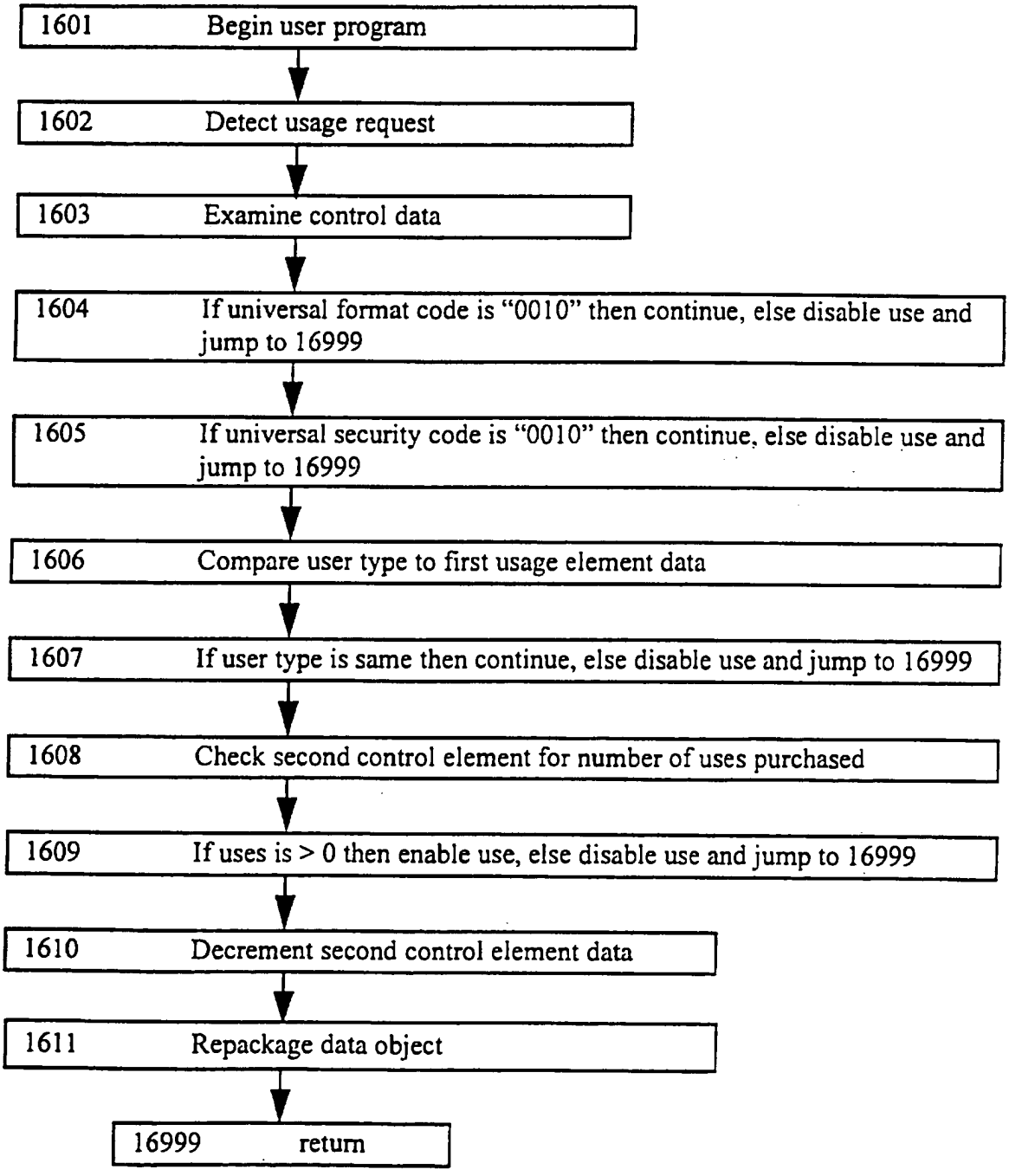
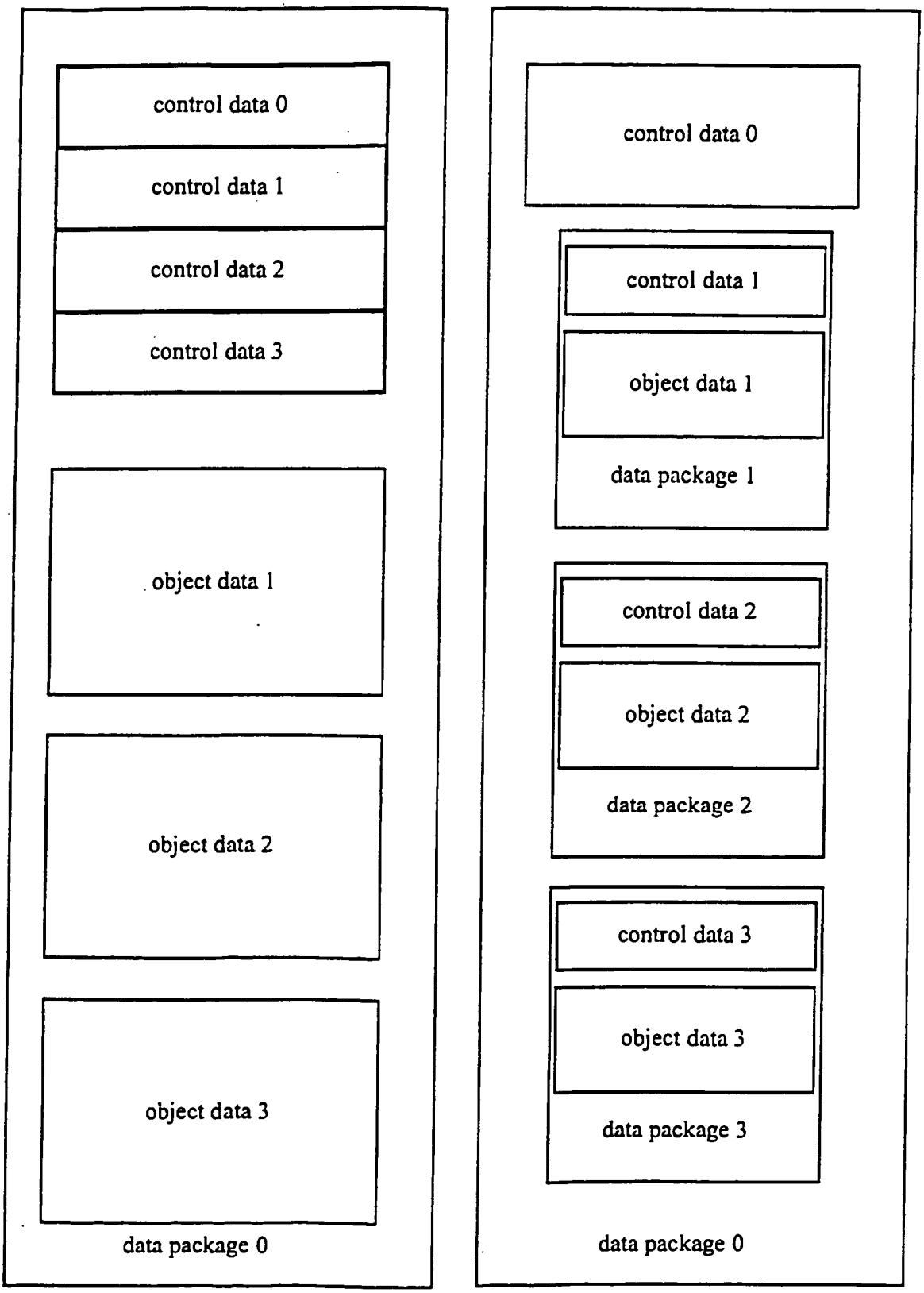


Fig 16



15/15

Fig 17



PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 1/00 — A1

(11) International Publication Number: **WO 92/20022**

(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).

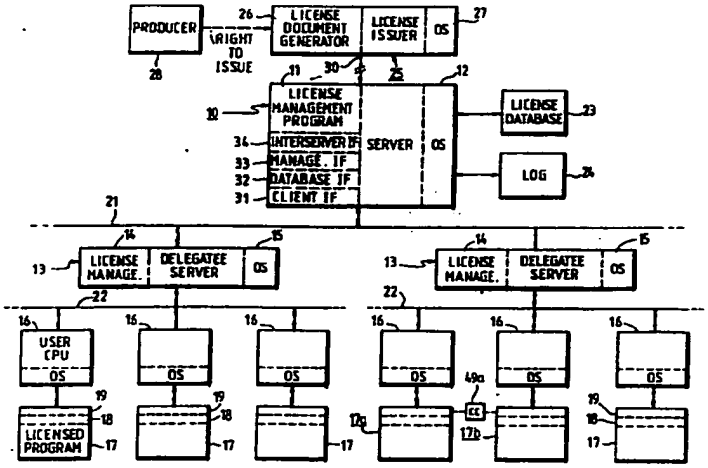
(81) Designated States: AT, AT (European patent), AU, BB, BE (European patent), BF (OAPI patent), BG, BJ (OAPI patent), BR, CA, CF (OAPI patent), CG (OAPI patent), CH, CH (European patent), CI (OAPI patent), CM (OAPI patent), CS, DE, DE (European patent), DK, DK (European patent), ES, ES (European patent), 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	KR	Republic of Korea	SN	Senegal
CI	Côte d'Ivoire	LI	Liechtenstein	SU	Soviet Union
CM	Cameroon	LK	Sri Lanka	TD	Chad
CS	Czechoslovakia	LU	Luxembourg	TG	Togo
DE	Germany	MC	Monaco	US	United States of America
DK	Denmark	MG	Madagascar		
ES	Spain				

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;

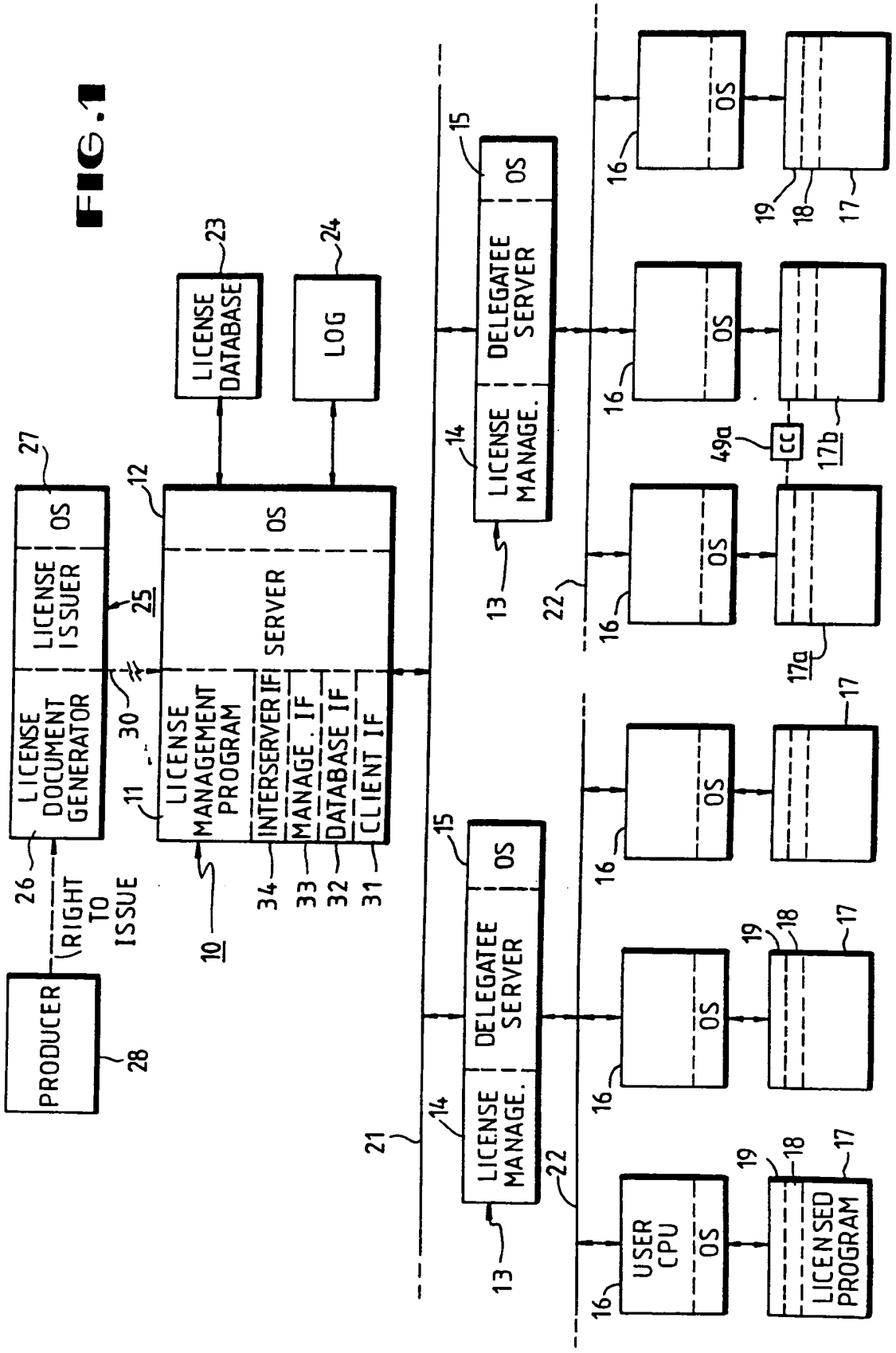
means for accessing said store to obtain information from said license authorization for said software item, in response to said request, and comparing said identification of said user and said software item with said information, to produce a grant or refusal of said request; and

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



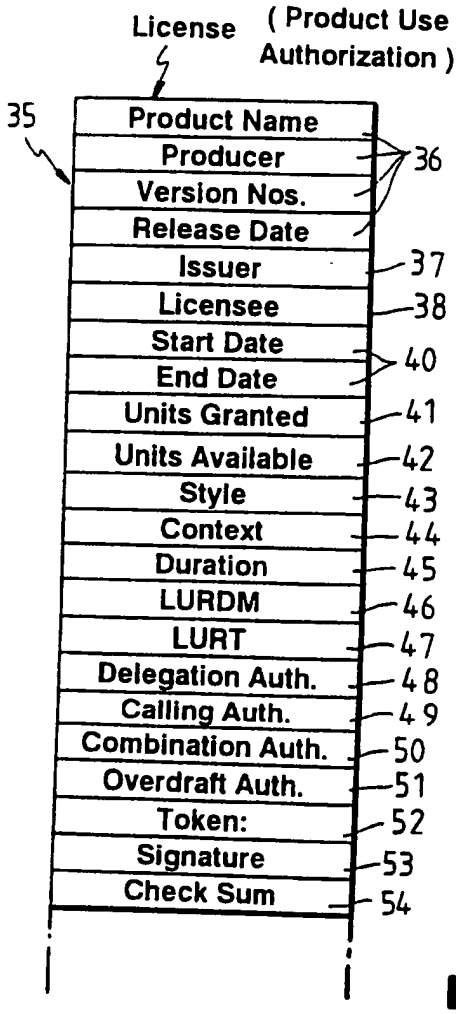


FIG.2

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

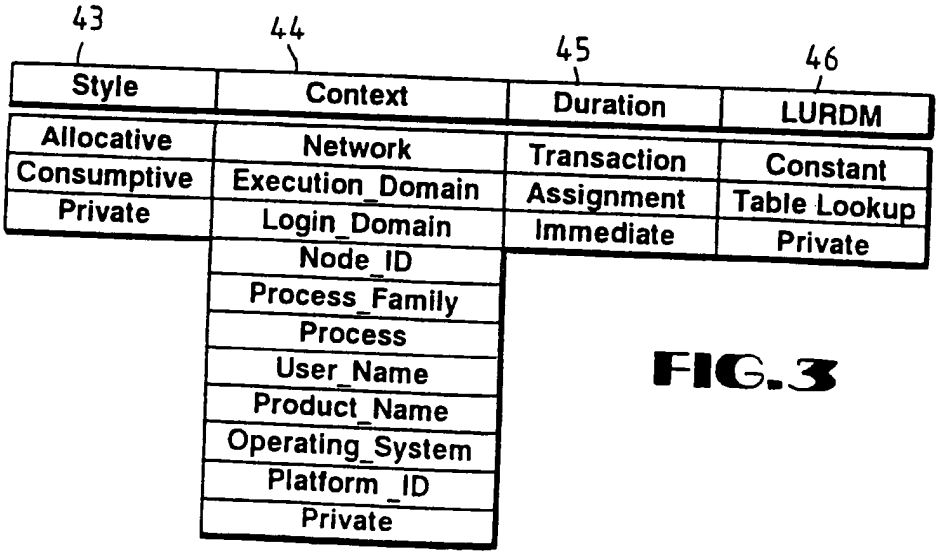


FIG.3

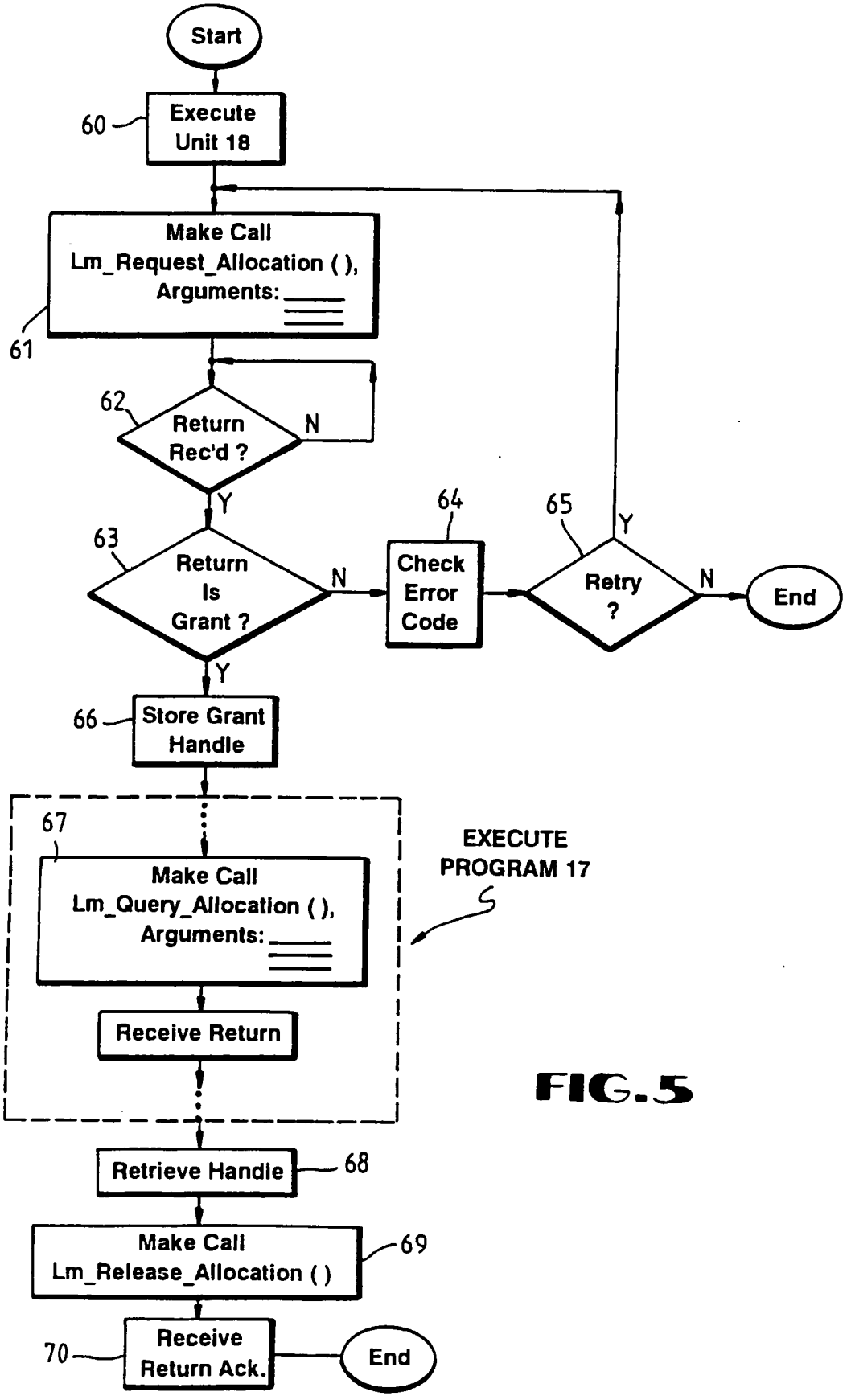


FIG.5

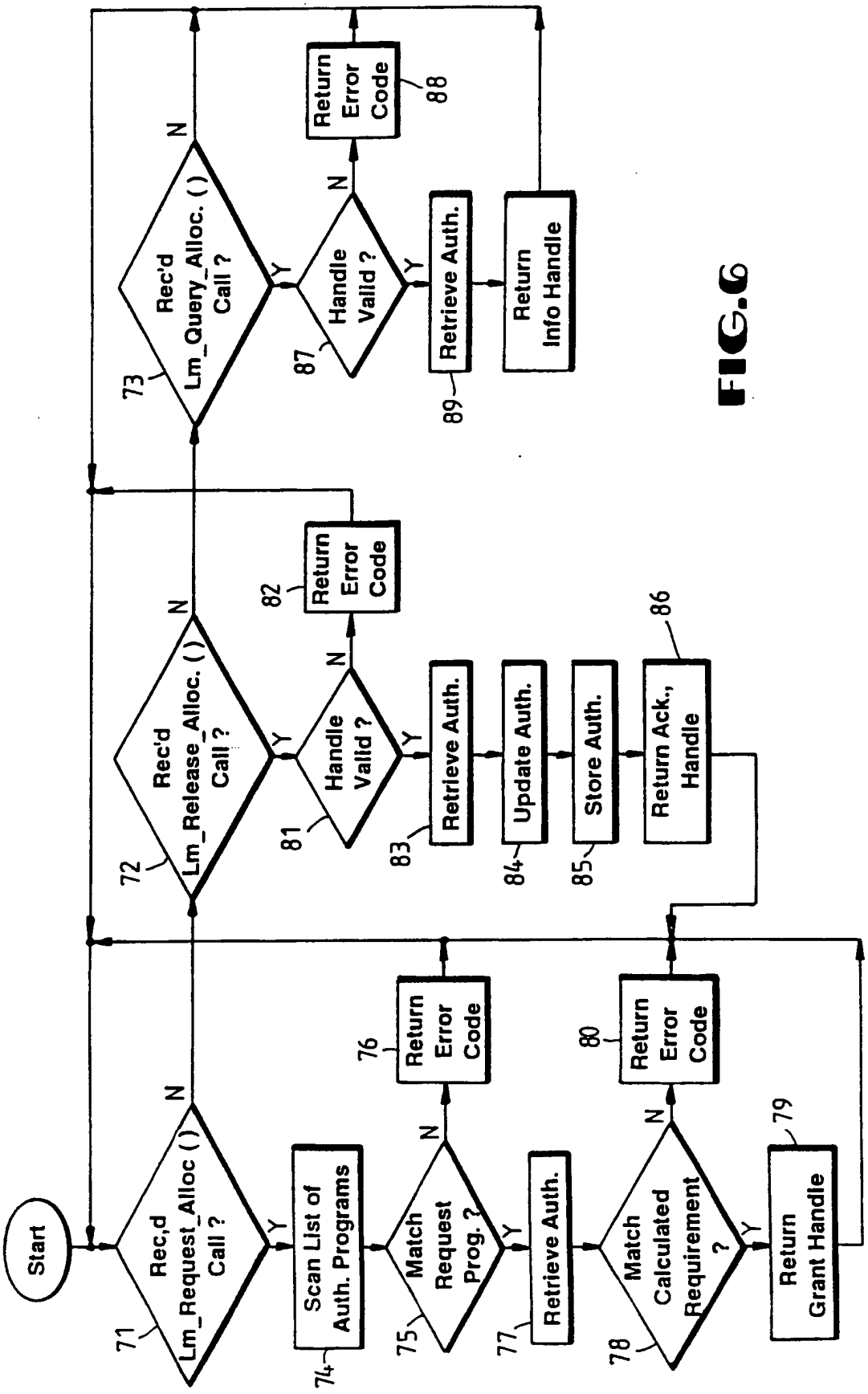


FIG.6

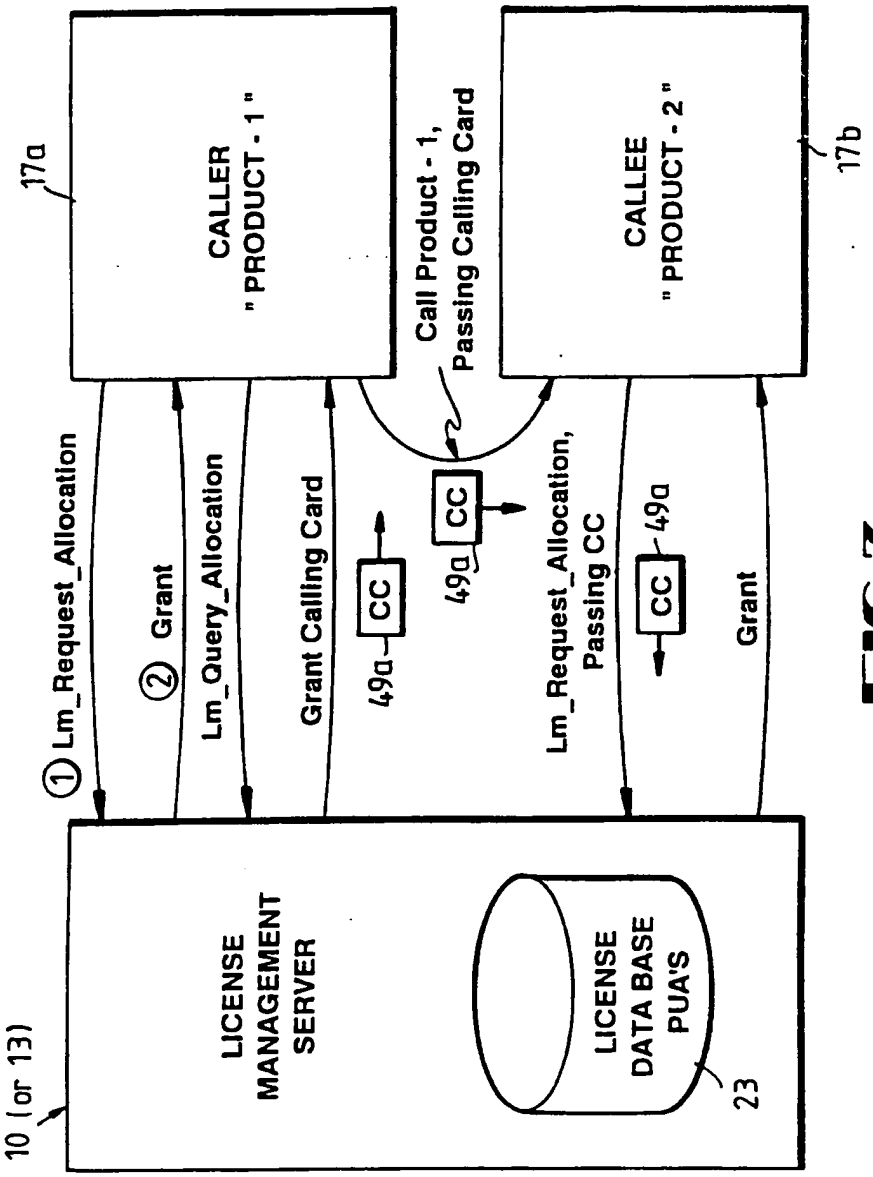


FIG.7

```
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

```
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

```
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

```
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
                        }
```

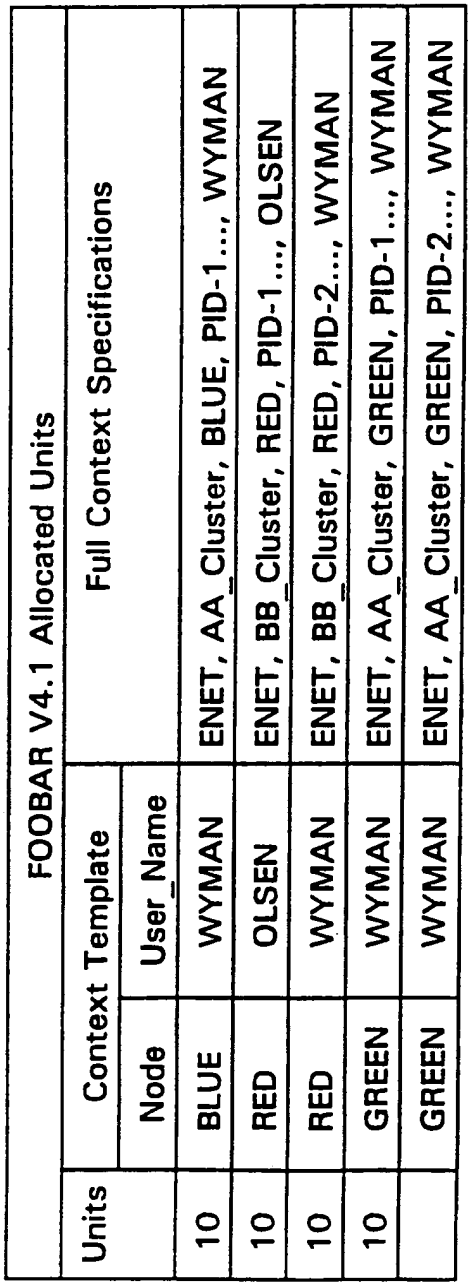
FIG. 27 Assignment Syntax Diagram

```
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	Full Context Specifications
	Node	
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

```
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

```
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

```
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

```
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 °	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

-/--

° 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 6 : H04L — A2

(11) International Publication Number: **WO 97/41661**

(43) International Publication Date: 6 November 1997 (06.11.97)

(21) International Application Number: PCT/US97/06161

(22) International Filing Date: 16 April 1997 (16.04.97)

(30) Priority Data:
08/639,457 29 April 1996 (29.04.96) US

(71) Applicant: MOTOROLA INC. [US/US]; 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

(72) Inventor: DORENBOS, David; 241 N. Larch, Elmhurst, IL 60126 (US).

(74) Agents: LUKASIK, Susan, L. et al.; Motorola Inc., Intellectual Property Dept., 1303 East Algonquin Road, Schaumburg, IL 60196 (US).

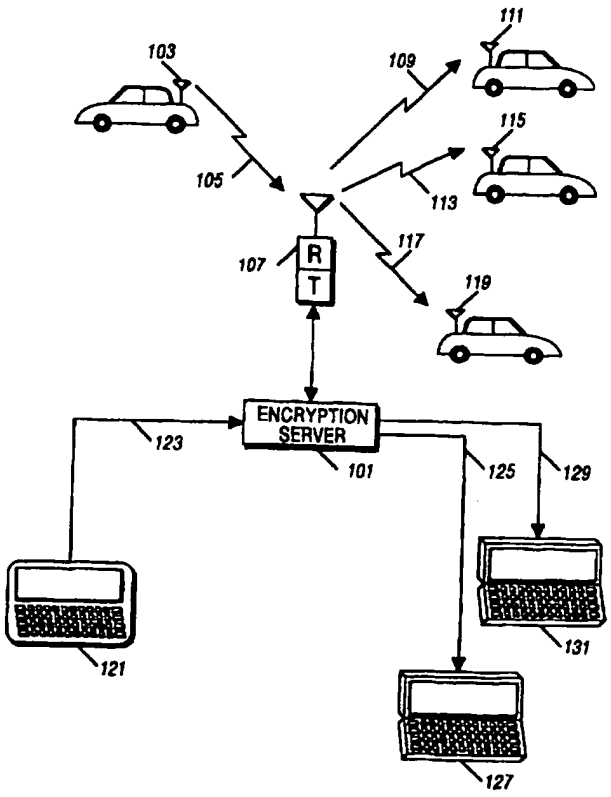
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), European patent (AT, BE, CH, DE, DK, ES, FI, 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
Without international search report and to be republished upon receipt of that report.

(54) Title: USE OF AN ENCRYPTION SERVER FOR ENCRYPTING MESSAGES

(57) Abstract

An encryption server receives a first encrypted message (105) and decrypts (403) the encrypted message using a first key, yielding a decrypted message comprising a second encrypted message (105A), an identification of a sender of the first encrypted message, and an identification of a first recipient. The second encrypted message, the identification of the sender, and the identification of the first recipient are determined (405) from the decrypted message. The second encrypted message and the identification of the sender are encrypted (409) with a second key, yielding a third encrypted message (109). The third encrypted message (109) is transmitted to the first recipient.



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		

USE OF AN ENCRYPTION SERVER FOR ENCRYPTING MESSAGES

Field of the Invention

This invention relates to communication systems, including but not limited to encrypted communication systems.

Background of the Invention

Encrypted voice and data communication systems are well known. Many of these systems provide secure communications between two or more users by sharing one or more pieces of information between the users, which permits only those users knowing that information to properly decrypt the message. This information is known as the encryption key, or key for short. Encryption keys may be private keys, where a single key is utilized for encryption and decryption, or public keys, where multiple keys are utilized for encryption and decryption.

Methods of encrypting using public-key encryption are well known in the art. Typically, a public-key encryption is a method of encryption by which a single message is encrypted using a sender's private key and then a recipient's public key. The recipient then decrypts the message using the recipient's private key and then the sender's public key. Typically, public keys are 512 bits long, although some public keys have as few as 256 bits. Some encryption experts recommend using 1024-bit keys. Because the computational power required to break a key increases exponentially with the length of the key, longer keys provide more security. In addition, because two keys are needed to decrypt a message, two longer keys are more difficult to decrypt if neither key is known.

Today, secure communication systems are used to transmit data in an encrypted fashion. If a user wishes to send the same message to five different recipients, the user must encrypt the message five different times, each time using the public key of a different recipient for the message. The user then transmits the five messages to the five recipients. Such a process, however, is troublesome when the user wishes to transmit to, for example, 100 or more recipients. In this instance, the user must encrypt each message individually 100 or more times, one for each recipient. If the user has a portable communication device, such as a laptop computer, the user's battery may run out of power before encryption and transmission of each message has occurred. In addition, the encryption and transmission process can consume a lot of time and processing power for the portable device, rendering the portable device unavailable for other activities by the user during the encryption and transmission time period. Thus, such transmissions would be impractical for portable users.

Accordingly, there is a need for a method of transmitting encrypted data messages to multiple users without resulting in a time or power barrier to the user's communication device.

Brief Description of the Drawings

FIG. 1 is a block diagram of a communication system having an encryption server in accordance with the invention.

FIG. 2 is a block diagram of an encryption server in accordance with the invention.

FIG. 3 is a flowchart showing a method of transmission of a digital data message to an encryption server in accordance with the invention.

FIG. 4 is a flowchart showing a method of transmission of an encrypted message by an encryption server in accordance with the invention.

Description of a Preferred Embodiment

The following describes an apparatus for and method of using an encryption server for encrypting messages. Messages are encrypted twice, once with the sender's private key and then with an encryption server's public key before transmission of the messages to the encryption server. The encryption server decrypts received messages with the encryption server's private key, yielding an encrypted message, a user identification (ID), and one or more recipient IDs. The encryption server encrypts the encrypted message and the user ID individually with each of the recipient's public keys and transmits the resultant message(s) to the appropriate recipient. Each recipient decrypts the messages using the recipient's private key and the sender's public key. A secure communication system is thereby provided, wherein portable communication devices are neither tied up nor drained of power because the device's user wishes to send a single encrypted message to multiple recipients.

A method of using an encryption server for encrypting messages comprises the steps of, at a communication unit operated by a user generating a digital data message. The digital data message is encrypted using a first key, yielding a first encrypted message. An identification of the user and an identification of a first recipient are appended to the first encrypted message, yielding an appended first encrypted message. The appended first encrypted message is encrypted using a second key, yielding a second encrypted message. The second encrypted message is transmitted to an encryption server. At the encryption server, the second encrypted message is received. The second encrypted message is decrypted using a third key, yielding the appended first encrypted message. The first encrypted message, the identification of the user, and the identification of the first recipient are determined from the appended first encrypted message. The first encrypted message and the identification of the user are encrypted with a fourth key, yielding a third encrypted message. The third encrypted message is transmitted to the first recipient. In the preferred embodiment, the first key is a private key associated with the user, the second key is a public key associated with the encryption server, the third key is a private key associated with the encryption server, and the fourth key

is a public key associated with the first recipient. Alternatively, the second key and the third key may be identical. The transmitting steps may be performed over wireless communication resources, such as radio frequency communication resources, or wireline communication resources, such as standard telephone lines or fiber optic cable.

In addition, the step of appending may further comprise the step of appending an identification of a second recipient to the first encrypted message, thereby yielding the appended first encrypted message. In this case, the method further comprises the steps of encrypting, by the encryption server, the first encrypted message and the identification of the user with a fifth key, yielding a fourth encrypted message, and transmitting the fourth encrypted message to the second recipient. In the preferred embodiment, the fifth key is a public key associated with the second recipient. Alternatively, the step of appending may comprise the step of appending three or more identifications of recipients to the first encrypted message, thereby yielding the appended first encrypted message.

A block diagram of a communication system having an encryption server is shown in FIG. 1. An encryption server 101 is shown at the center of FIG. 1. Further details of the encryption server 101 are shown in FIG. 2 described below. A user of a first communication unit 103 utilizes the first communication unit 103 to generate an digital data message that is encrypted in two stages in the preferred embodiment. In the first stage, the digital data message is encrypted using a first key, which is the user's private key in the preferred embodiment. The result of this encryption is a first-stage encrypted message. (In an alternate embodiment, the digital data message is not encrypted using the first key.) The user's identification (ID) and one or more recipient IDs are appended to the first-stage encrypted message, yielding an appended message. The appended message is encrypted using a second key, yielding a second-stage encrypted message 105. In the preferred embodiment, the second key is the public key associated with the encryption server 101. The communication unit transmits the second-stage encrypted message 105 to the encryption server via a wireless communication link to a wireless communication device 107, such as a radio frequency (RF) base station, repeater, or radio, or infrared

communication device. The second-stage encrypted message 105 is conveyed by the wireless communication device 107 to the encryption server 101.

The encryption server 101 decrypts the second-stage encrypted message 105 using an appropriate key. In the preferred embodiment, the appropriate key is the encryption server's private key. The encryption server 101 then determines the user's ID from the decrypted message and also determines the IDs of all recipients that the user indicated as intended targets of the first-stage encrypted message. The encryption server 101 then encrypts the user's ID along with the first-stage encrypted message by encrypting with the public key of the first recipient. The resultant message 109 is transmitted to the first recipient, who utilizes communication unit 111. The encryption server then encrypts the first-stage encrypted message along with the user's ID by encrypting with the public key of the second recipient and transmitting the resultant encrypted message 113 to the second recipient, who utilizes communication unit 115. This process continues until the encryption server reaches the last recipient ID on the user's list, and encrypts the first-stage encrypted message along with the user's ID by encrypting with the public key of the last recipient and transmitting the resultant encrypted message 117 to the last recipient, who utilizes communication unit 119.

The encryption server 101 may also receive user requests for encryption from wireline communication devices 121 via wireline channels. As with the wireless transmission, the encryption server decrypts the received message 123 using the private key of the encryption server, then encrypts the resultant message individually for each different recipient using the appropriate recipient's individual public key. These recipients may be wireline devices 127 and 131, which receive the messages 125 and 129 via wireline communication channels.

The above examples describe RF to RF transmission and wireline to wireline transmission of encrypted messages. Nevertheless, the method of the present invention is equally successful if a wireline device 121 requests transmission to wireless communication units 111, 115, and 119. Similarly,

a wireless communication unit 103 may request transmission from the encryption server 101 to wireline communication devices 127 and 131. In addition, the recipients may be a combination of both wireless and wireline communication units 111, 115, 119, 127, and 131, regardless of whether the sender uses a wireless communication unit 103 or a wireline communication device 121.

Upon receipt of the encrypted message from the encryption server, each recipient decrypts the message with the recipient's own private key, and after determining the user's ID, decrypts the resultant message with the user's public key, thereby yielding the original digital data message. The user is also referred to as the sender of the (second-stage) encrypted message 105.

A block diagram of an encryption server 101, including its input signals 105 and output signals 109, 113, 125, and 117, is shown in FIG. 2. In the preferred embodiment, the encryption server 101 is a Sun SparcServer2000 in a multiprocessor configuration, available from Sun Microsystems. The encryption server 101 comprises one or more processors 201, such as microprocessors or digital signal processors, as are well known in the art. The processors 201 have access to encryption and decryption algorithm(s) 203, a public key data base 205, and memory 211. The encryption/decryption algorithms 203 include public key algorithms, private algorithms, and other algorithms as may be used in the art. The public key data base 205 includes a list of IDs, as used by senders (users) and recipients, and the public keys associated with each of these IDs. The memory 211 includes programming and other data as is necessary to provide functionality as described herein for the encryption server 101. A receive block for wireline and wireless communications 207 and a transmit block for wireline and wireless communications 209 are also connected to the processors 201. The receive block for wireline and wireless communications 207 performs appropriate demodulation techniques on received messages 105 and 123. The transmit block for wireline and wireless communications 209 performs appropriate modulation techniques on messages 109, 113, 124, and 117 to be transmitted. In addition, the encryption server 101 may be equipped with hardware

and/or software to provide the encryption server 101 with over-the-air-rekeying capabilities.

As shown in FIG. 2, a user message 105 comprises a second-stage encrypted (encrypted using the encryption server's public key) message comprising the digital data message 105A, first-stage encrypted with the user's (sender's) private key, in addition to the user ID and a number of recipient IDs. Alternatively, the user message 105 may comprise an unencrypted digital data message 105A, the user ID, and one or more recipient IDs. The user message 105 is input to the receive wireline/wireless block 207, the output of which is input to the processor(s) 201. The processor(s) 201 utilize(s) the encryption/decryption algorithm(s) 203 and the public key data base 205 to decrypt the message 105 using the private key of the encryption server. The processor(s) 201 then determine(s) the first-stage encrypted message 105A, the user ID, and the first recipient ID from the decrypted message. The processor(s) 201 then determine(s) the first recipient's public key from public key data base 205, and the encrypt the first-stage encrypted message 105A and the user ID by using the encryption/decryption algorithms 203 and the first recipient's public key. The processor(s) 201 then append(s) the first recipient ID, thereby yielding a message 109 that is sent to the transmit wireline/wireless block 209 for transmitting to the first recipient's communication unit 111, as shown in FIG. 1. A similar process is performed on the first-stage encrypted message (or unencrypted digital data message) 105A and the user ID for each of the recipients listed in the user's message 105.

In an alternate embodiment, the encryption server 101 may be physically distributed as one or more encryption servers. In this embodiment, the encryption server 101 encrypts the message using a second set of private and public keys associated with a second server. The message so encrypted is transmitted to the second encryption server. The second server decrypts the message and then encrypts the message using the public key(s) of the recipient(s). When traffic is heavy, the encryption server 101 may optimize its efficiency by determining the computation required to transmit directly to each recipient or transmit the request to one or more distributed servers. This process is transparent to the user.

The flowchart of FIG. 3 shows a method for use by a communication unit in transmitting a digital data message to an encryption server 101. At step 301, a digital data message is generated. If at step 303 the digital data message is not to be encrypted, the process continues with step 307. If at step 303 the digital data message is to be encrypted, the process continues with step 305, where the digital data message is encrypted using the private-key of the user who wishes to communicate the message. At step 307, it is determined if the IDs of the user and/or recipient(s) are to be encrypted. If the IDs are to be encrypted, the process continues with step 309, where the user ID and recipient ID(s) are appended to the encrypted message from step 305 or the unencrypted message from step 301 if no encryption took place. At step 311, the message from step 309, including the appended IDs, is encrypted using the public key of the encryption server 101. The process continues with step 317, where the encrypted message is transmitted to the encryption server 101. If at step 307 the IDs are not to be encrypted, the process continues with step 313, where the encrypted message of step 305 (or the unencrypted message from step 301 if no encryption took place) is encrypted with the public key of the encryption server 101. At step 315, the user ID and recipient ID(s) are appended to the encrypted message of step 313, and the process continues with step 317.

In an alternative embodiment, i.e., when the digital data message is not to be encrypted at step 303 of FIG. 3, the sender or user may decrypt the digital data message and, if desired, the recipient IDs only once, using the encryption server's public key. The encryption server then decrypts the message using the encryption server's private key, and encrypts the message individually for each of the recipients with the recipient's public key. The recipient then decrypts the message using only the recipient's private key. This method requires the user to locally store only one public key, the key of the encryption server. With this method, a single symmetrical key may be used to encrypt and decrypt the messages between the user and the encryption server 101, and one or more keys may be used to encrypt the messages between the encryption server and the recipient. Nevertheless, for better security, the encryption server 101 engaged in this embodiment should be a physically secured, e.g., locked away with limited

access, because unencrypted information is present inside the encryption server 101. An advantage of such a system includes enabling law enforcement officials the ability to read the decrypted message as available in the encryption server 101.

The flowchart of FIG. 4 shows the method performed by the encryption server 101 in accordance with the present invention. At step 401, the encryption server receives the encrypted message transmitted by the communication unit 103. At step 403, the encryption server decrypts the message received at step 401 with the private key of the encryption server 101. At step 405, the encryption server determines the user ID, the recipient ID(s), and the encrypted (generated at step 305 of FIG. 3) or unencrypted (generated at step 301 of FIG. 3) data message. In an alternate embodiment, the encryption server 101 may be equipped with the appropriate keys to decrypt the digital data message 105A (when the message 105A is encrypted) so that law enforcement agencies may have full access to all information transmitted in the system.

At step 407, it is determined if the IDs (i.e., the user ID and/or recipient ID(s)) are to be encrypted before transmission. If the IDs are to be encrypted, the process continues with step 409, where the encryption server encrypts the encrypted data message along with the user ID, and the recipient's ID if desired, with the recipient's public key. At step 411, the encryption server transmits the encrypted message to the recipient whose public key was used at step 409. If at step 413 there are more recipients identified by the user to which the encryption server has not yet encrypted and transmitted the message, the process continues with step 407. If there are no more recipients at step 413, the process ends. If at step 407, the IDs are not to be encrypted, the process continues with step 415, where the encrypted data message is encrypted with the recipient's public key, and the user ID and the recipient's ID are appended to that encrypted message without further encryption, and the process continues with step 411.

Optionally, all messages may be encrypted at one time, and then transmitted in succession at one time, rather than encrypting a first message with one public key, then transmitting the encrypted first message

right away, then encrypting a second message using another public key, and transmitting the encrypted second message immediately, and so forth.

The above text and associated drawings describe a method using public-key encryption. Private-key encryption, where the same key is used to encrypt and decrypt a message, may also be used. For example, the key used to encrypt the message send to the encryption server may be the same or identical key used to decrypt the encrypted message at the encryption server. In addition, the encryption method employed by the user to encrypt the original digital data message 105A may also be private-key encryption, rather than public-key encryption. In addition, a different encryption algorithm may be utilized for the user's first stage of encryption than for the user's second stage of encryption, the result of which is transmitted to the encryption server.

In the above manner, the encryption server encrypts the user's data message individually for each different recipient using that particular recipient's public key. The encryption server has more computing resources available to it than an individual communication unit, and can encrypt and transmit a message multiple times to many different users in a more efficient manner than can an individual communication unit. Individual communication units need not store all possible recipient's public keys, but instead need store only the encryption server's public key. Encryption of the recipient's ID(s) helps to secure the identity of the recipient(s) and eliminates a source of information for traffic analysis by undesired readers/interceptors of such information.

What is claimed is:

Claims

1. A method comprising the steps of:

at a communication unit operated by a user:

generating a digital data message;

encrypting the digital data message using a first key, yielding a first encrypted message;

appending an identification of the user and an identification of a first recipient to the first encrypted message, yielding an appended first encrypted message;

encrypting the appended first encrypted message using a second key, yielding a second encrypted message;

transmitting the second encrypted message to the encryption server, wherein the encryption server is not the first recipient.

2. The method of claim 1, wherein the first key is a private key associated with the user and wherein the second key is a public key associated with the encryption server.

3. The method of claim 1, wherein the step of appending further comprises the step of appending an identification of a second recipient to the first encrypted message, thereby yielding the appended first encrypted message.

4. A method comprising the steps of:

at an encryption server:

receiving a first encrypted message;

decrypting the encrypted message using a first key, yielding a decrypted message comprising a second encrypted message, an identification of a sender of the first encrypted message, and an identification of a first recipient;

determining the second encrypted message, the identification of the sender, and the identification of the first recipient from the decrypted message;

encrypting the second encrypted message and the identification of the sender with a second key, yielding a third encrypted message;

transmitting the third encrypted message to the first recipient.

5. The method of claim 4, wherein the first key is a private key associated with the encryption server and wherein the second key is a public key associated with the first recipient.

6. The method of claim 4, further comprising, when a second identification of a second recipient is part of the decrypted message, the steps of encrypting, by the encryption server, the second encrypted message and the identification of the sender with a third key, yielding a fourth encrypted message, and transmitting the fourth encrypted message to the second recipient.

7. A method comprising the steps of:

at a communication unit operated by a user:

generating a digital data message;

encrypting the digital data message using a first key, yielding a first encrypted message;

encrypting the first encrypted message using a second key, yielding a second encrypted message;

appending an identification of the user and an identification of a first recipient to the second encrypted message, yielding an appended second encrypted message;

transmitting the appended second encrypted message to the encryption server;

at the encryption server:

receiving the appended second encrypted message;

determining the second encrypted message, the identification of the user, and the identification of the first recipient from the appended second encrypted message;

decrypting the second encrypted message using a third key, yielding the first encrypted message;

encrypting the first encrypted message with a fourth key, yielding a third encrypted message;

transmitting the third encrypted message to the first recipient.

8. The method of claim 7, wherein the step of appending further comprises the step of appending an identification of a second recipient to the second encrypted message, thereby yielding the appended second encrypted message.

9. The method of claim 7, wherein the first key is a private key associated with the user, wherein the second key is a public key associated with the encryption server, wherein the third key is a private key associated with the encryption server, and wherein the fourth key is a public key associated with the first recipient.

10. The method of claim 7, wherein the identification of the user is encrypted using the second key before the step of appending.

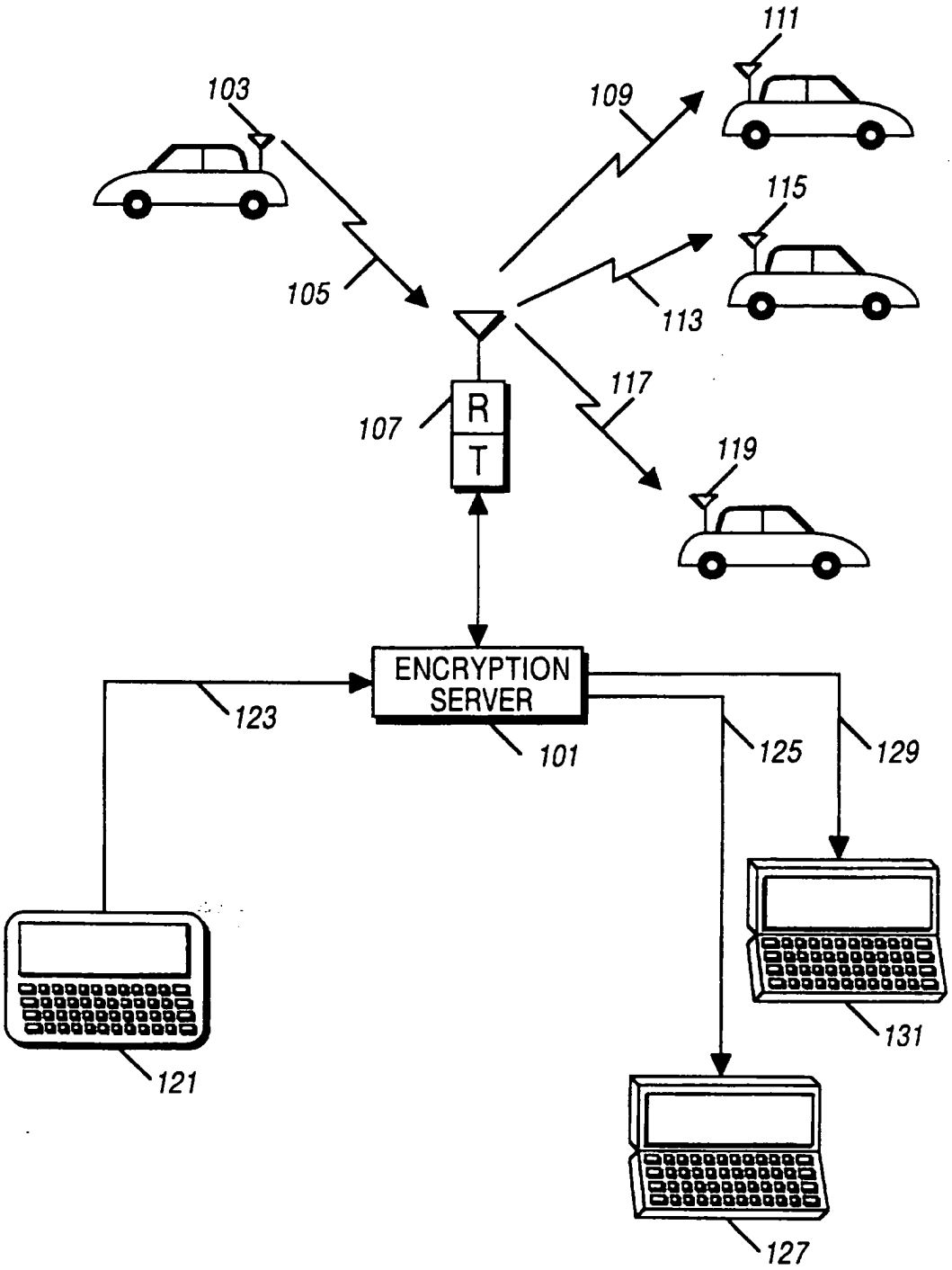

111
103
109
115
105
113
R
T
107
117
119
ENCRYPTION SERVER
123
101
125
129
121
131
127

FIG. 1

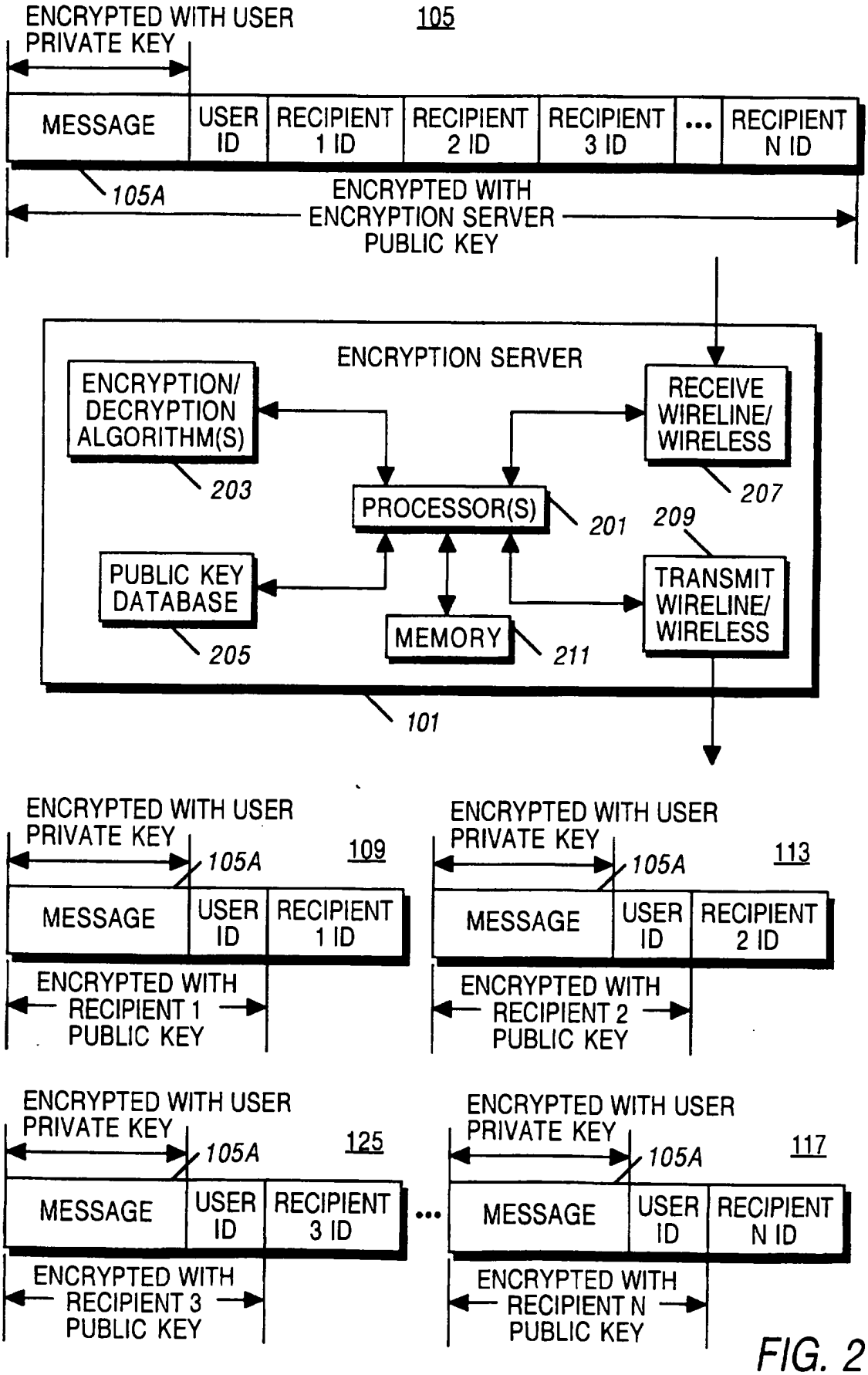

ENCRYPTED WITH USER PRIVATE KEY
105
MESSAGE
USER ID
RECIPIENT 1 ID
RECIPIENT 2 ID
RECIPIENT 3 ID
...
RECIPIENT N ID
105A
ENCRYPTED WITH ENCRYPTION SERVER PUBLIC KEY
ENCRYPTION SERVER
ENCRYPTION/ DECRYPTION ALGORITHM(S)
203
PROCESSOR(S)
201
RECEIVE WIRELINE/ WIRELESS
207
209
PUBLIC KEY DATABASE
205
MEMORY
211
TRANSMIT WIRELINE/ WIRELESS
101
ENCRYPTED WITH USER PRIVATE KEY
105A
109
MESSAGE
USER ID
RECIPIENT 1 ID
ENCRYPTED WITH RECIPIENT 1 PUBLIC KEY
ENCRYPTED WITH USER PRIVATE KEY
105A
113
MESSAGE
USER ID
RECIPIENT 2 ID
ENCRYPTED WITH RECIPIENT 2 PUBLIC KEY
ENCRYPTED WITH USER PRIVATE KEY
105A
125
MESSAGE
USER ID
RECIPIENT 3 ID
ENCRYPTED WITH RECIPIENT 3 PUBLIC KEY
...
ENCRYPTED WITH USER PRIVATE KEY
105A
117
MESSAGE
USER ID
RECIPIENT N ID
ENCRYPTED WITH RECIPIENT N PUBLIC KEY

FIG. 2

FIG. 3

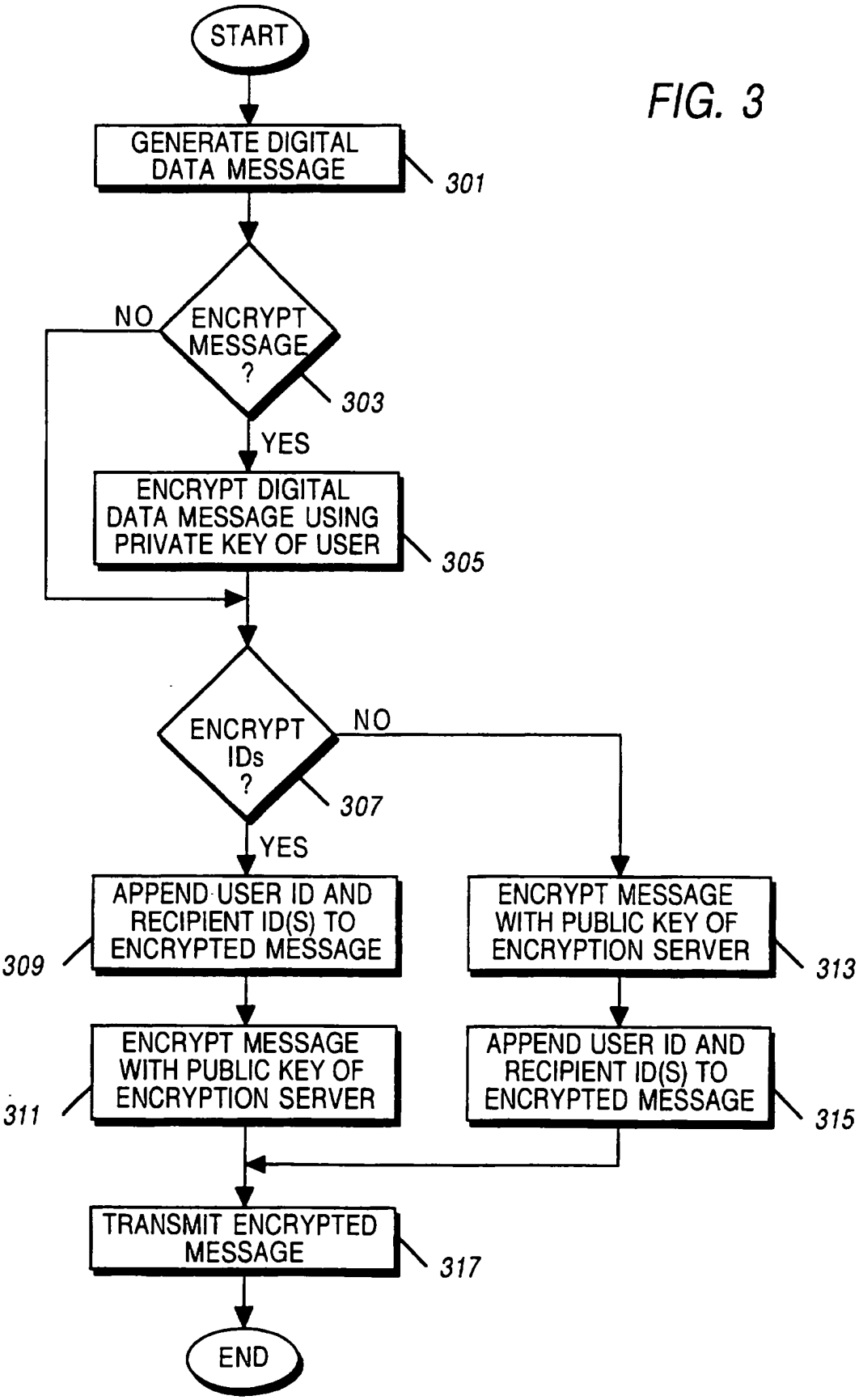
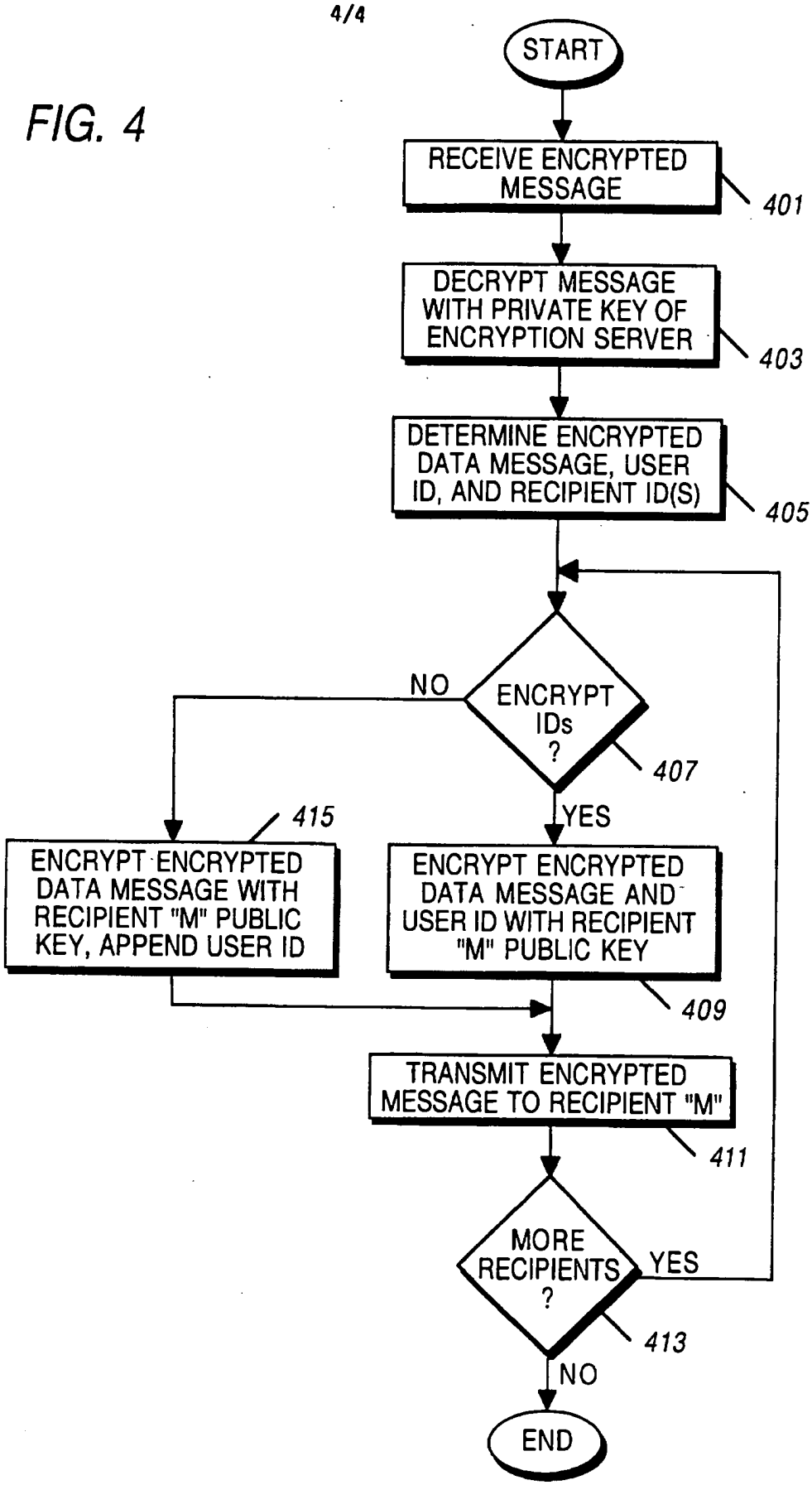

START
GENERATE DIGITAL DATA MESSAGE
301
ENCRYPT MESSAGE ?
303
NO
YES
ENCRYPT DIGITAL DATA MESSAGE USING PRIVATE KEY OF USER
305
ENCRYPT IDs ?
307
NO
YES
APPEND USER ID AND RECIPIENT ID(S) TO ENCRYPTED MESSAGE
309
ENCRYPT MESSAGE WITH PUBLIC KEY OF ENCRYPTION SERVER
311
ENCRYPT MESSAGE WITH PUBLIC KEY OF ENCRYPTION SERVER
313
APPEND USER ID AND RECIPIENT ID(S) TO ENCRYPTED MESSAGE
315
TRANSMIT ENCRYPTED MESSAGE
317
END

FIG. 4


START
RECEIVE ENCRYPTED MESSAGE
401
DECRYPT MESSAGE WITH PRIVATE KEY OF ENCRYPTION SERVER
403
DETERMINE ENCRYPTED DATA MESSAGE, USER ID, AND RECIPIENT ID(S)
405
ENCRYPT IDs ?
407
NO
YES
415
ENCRYPT ENCRYPTED DATA MESSAGE WITH RECIPIENT "M" PUBLIC KEY, APPEND USER ID
ENCRYPT ENCRYPTED DATA MESSAGE AND USER ID WITH RECIPIENT "M" PUBLIC KEY
409
TRANSMIT ENCRYPTED MESSAGE TO RECIPIENT "M"
411
MORE RECIPIENTS ?
413
YES
NO
END

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04N 5/913

A1

(11) International Publication Number: **WO 97/37492**

(43) International Publication Date: 9 October 1997 (09.10.97)

(21) International Application Number: PCT/US97/05257

(22) International Filing Date: 31 March 1997 (31.03.97)

(30) Priority Data:
60/014,684 1 April 1996 (01.04.96) US

(71) Applicant *(for all designated States except US)*: MACROVISION CORPORATION [US/US]; 1341 Orleans Drive, Sunnyvale, CA 94089 (US).

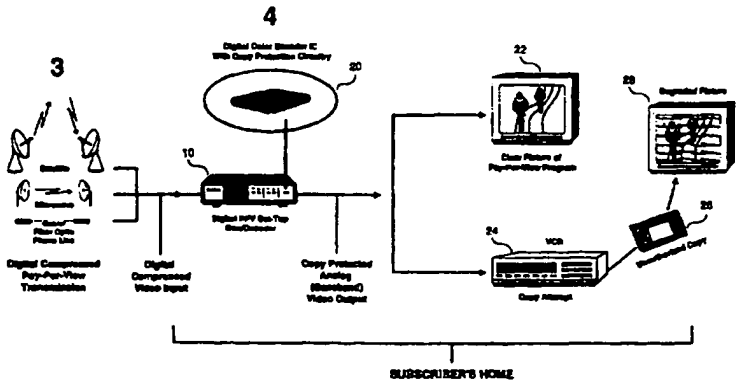
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: WONFOR, Peter, J. [US/US]; 962 Malaga, El Granada, CA 94089 (US). NELSON, Derek [US/US]; 3250 A. Glendale Avenue, Menlo Park, CA 94025 (US).

(74) Agent: BRILL, Gerow, D.; Macrovision Corporation, 1341 Orleans Drive, Sunnyvale, CA 94089 (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, GE, HU, IL, 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, TJ, TM, TR, TT, UA, UG, US, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, 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.
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: A METHOD FOR CONTROLLING COPY PROTECTION IN DIGITAL VIDEO NETWORKS



(57) Abstract

A method and system of providing copy protection of video analog and digital signals and the like, wherein the signals are transmitted via a digital delivery network, and may comprise, for example, pay-per-view (PPV) program materials protected by copyrights of respective program rights holders. The right holders authorize video service providers (3) to apply copy protection to the program material. The copy protection process is supplied to the rights holders or the service providers (3) by a copy protection process licensor. The video service providers (3) supply suitable copy protection control software via respective control and billing (tracking) centers to generate commands which activate, control and reconfigure the copy protection process being applied to the programs being transmitted. A set-top box (10) is provided to each consumer and contains a copy protection circuit which is adapted to apply selected anticopy waveforms to the video signal corresponding to the program material in response to the commands from the service providers (3). Usage data pertinent to each consumer is returned by the set-top box (10) to the service providers (3), which then report the copy protection usage to the respective rights holders and process licensor.

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		

A METHOD FOR CONTROLLING COPY PROTECTION IN DIGITAL VIDEO NETWORKS

BACKGROUND OF THE INVENTION

Field of the Invention

This disclosure is directed to a method of controlling copy protection in digital video networks where it is desired to copy protect an analog or digital video output signal associated with a digital video network.

Background of the Invention

Various well known copy protection schemes for video signals include that disclosed in U.S. Patent No. 4,631,603, John O. Ryan, December 23, 1986 and assigned to Macrovision Corporation, incorporated by reference, directed to modifying an analog video signal to inhibit making of acceptable video recordings therefrom. This discloses adding a plurality of pulse pairs to the otherwise unused lines of a video signal vertical blanking interval, each pulse pair being a negative-going pulse followed closely by a positive-going pulse. The effect is to confuse AGC (automatic gain control circuitry) of a VCR (video cassette recorder) recording such a signal, so that the recorded signal is unviewable due to the presence of an excessively dark picture when the recorded signal is played back.

Another analog video protection scheme is disclosed in U.S. Patent No. 4,914,694 issued April 3, 1990, to Leonard, and assigned to Eidak Corp., incorporated by reference. The Eidak system (see Abstract) increases or decreases the length of each video field from the standard length, either by changing the time duration of the respective horizontal line intervals in each field while keeping a constant, standard number of lines per frame, or by changing the number of horizontal line intervals which constitute a frame while maintaining the standard duration of each line interval.

These video protection systems modify the video signal to be recorded (for instance on tape) or to be broadcast (for instance protected pay-per-view television programs) to make copying by ordinary VCRs difficult or impossible. When a video tape on which is recorded the copy protected video signal is played back for viewing using a VCR, the copy protection process is essentially transparent, i.e., it does not interfere with viewing. However, any attempt made to copy the video signal from the tape using a second VCR to record the output of the first (playback) VCR yields a picture degraded to some extent, depending on the efficacy of the particular copy protection system. These present video copy protection systems protect only analog video signals, which are the type of video signals broadcast and recorded using current consumer video technology.

Some digital and hybrid solutions to the copy protection problem were solved by US Patent 5,315,448, issued May 24, 1994, issued to Ryan and assigned to Macrovision Corporation, incorporated by reference. This patent is directed to copy protection for use with digital signal recording where it is desired to copy protect both an analog and digital signal associated with a digital VCR, and any signal material where the original source material is not copy protectable.

A fundamental revolution is under way that will dramatically affect the delivery of home entertainment. Consumers will soon have hundreds of viewing options from which to choose because of advances in digital compression technologies and the associated reduction in costs accompanying each advance. Because of the increased number of channels more channels will be allocated for pay-per-view (PPV). The increased number of PPV channels will mean video service providers (VSP), also known as PPV providers or system operators, can provide a greater number of movies and more start times, ultimately changing the way many consumers purchase and view movies in their homes. Already, market research experts are predicting that the pay-per-view business will rival today's videocassette rental and sell-through business within 3-5 years.

Even with such a positive outlook for the future of PPV, the full benefits to the consumer of PPV programming may be delayed unless new digital video networks can protect PPV program copyrights. Rights owners are concerned that when digital programming is delivered to the home any digital set-top box will be able to produce a commercial quality video when recorded by a consumer VCR.

SUMMARY OF THE INVENTION

In this new world of direct-to-home video programming, video service providers will be called upon to protect PPV programming against unauthorized copying. They will be obligated to develop and manage the headend (cable) or uplink (satellite) systems which monitor, control, track, and report the application of copy protection on each pay-per-view video program. To this end, the present invention provides copy protection management framework which meets these needs while complementing the more technically detailed copy protection management strategy for video service providers. This framework serves to integrate all components of copy protection delivery in a digital network, and is designed to fit the diverse needs of DBS, Telco, and Cable operators while meeting the requirements of rights owners for a robust and secure environment in which to deliver copy protected PPV programming.

The value of PPV copy protection is maximized when the appropriate control and tracking systems are in place at the video service provider's control and billing centers. These control and tracking systems are best specified during the design phase of the digital signal material delivery system. At a minimum, the following system components are required:

- Copy protection-capable set-top boxes
- Capability to deliver programmable copy protection configuration
- Capability to deliver real time on/off/mode command
- Transaction/billing reporting systems/programs

A control and tracking system in accordance with the invention, for providing copy protection for a typical digital delivery system can be best understood through a short case study which begins when a consumer, that is a subscriber, receives a new set-top box. Each set-top box includes a copy protection capable digital-to-analog encoder chip. When the set-top box is initially powered on, the encoder chip is remotely programmed via a video service provider with the desired copy protection configuration. Thus the video service provider's system management software (SMS), also termed hereinafter as system control software (SCS), has the ability to store and track the designated configuration. The configuration information

applies to all copy protected programming and is updated only when a video service provider is informed of a change in the process or when a set-top box is initialized.

The copy protection status or option of each program is contained in the video service provider's system control software database. There are several potential copy protection status options. For example, a first option is for copy protection which allows for viewing only at a PPV transaction fee. A second option is for copy protection which allows for taping at a higher transaction fee. A third option is for non-protected program material for which no copy protection is required (for example, broadcast television).

When the consumer selects a viewing choice via an electronic program guide, a correct menu of options is displayed. Once a PPV program is selected by the consumer, the correct copy protection status is applied as determined by the consumer's chosen option and scheduling software of the system control software database. Either the headend/uplink facility's control software or software at the set-top box can determine and send the appropriate on/off/mode command to the copy protection capable digital-to-analog chip of previous mention.

The headend/uplink software communicates the on/off/mode command to the set-top box to correctly set the copy protection for a particular program. The system scheduling software has the capability to prevent copy protection from being applied to any type of program other than PPV programming since copy protection is licensed only for use on PPV programming. After a PPV program is viewed by a consumer, the set-top box is able to communicate to a billing subsystem of the system control software all relevant transaction data. From this data the billing subsystem is able to add this information to copy protection activity reports. These reports contain information such as the number of purchases, retail price, and copy protection usage fees owed to a licensor.

The copy protection process is applied to the analog video signal just prior to its exiting the consumer's set-top box. The application of the copy protection process is controlled and managed by system control/access software of the system control software that resides in the video service provider's operations control and billing center.

All set-top boxes in the network need to contain copy protection circuitry. If a set-top box does not have copy protection capability then the video service provider

is able to identify those set-top boxes and deny them copy protected PPV programming.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a diagram depicting a summary of the functions of the present invention.

Fig. 2 is a block diagram depicting a typical digital set top box/decoder of the present invention.

Fig. 3 is a block diagram illustrating an example of the circuitry and architecture of the set-top box of Fig. 2 in further detail.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

The basic copy protection which is controlled and tracked in accordance with the present invention, is the subject of numerous patents and co-pending applications. The PPV copy protection process works by exploiting the differences between the way television (TV) sets and VCRs respond to video signals. The two components of the anticopy process are known as the automatic gain control (AGC) and Colorstripe™ processes. The purpose of these two separate components or processes is to modify the video signal in a manner which has no effect on a TV set but which inhibits a recording VCR from making a watchable copy.

The combination of the AGC based anticopy process and the Colorstripe™ technology developed specifically for PPV applications results in an overall effectiveness rating of more than 95%. This means that over 95% of unauthorized copies will be either unwatchable or have substantially reduced entertainment value.

Security is also a major factor in the operational effectiveness of PPV copy protection. Security is a measure of the difficulty in bypassing or defeating the anticopy process. Ideally the system is completely undefeatable, but as a practical matter the copy protection system needs to be secure enough to thwart attempted breaches by typical consumers, including reasonably sophisticated consumers. The security system is successful if the vast majority of consumers are prevented from taping PPV programs in the home.

Both video service providers (VSPs), that is, PPV providers, and rights owners benefit when current movie programming is offered to consumers at the same time or shortly after these movies are available on videocassette. Subscribers benefit as well since this scenario provides them with more choices and added convenience.

As digital PPV programming generates increasing revenue for rights owners and becomes a viable viewing option to prerecorded videocassettes, video service providers will be called upon to copy protect PPV programming so that the videocassette rental and videocassette sell-through businesses are not compromised. Rights owners also will require video service providers to monitor, control, track, and report the application of copy protection on each video program for billing purposes.

Copy protection has emerged as a key element in the delivery of PPV programming via digital signal delivery networks. The aggregate system implications of copy protection are very manageable, but only when designed as a part of the overall digital delivery system architecture.

The description of the present invention is intended to apply to systems where one or more video service providers are, or will be in the future, connected to a pay-per-view (PPV) service. The PPV service can be either a video-on-demand (VOD) format, or a near video-on-demand (NVOD) format and digital delivery network, and where set-top boxes (STBs) from multiple manufacturers may be connected to the network. It is assumed that one class of technology will be deployed initially [such as Direct Broadcast Satellite (DBS), Multi-point Microwave Distribution System (MMDS), telephone line or Hybrid-Fiber Coax (HFC)] to be followed by another class of technology at some future date. Although a different technology may arise, it is intended that the invention is applicable to use with multiple platforms and technologies.

Fig. 1 illustrates a control and tracking method and system for enabling and controlling the application of copy protection of video signals and the like via digital video networks. Station 1 represents the issuance of instructions to video service providers by program rights holders who hold the copyrights, for the application by the providers of copy protection to the programs which are protected by per-per-view (PPV) or pay-to-tape (PTT) requirements.

Station 2 depicts a control and billing center of the licensed video service providers who supply copy protection control software for the respective protected programs being broadcast, to generate the commands required to activate, control and reconfigure the copy protection process for each specific PPV/PTT program offering. Although a single provider is depicted, it is understood that station 2 represents any plurality of video service providers each with their respective proprietary control and tracking (billing) software, in accordance with the present invention.

Station 3 represents the procedure of transmitting the particular copy protection command codes of the respective providers, for the PPV/PTT program offerings, via the typical broadcasting networks. Such transmissions may be made by satellite, by microwave, by phone line or by cable transmission systems as depicted.

Station 4 represents the subscriber's home, or other receiving facility, and includes a set-top box 10 for each of a multitude of subscribers. Each set-top box contains copy protection circuitry including a digital color encoder integrated chip (IC), which is adapted to apply selected anticopy waveforms to the analog or digital video signal which is supplied therefrom to a television set or monitor. The receiving facility is further described in Fig. 2.

Station 5 represents the procedure whereby data identifying each PPV or PTT transaction, including copy protection usage, is sent by the set-top box 10 back through the transmission networks of station 3, generally to the respective video service provider's control and billing (tracking) center. The center includes billing procedures which are a subset of the system control software and which process the return transaction data to provide for billing the subscriber for the PPV or PTT transaction usage.

Station 6 represents the procedure whereby each of the licensed video service providers report the copy protection usage to the program rights holder, whereby the provider pays the copy protection fees to the rights holder, i.e., the licensor.

Fig. 2 illustrates in further detail the subscriber's facility, station 4 of Fig. 1, receiving the digital, and usually compressed, pay-per-view transmissions from the broadcasting networks depicted as station 3 of Fig. 1. The compressed digital video

signal, or the like, is supplied to the respective set-top box 10 of a multitude of set-top boxes, wherein each box includes conventional circuits for converting and decoding the digital compressed video signal to an analog (baseband) video signal. The set-top box 10 also includes a digital color encoder IC 20 of previous mention which contains copy protection circuitry for applying the selected copy protection waveforms to the analog (or digital) video signal, namely, the programs which are being protected. In this example, the copy protected analog baseband video is supplied by the set-top box to a TV set 22 where the pay-per-view protected program clearly is displayed for viewing if the subscriber is authorized to view the program. If the subscriber is not authorized for a particular PPV protected program, the corresponding picture is modified so as to be un-viewable.

In the event a subscriber records the PPV protected program via a VCR 24 to obtain a taped copy 26 without authorization, the unauthorized copy will be degraded to the degree that it is un-watchable, as depicted by a TV set 28. However, if the subscriber subscribes to a pay-to-tape transaction and to the required higher PTT transaction fee, then the copy is authorized and the resulting taped copy would readily be watchable.

Referring to Fig. 3, there is illustrated in further detail an architecture of the set-top box(es) 10 of Figs. 1, 2. Upon power up of the set-top box 10 the configuration bits stored in flash memory 48 are read and written into the appropriate CP control registers 52 in the NTSC/PAL encoder 20. When the compressed digital video signal, including the copy protection control commands of previous and following discussion, are supplied by the delivery network of previous mention (satellite, HFC, MMDS, phone line) to a demodulator circuit 32, as depicted by an input lead 30. The demodulated video/audio and control signals are supplied to a demultiplexer circuit 34 where the video/audio signals are separated into respective channels and supplied to an MPEG-2 decoder and digital decompression circuit 36. The copy protection control commands are supplied from the demultiplexer 34 to a conditional access system module 38. The commands are supplied to a microprocessor in a CPU 40. The CPU processes information located in memory that is associated with the Electronic Program Guide (EPG) 46 or runs the copy protection application software 44 residing in memory 42 to deliver the activation command to the NTSC/PAL encoder 20. The EPG may also have data

which is used to determine if copy protection should be activated. There are additional methods that may be employed to activate copy protection.

In response to the control commands, the CPU 40 supplies control signals to the NTSC/PAL encoder IC 20 of previous mention, Fig. 2. The encoder IC 20 includes copy protection control registers 50, 52 for receiving the mode bits and configuration control bits respectively, of previous and following discussion. The configuration bits 52 determine the form of the copy protection (i.e., where the Pseudo Sync and AGC pulses will be located or positions of the colorstripe lines etc.) The on/off/mode byte 50 determines which components of the copy protection process will be activated. See table 1 below. The encoder IC 20 also receives decompressed video from the MPEG-2 decoder and digital decompression circuit 36. Encoder IC 20 outputs a RF signal, a composite video signal and/or an S-video signal via video leads 54. The decompressed audio signal is supplied from the circuit 36 to an audio processing circuit 56 which, in turn, outputs left and right channel stereo signals and/or an AC-3 signal on audio leads 58.

In accordance with the invention, the set-top box needs to satisfy certain requirements to insure that the copy protection process is correctly generated, controlled and tracked. Control and tracking of the copy protection process usage takes place at the VSP's control and billing center, station 2 of Fig. 1. This in turn requires that certain capabilities exist which involve the set-top box, the system control and the billing systems and programs in order to satisfy these requirements.

There follows a description of the requirements which ensure that the copy protection process or technique is correctly activated and controlled and its usage tracked. It is expected that if non-compliant set-top box hardware is attached to the digital delivery network, that each licensed service provider will be able to identify such hardware as non-compliant and will withhold copy protected programs from the respective subscriber.

Implementation of these control requirements over the network (i.e. control of the anticopy process from the program origination control and billing center) requires knowledge of the set-top box control system and process, the application program interfaces (API) present at the box and the dialog between it and the integrated circuit (IC) which incorporates the copy protection apparatus.

Copy protection control software (CPCS) is a software module or set of software modules that reside in the service provider's system control software (SCS). It provides a system operator (that is, the service provider) with an interface to manage the necessary attributes of the pay-per-view copy protection in accordance with the present invention.

For security reasons there needs to be the capability to control access to the CPCS from the system control software. This restriction is designed to limit access to the CPCS for control of the copy protection process. The operating system supporting the SCS is generally the first level of security. Every employee is required to enter a login account and password. Without these an employee is denied access. The employee's account specifies the respective privileges. A system administrator of the service provider is responsible for the assignment of the employee's privileges.

Thus, every executable file residing on the host which is capable of modifying the operational status of the copy protection process has permissions restricted to authorized personnel. Without the proper permissions, the personnel are unable to run the executable software.

The CPCS is the portion of the video service provider's software control where the decision to apply the options of pay-per-view and pay-to-tape are applied on a program-by-program basis.

There is access control to the CPCS either through password control or the assignment/denial of privileges through software. If password control is the selected method then once the correct initial password is entered, CPCS forces the selection of a new password for future access to CPCS. In this way the service provider can limit access to CPCS to those employees who carry the authority to modify the copy protection database. The password is valid for a reasonable amount of time before it expires and selection of a new password is required.

Additionally there is an access control to a subsystem within the CPCS that allows the modification of selected bits which define the configuration control and mode, and thus determine the characteristics, of the copy protection process. Any unauthorized changes to these bits can result in severe playability and effectiveness problems. In order to maximize the security of the system the video service provider needs to have a short list of personnel who are authorized to change these bits.

A mode control group controls access to the mode bits. This group has the ability to change the contents of the mode byte(s) which is sent with each PPV program to activate or deactivate the copy protection process. The membership of this group is controlled by the system administrator. The number of the service provider's personnel allowed in this group is kept to a minimum.

Similarly, a configuration control group controls access to the configuration bits. This group has the ability to change the contents of the configuration bits which define the copy protection process. These are the bits that are sent periodically to every set-top box to assure that all boxes are using the correct version of the process. The number of the service provider's personnel allowed in this group also is kept to a minimum.

Each password described below should be at least eight (8) alpha-numeric characters in length. The system administrator is responsible for defining and distributing the current password to the authorized personnel. Each password described below should have a life of no more than four months before the system administrator changes the password.

Password access to the software that applies or removes the copy protection process on a program-by-program basis is designed to query mode or configuration control group authorized personnel for an authorization password to ensure that they are a member. If the authorized personnel correctly enter the password they will be allowed to apply or remove the copy protection for a particular PPV or series of PPV events. Conversely, if authorized personnel fail to enter the password they must be denied access to that portion of the database. It is the system administrator's responsibility to ensure that only authorized personnel know the password for either the mode or configuration control. An authorized personnel will be given three attempts to login before a message is generated for the system administrator that an unauthorized request to modify the application or remove the copy protection has been made.

Alternative proposals for accessing CPCS and controlling access to the mode and configuration of the copy protection process may be developed by one skilled in the art.

The CPCS will perform the following functions: Copy protection on/off and mode control; copy protection validation; functionally unlocking copy protection

capability in a set-top box; and copy protection process configuration reprogramming.

The copy protection process which is incorporated in the set-top box is controlled by the CPCS at the licensed video service provider's control and billing central location. The need to invoke copy protection on an individual program forms part of a descriptor for each program. A default for copy protection within the descriptor needs to be turned off (i.e., no copy protection).

Steps need to be taken to prevent copy protection being applied to non-PPV program channels, since copy protection can be licensed only for PPV programming. If the system control software automatically verifies that a program is designated for PPV use, this requirement may be automated. Similarly, access to CPCS may be automatically denied for non-PPV programming. If such an automatic verification is not made, a warning notice is generated when CPCS is accessed to change the copy protection status of a program. This notice needs to be displayed until a specific keyboard entry is made to acknowledge the warning.

In the case of MPEG signals, the MPEG copyright header bits on their own are not sufficient to activate copy protection in the set-top box. The following reasons are the basis for not allowing the MPEG header bits to be used as the sole control of the copy protection process. An application routine is required in order to (a) differentiate between digital-to-digital and digital-to-analog copy protection conditions, (b) provide sufficient control capacity to set the copy protection operating mode, and (c) facilitate access to the copy protection system only by licensed video service providers.

It is preferred that the anticopy process on/off control is achieved by setting all the individual parameter on/off and mode control bits rather than a master on/off control. This requires that the N0 (N-zero) bits in the control bit listing be set as required. Depending on the individual system, this will require the control of from 5 to 8 bits.

The delivery of the mode byte to the set-top box to activate or deactivate the copy protection process may be accomplished in several ways. Each method has its positive aspects as well as its negative aspects. When selecting a mechanism to control the copy protection technology, a service provider selects one of the following means or may develop an entirely new means.

One method may be for the mode byte to be delivered via the conditional access system via the entitlement control message (ECM). Another method might be to include the mode byte in a private data field in the MPEG transport data stream.

Another method may deliver the mode byte in a user defined section of the electronic program guide (EPG) that is not identified in released documentation as controlling copy protection. This method also requires some additional security to keep the memory location of the mode byte from being accessed for unauthorized changes and the setting of a return flag that indicates the actual status of the mode byte when transmitted to the NTSC encoder.

Another method may be a combination of the conditional access ECM and EPG. The transport of the mode byte in the EPG could be combined with two bits within the ECM. To activate the copy protection technology then it would be an or operation between the ECM bits and the EPG bits. If either is set, the copy protection technology, both ECM and EPG would have to indicate that deactivation is necessary.

When a copy protected PPV program is viewed, part of the information that will need to be tracked will be the actual setting of the mode byte. In this way both the copy protection process and the service provider will have a means to discover if copy protection has been circumvented in the set-top box. The return flag may be a simple bit set to 'true' to indicate that the copy protection process was correctly activated and 'false' if it was incorrectly activated. It is required that the mode byte be sent to the NTSC encoder on a periodic basis. The frequency of the transmission is on the order of once every minute.

Setting the operating mode of the copy protection process requires independent activation of the three component parts of the copy protection process (pulses within the vertical blanking interval, pulses at end of field, colorburst phase modification) and up to 5 additional mode set parameters using N0 bits as indicated above.

Access to copy protection at the set-top box by the video service provider needs to be restricted to authorized providers. This should not to be confused with access to the CPCS as defined earlier. It follows that each system operator or video service provider is required to procure the means (i.e., keys/codes, etc.) to activate

the copy protection system control software on a program-by-program basis. When a service provider obtains the means to activate copy protection, the provider will gain access to the copy protection process at the set-top box. The copy protection process (i.e. on/off/mode or reprogramming commands) at the set-top box needs to have controlled access such that only authorized providers can issue valid commands to the box. The set-top box needs to reject commands for the copy protection process from unauthorized video service providers.

Set-top boxes such as depicted in Figs. 1, 2, may be shipped by the manufacturer with the copy protection capability installed, but functionally locked. This means that the set-top box will not respond to any copy protection control codes. However, the set-top box will be unlocked (i.e. enabled) by a message initiated via the CPCS or SCS and sent through the system by a licensed video service provider. This message may be sent as part of the log-on routine when a subscriber accesses a provider. This message need only be acted upon once by the set-top box during the lifetime of the box. Only authorized video service providers are provided with the unlocking message data.

The copy protection unlock message consists of at least 8 bytes. The set-top boxes are manufactured with an appropriate unlock message code. This code is provided by the set-top box manufacturer only to a copy protection licensor, who in turn provides the code to licensed video service providers. The copy protection unlock message is different for each set-top box manufacturer, but is the same for all boxes made by that manufacturer.

Alternative proposals on the methodology to enable the copy protection process in the set-top box will be apparent to those skilled in the art.

To ensure that over the life of the set-top box the copy protection process provides the maximum effectiveness with VCRs and compatibility with TV sets, the copy protection system needs to be upgradeable on a system-wide basis by means of commands initiated by the CPCS. This will result in new process configuration data being transmitted. In response, the set-top box processes the data to reconfigure the adjustable parameters of the copy protection process. The set-top box may be placed in a "diagnostics" mode for this feature implementation, or the configuration data may be sent and acted on by the box on a routine basis as part of the program description data or log-on routine.

However, it is recommended that the entitlement control message (ECM) be used. The ECM is embedded in the conditional access system.

In one version, configuration data of 108 bits is provided to accommodate the reconfiguration data, however, 108 bits does not fall on a byte boundary. Therefore, it is recommended that 112 be sent with a pad 0. The data is presented to the service provider in the form of hexadecimal numbers for entry into the CPCS. The 112 bits thus are entered as a string of 28 hexadecimal numbers.

In another version, configuration data of 132 bits is provided to accommodate the reconfiguration data, however, 132 bits does not fall on a byte boundary. Thus, it is recommended that 136 be sent with a pad 0. The data is presented to the provider in the form of hexadecimal numbers for entry into the CPCS. The 136 bits thus are entered as a string of 34 hexadecimal numbers.

It is possible to verify the current configuration stored by the CPCS by accessing the current contents of the configuration bits presented as the correct number hexadecimal characters. An alpha-numeric password of at least 8 bytes is required to gain access to change the programming data within CPCS. This password is separate from the password which allows access to CPCS. The service provider has the option of receiving the 'C' source code of an executable file to which to pass parameters.

The following warning notice is presented on the screen of the operational control and billing center of a provider after entering the correct password:

WARNING

Changing this copy protection configuration data without the written authorization carries the serious risk of problems with the performance of the copy protection system and degraded picture quality.

This warning notice is displayed until a specific keyboard entry is made to acknowledge the warning.

By way of example only, Table 1 illustrates a mode control bit listing which defines the corresponding bit pattern or command, which provides the routine on/off

and mode selection functions when transmitted to the set-top boxes via the delivery networks. The configuration control bit listing is generally equivalent to that of the mode control, though relatively longer since it controls considerably more control and reprogramming functions.

TABLE 1
Mode Control Bit Listing
Routine On/Off and Mode Selection

N0	**On/off and mode control; 8 bits**		
N0[7]	Reserved		CPC0[3]
N0[6]	Pay-to-tape allowed/prohibited	(Allowed=1, Default=0)	CPC0[2]
N0[5]	VBI pulses On/Off (VBIP)	(ON=1)	CPC0[1]
N0[4]	End of Field Back Porch Pulses on/off (EOFP)	(ON=1)	CPC0[0]
N0[3]	Colorstripe process On/Off (CSP)	(ON=1)	CPC1[3]
N0[2]	AGC pulse normal (amplitude cycling)/static mode select (AGCY)	(Cycling=Default=1)	CPC1[2]
N0[1]	H-sync amplitude reduction On/Off (HAMP)	(ON=1)	CPC1[1]
N0[0]	V-sync amplitude reduction On/Off (VAMP)	(ON=1)	CPC1[0]

The pay-per-view transaction information is collected by each video service provider for each subscriber so that monthly copy protection activity reports required for royalty payments and other fees may be generated. The reports include information regarding the number of subscribers accessing each copy protected program, with subtotals of the copy protection status or options selected by respective subscribers. The reports further include information sorted by PPV title, PPV program supplier, copy protection activation status requested by the subscriber, and by set-top box model code. The reports are provided by the report generating software of previous mention at the video service provider centers.

The activity report includes a manufacturer and model type descriptor code in the transaction acknowledgment between the set-top box and the control and billing system when a PPV purchase transaction is reported to the provider.

The CPCS and the set-top box are capable of applying and reporting anticopy usage according to the following conditions. The overall system allows the subscriber's copy protection to be turned off at the box only as permitted by the PPV program rights holder.

(a) PPV program rights holder permits viewing only:

The pay-to-tape mode is prohibited (off). All STBs output copy protected waveform only. I.e., the copy protection waveform unconditionally appears on the set-top box analog video output signal.

This is reported to the billing system as a "pay-per-view" copy protected transaction.

(b) PPV program rights holder permits viewing and recording:
The pay-to-tape mode bit is set for pay-to-tape permitted (on). Under this option, when the subscriber selects the "pay-to-tape" option, the copy protection process is turned "off" in the STB to allow the PPV program to be recorded (taped) for a higher transaction fee than for "viewing only." I.e., the copy protection waveform will not be present on the STB analog video output signal.

This is reported to the billing system as a "pay-to-tape" copy protected transaction.

The following Table 2 provides a summary of the control options and includes additional information.

TABLE 2
Pay-per-view and Pay-to-tape Control Options for Pay-per-view Programs

Program Descriptor of PPV Program	Consumer Request (Pay-per-view or Pay-to-tape)	Result
Copy protection NOT required	N/A	ACP off
Copy protection REQUIRED Taping NOT permitted	Pay-per-view	ACP will be ON. Pay-per-view transaction cost incurred by consumer.
Copy protection REQUIRED Taping NOT permitted	Pay-to-tape	Requested option not available. ACP will be ON. Pay-per-view transaction cost incurred by consumer.
Copy protection REQUIRED Taping permitted (at higher transaction cost)	Pay-per-view	ACP will be turned ON by STB control system. Pay-per-view transaction cost incurred by consumer.
Copy protection REQUIRED Taping permitted (at higher transaction cost)	Pay-to-tape	ACP will be turned OFF by STB control system. Pay-to-tape transaction cost incurred by consumer.

It is to be understood that various terms employed in the description herein are interchangeable. For example, a "video service provider" also is known as a pay-per-view (PPV) provider or a system operator, and the "system management software" preferably is referred to as the system control software. Likewise, the "control and billing centers" of the PPV providers represented by station 2 (and generally station 5) also may be referred to as operations control/tracking centers, program origination/termination centers, headend (cable)/uplink (satellite) control centers, etc. A licensed PPV provider facility supplies the necessary control instructions to associated software and/or circuitry in a set-top box to allow a respective subscriber access to program material to which he or she is entitled, and also receives at designated times of the week, month, etc., the usage data

automatically returned by the set-top box. A billing and license fees software subset of the system control software then enables each PPV provider to bill the subscribers and to report and pay the attendant licensing fees to the rights holders, etc.

Accordingly, the above description of the invention is illustrative and not limiting. Further modifications will be apparent to one of ordinary skill in the art in light of this disclosure. For example, although the invention is described herein relative to a video signal, and primarily an analog video signal, it is to be understood that the invention concepts may be applied to other signals with properties equivalent to a video signal where copy protection is desired. Likewise, the invention is applicable to the copy protection of digital as well as analog signal materials, such as those disclosed in the U.S. Patent No. 5,315,448 of previous mention. Further, although a specific example of a code word is disclosed herein for enabling the copy protection process via the set-top box, other combinations and numbers of bits may be employed. In addition, a selected portion of the control software for effecting the copy protection process may reside in the set-top box in the form of an insertable "smart" card, wherein for example the smart card contains the data concerning the subscriber's options and privileges.

Thus, the scope of the invention is defined by the following claims and their equivalents.

What is claimed is:

1. A method of providing copy protection of signal material transmitted via digital delivery networks, to prevent unauthorized viewing or copying of the signal material, comprising the steps of:

supplying copy protection controls indicative of desired copy protection for the signal material;

transmitting commands derived from and in response to the copy protection controls which activate the copy protection for the signal material; and

applying anticopy waveforms to the signal material in response to the commands to prevent the unauthorized viewing or copying of the signal material.

2. The method of claim 1 wherein the step of supplying includes:

establishing selected requirements for activating and controlling a process which enables said copy protection and which reports the corresponding usage thereof; and

providing copy protection control software in response to the selected requirements, which software provides said copy protection controls to activate and control the copy protection process and the usage reports.

3. The method of claim 2 wherein the step of establishing includes:

establishing requirements which differentiate between digital-to-digital and digital-to-analog copy protection conditions, which determine a copy protection process operating mode and configuration, and which ensure that there is only authorized access to the copy protection process.

4. The method of claim 2 wherein the step of providing includes:

generating the commands in the form of a bit pattern in response to the copy protection control software; and

said commands including a first bit pattern which enables real time on/off/mode control, and a second bit pattern which determines a programmable copy protection configuration.

5. The method of claim 4 including the step of:

receiving the transmitted first and second bit patterns to activate the copy protection and to control and reconfigure the copy protection process respectively in response thereto; and wherein the anticopy waveforms are applied to the signal material to provide the copy protection.

6. The method of claim 2 including the step of:

limiting access to the steps of establishing and providing to prevent unauthorized access to the application of the copy protection process or to the copy protection control software which activates and controls the process.

7. The method of claim 2 wherein the step of applying includes:

storing the copy protection controls in memory at a service provider receiving facility; and

storing control data in memory at a signal material receiving facility, which stored control data is responsive to the commands to activate, control and reconfigure the stored copy protection process.

8. The method of claim 2 including the step of:

collecting periodic copy protection activity information including copy protection activation status such pay-per-view and pay-to-tape number of signal material events watched.

9. The method of Claim 8 including the steps of generating reports which include the number of accessing receiving facilities, the rights holder of the signal material events, the number of total events watched, and corresponding billing information.

10. The method of claim 2 wherein the step of applying includes:

modifying a selected synchronizing signal in a corresponding blanking interval of a television line in response to said commands to degrade a subsequent

decoding of the synchronizing signal in the event that a recording is made of the corresponding signal material.

11. The method of claim 2 wherein the signal material is a video analog or digital signal.

12. Apparatus for controlling copy protection of proprietary signal material transmitted via digital delivery networks, wherein a service provider enables a copy protection process which prevents unauthorized copying of the signal material by consumers, the apparatus comprising:

a control/billing center for supplying copy protection control signals as directed by the service provider;

means for transmitting selected commands in response to the copy protection control signals to selectively control the copy protection process; and

means located with each consumer for applying the copy protection process to the signal material in response to the transmitted selected commands to prevent or allow viewing or copying of the signal material.

13. The apparatus of claim 12 wherein the copy protection control signals of the service provider include:

a mode command for activating the box means; and

a configuration bit pattern for determining the copy protection process's operating configuration.

14. The apparatus of claim 13 wherein the copy protection control signals include an access password for identifying that a service provider's authorized personnel have access to and control of the copy protection process.

15. The apparatus of claim 13 wherein the box means includes a set-top box having encoder means containing a copy protection circuit adapted to add anticopy signals to the signal material in response to the command signals.

16. The apparatus of claim 15 wherein the set-top box includes:

memory means for storing the copy protection configuration and/or copy protection mode; and said encoder means including means for receiving the mode command and the configuration bit pattern and for controlling the activation and configuration of the stored copy protection process in response to the command and bit pattern.

17. The apparatus of claim 15 wherein the set-top box includes software for returning usage data back to the service provider's control/billing center, said usage data being used by the service provider to bill the consumers and to provide a report of the usage and corresponding license fees.

18. The apparatus of claim 13 wherein the signal material is a pay-per-view or pay-to-tape video analog or digital signal.

19. The apparatus of claim 12 wherein the control/billing center includes:

instructional information establishing requirements for activating and controlling the copy protection process and for reporting the copy protection activity; and

wherein the service provider supplies copy protection control software commensurate with said requirements, and said copy protection control signals in response to the control software.

20. A method of providing copy protection of signal material transmitted via a digital delivery network, wherein a service provider enables a copy protection process via a set-top box located at a consumer's facility, comprising the steps of:

supplying selected control bit patterns from the service provider to the consumer's facility via the digital delivery network;

storing a copy protection configuration in the set-top box;

receiving the control bit pattern in said set-top box; and

applying the copy protection process to the transmitted signal material in response to the control bit pattern each time a selection of the material is made at the consumer's facility to prevent or allow the selected signal material to be copied.

21. The method of claim 20 wherein the step of supplying includes:

developing copy protection control software which describes selected control signals for applying the copy protection process to the signal material and for returning to the service provider usage data indicative of the signal material selected at the consumer's facility;

generating said selected control bit patterns in response to the copy protection control software; and

transmitting said selected control bit patterns to the set-top box of the consumer's facility when the consumer joins the delivery network and thereafter on a prescribed routine basis.

22. The method of claim 21 including the steps of:

storing in the set-top box copy protection application software which activates and controls the copy protection process; and

enabling the stored application software in response to the transmitted control bit pattern to selectively activate and/or modify the configuration of the copy protection process.

23. The method of claim 22 including the steps of:

modifying the configuration control bit pattern commensurate with a desired change in the copy protection process; and

transmitting the modified configuration control bit pattern to the set-top-box to effect the change in the copy protection process.

24. The method of claim 21 including the steps of:

storing consumer information in the set-top box which is indicative of viewing and/or copying options desired at the consumer's facility; and

comparing the control bit pattern to the stored consumer's information in the set-top box when a selection of the signal material is made to determine if the consumer is authorized to view only and/or to copy the material.

25. The method of claim 20 wherein:

the signal material is a pay-per-view (PPV) or pay-to-tape (PTT) signal; and

the step of supplying includes establishing selected requirements for activating and controlling the PPV and PTT copy protection process and for reporting the corresponding usage activity of the process to the service provider; and

providing copy protection control software in response to the selected requirements, which software provides said control bit pattern to activate, control and modify the PPV and PTT copy protection process.

26. The method of claim 25 including the step of:

providing limited access to the steps of establishing and providing to prevent unauthorized access to the control of the copy protection process or to the copy protection control software.

27. The method of claim 25 wherein the signal material is a pay-per-view or pay-to-tape video analog or digital signal.

28. The method of claim 27 wherein the step of applying includes:

modifying a selected synchronizing signal in a corresponding blanking interval of a television line in response to said control bit pattern to degrade any subsequent decoding of the synchronizing signal when an unauthorized attempt is made to view or copy the pay-per-view signal.

29. A method of providing copy protection of signal material transmitted via a digital delivery network, wherein a service provider enables a copy protection process via set-top boxes located at consumers' facilities, comprising the steps of:

establishing selected requirements for activating, controlling and modifying a copy protection process for the signal material and for reporting the corresponding usage thereof;

providing copy protection control software in response to the selected requirements;

generating via the control software, mode and configuration control bit patterns which enable real time on/off mode control and programmable copy protection process configuration control respectively;

transmitting the mode control and configuration control code words to the set-top boxes;

selectively applying the copy protection process to the transmitted signal material in response to the transmitted mode bit pattern each time a selection of the signal material is made via the set-top boxes to prevent or allow the selected signal material to be viewed or copied.

30. The method of claim 29 including the steps of:

storing the application software in the set-top boxes; receiving and writing the mode bit pattern in the set-top boxes; and

wherein the stored application software responds to the transmitted mode bit pattern to activate, control and modify the copy protection process as defined by the configuration control bit pattern.

31. The method of claim 30 wherein the set-top box is functionally locked including: downloading via the service provider a selected bit pattern or software adapted to functionally unlock the set top box.

32. The method of claim 30 wherein the set-top box is functionally locked including activating at the service provider's facility selected software adapted to functionally unlock the set-top box

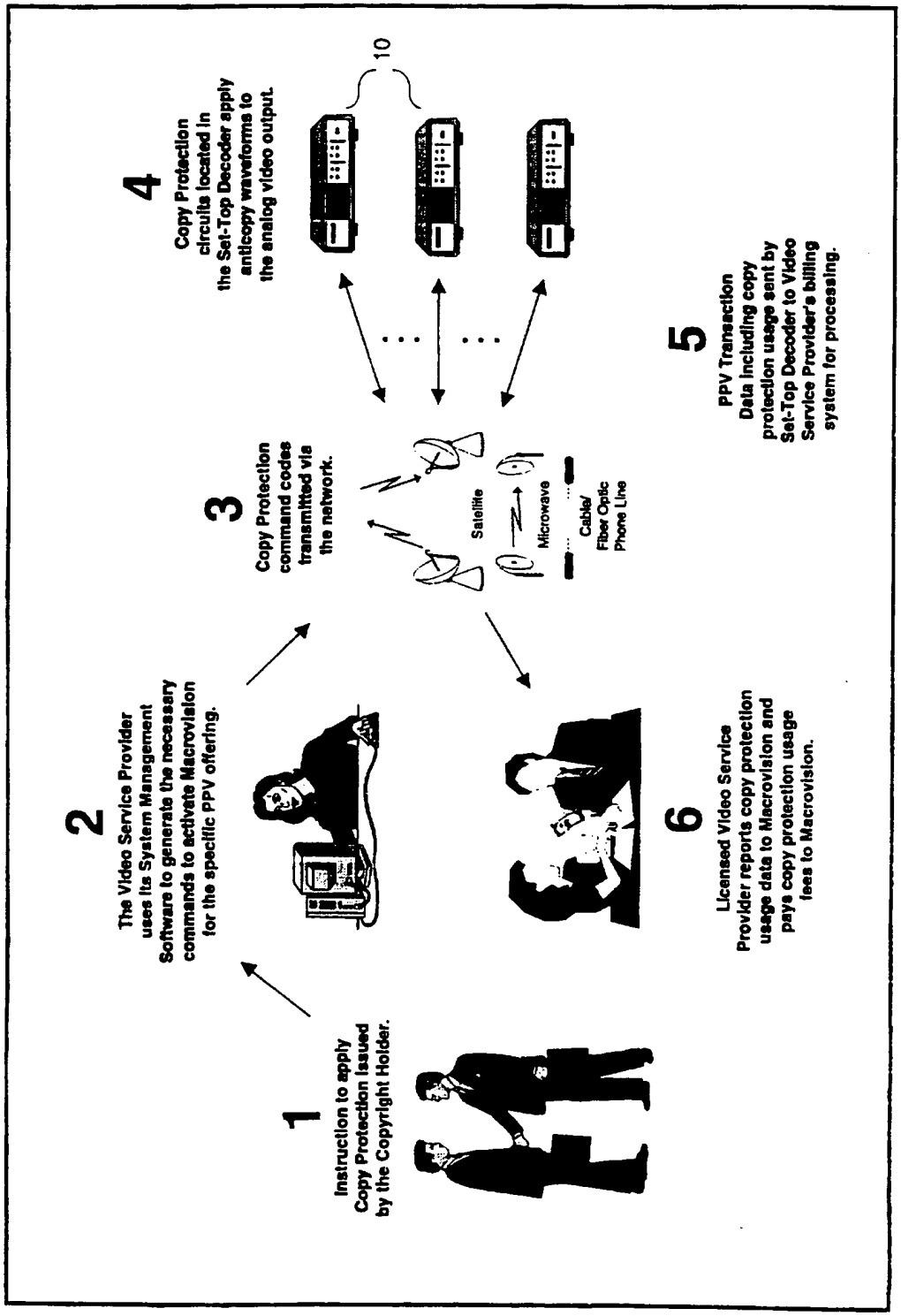

1
Instruction to apply Copy Protection issued by the Copyright Holder.
2
The Video Service Provider uses its System Management Software to generate the necessary commands to activate Macrovision for the specific PPV offering.
3
Copy Protection command codes transmitted via the network.
Satellite
Microwave
Cable/ Fiber Optic Phone Line
4
Copy Protection circuits located in the Set-Top Decoder apply anticopy waveforms to the analog video output.
10
5
PPV Transaction Data including copy protection usage sent by Set-Top Decoder to Video Service Provider's billing system for processing.
6
Licensed Video Service Provider reports copy protection usage data to Macrovision and pays copy protection usage fees to Macrovision.

FIG. 1

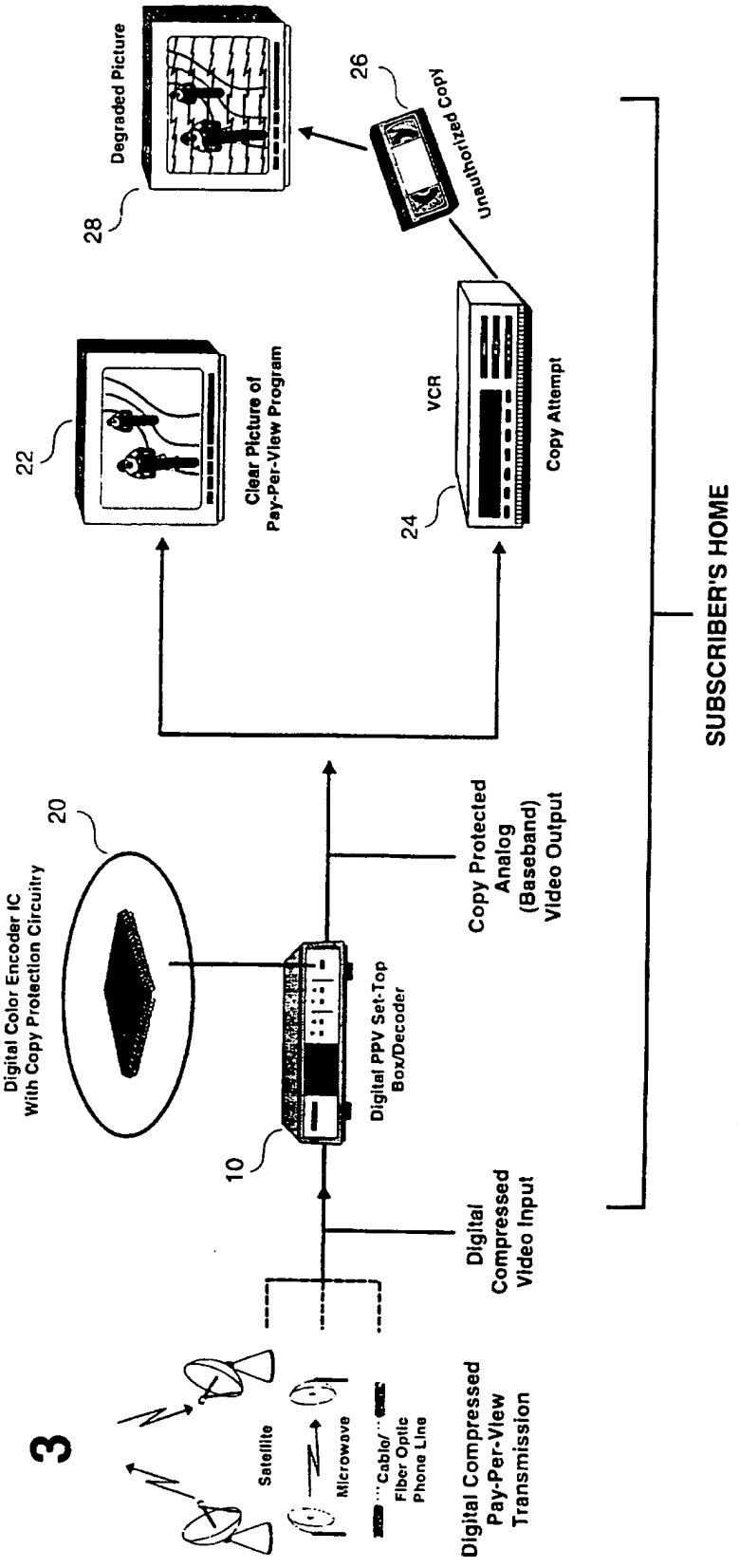

3
Satellite
Microwave
Cable/
Fiber Optic
Phone Line
Digital Compressed
Pay-Per-View
Transmission
Digital
Compressed
Video Input
10
Digital PPV Set-Top
Box/Decoder
4
Digital Color Encoder IC
With Copy Protection Circuitry
20
Copy Protected
Analog
(Baseband)
Video Output
SUBSCRIBER'S HOME
22
Clear Picture of
Pay-Per-View Program
24
VCR
Copy Attempt
26
Unauthorized Copy
28
Degraded Picture

FIG. 2

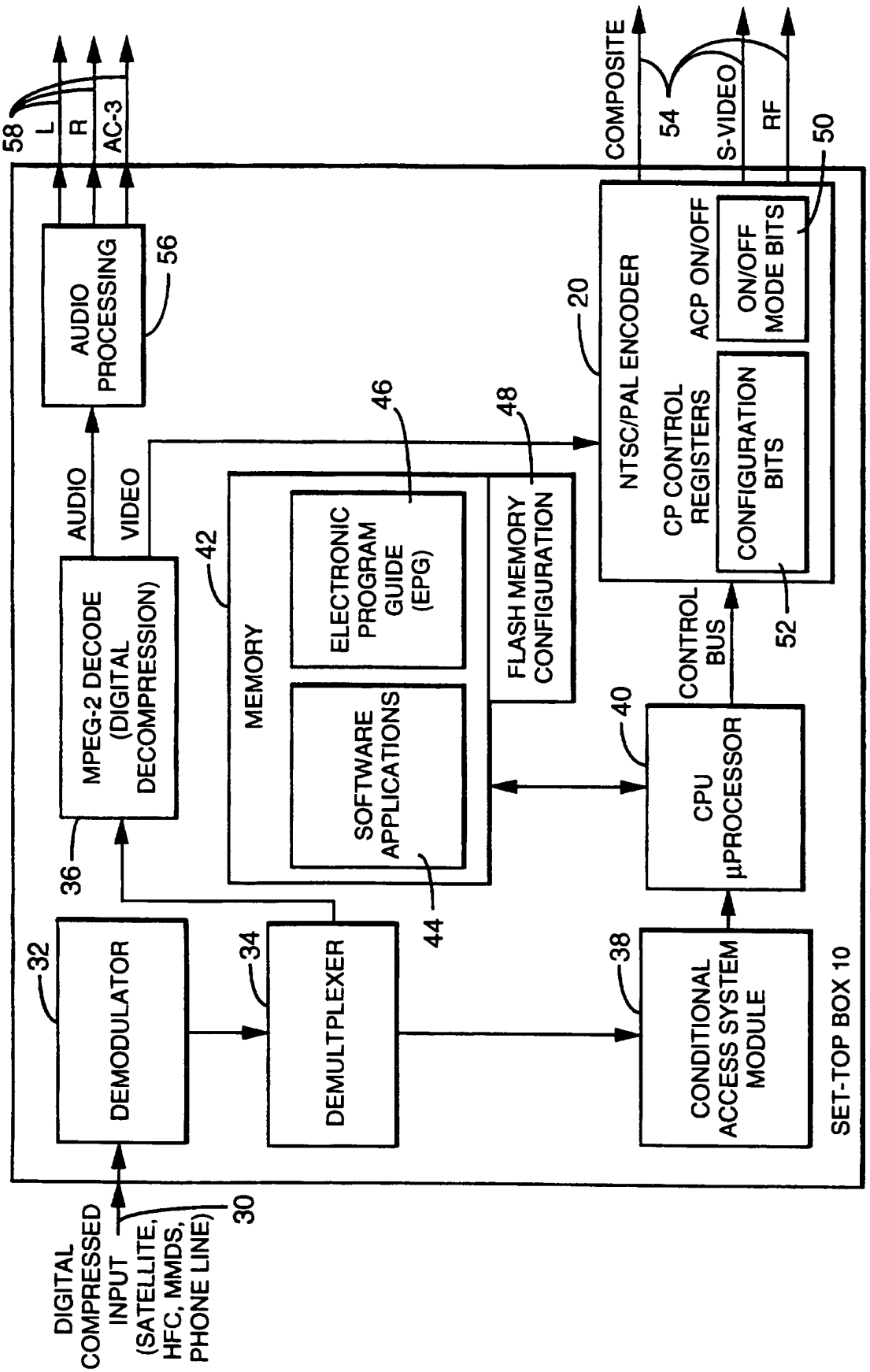

DIGITAL COMPRESSED INPUT (SATELLITE, HFC, MMDS, PHONE LINE)
30
SET-TOP BOX 10
DEMODULATOR
32
DEMULTPLEXER
34
CONDITIONAL ACCESS SYSTEM MODULE
38
CPU µPROCESSOR
40
CONTROL BUS
52
MPEG-2 DECODE (DIGITAL DECOMPRESSION)
36
AUDIO
VIDEO
MEMORY
42
ELECTRONIC PROGRAM GUIDE (EPG)
46
SOFTWARE APPLICATIONS
44
FLASH MEMORY CONFIGURATION
48
NTSC/PAL ENCODER
20
CP CONTROL REGISTERS
CONFIGURATION BITS
ACP ON/OFF
ON/OFF MODE BITS
50
AUDIO PROCESSING
56
58
L
R
AC-3
COMPOSITE
54
S-VIDEO
RF

FIG. 3

INTERNATIONAL SEARCH REPORT

Int. .onal Application No
PCT/US 97/05257

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04N5/913

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 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)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 691 787 A (SONY CORPORATION) 10 January 1996 see the whole document ---	1,2,5, 11,12, 15,18, 20,21, 27,29
A	US 5 315 448 A (RYAN) 24 May 1994 cited in the application see the whole document -----	1,4, 10-12, 18,20, 21,27-29

☐ 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

1

Date of the actual completion of the international search	Date of mailing of the international search report
13 August 1997	22. 08. 97
Name and mailing address of the ISA 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	Authorized officer Verleye, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int ional Application No
PCT/US 97/05257

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 691787 A	10-01-96	CN 1115150 A	17-01-96
		JP 8077706 A	22-03-96
US 5315448 A	24-05-94	AU 677999 B	15-05-97
		AU 6359394 A	11-10-94
		BR 9406002 A	02-01-96
		CA 2158021 A	29-09-94
		CN 1122177 A	08-05-96
		EP 0689751 A	03-01-96
		HU 73989 A	28-10-96
		JP 8507912 T	20-08-96
		PL 310623 A	27-12-95
		WO 9422266 A	29-09-94

Form PCT/ISA/210 (patent family annex) (July 1992)

A12

wais-concepts
Wide Area Information Server Concepts

Brewster Kahle
Thinking Machines

11/3/89
Version 4, Draft

Wide Area Information Servers answer questions over a network feeding information into personal workstations or other servers. As personal workstations become sophisticated computers, much of the role of finding, selecting, and presenting can be done locally to tailor to the users interests and preferences. This paper describes how current technology can be used to open a market of information services that will allow user's workstation to act as librarian and information collection agent from a large number of sources. These ideas form the foundation of a joint project between Apple Computer, Thinking Machines, and Dow Jones. This document is intended for those that are interested in the theoretical concepts and implications of a broad-based information system.

The paper is broken up in three parts corresponding to the three components of the system: the user workstation, the servers, and the protocol that connects them. Whereas a workstation can act as a server, and a server can request information from other servers, it is useful to break up the functionality into client and server roles. A final section in the appendix outlines related systems.

Ideas for this have come from Charlie Bedard, Franklin Davis, Tom Erlickson, Carl Feynman, Danny Hillis, the Seeker group, Jim Salem, Gitta Salomon, Dave Smith, Steve Smith, Craig Stanfill, and others. I am acting as scribe. Comments are welcome (brewster@think.com).

Table of Contents

I. Introduction

Distributing knowledge was first done with human memory and oral tradition, later by manuscript, and then by paper books. While paper distribution is still efficient distribution mechanism for some information, electronic transmission makes sense for other. This project attempts to install an electronic "backbone" for distribution of information. Some information is already distributed electronically whether it is printed before it is consumed or not. This project attempts to make electronic networks the distribution technique for more types of information by exploiting new technology and standardizing on an information interchange protocol.

The problems that are being addressed in the design of this system include human interface issues, merging of information of many sources, finding applicable sources of information, and setting up a framework for the rapid proliferation of information servers. Accessing private, group, and public information with one user model implemented on personal workstations is attempted to allow users access to many sources without learning specialized commands. A system for finding information in the sea of possible sources without asking every question of every source can be accomplished by searching descriptions of sources and selecting the sources by hand.

An open protocol for connecting user interfaces on workstations and server computers is critical to the expansion of the available information servers. The success of this system lies in a "critical mass" of users and servers. This protocol, then, could be used on any electronic network from digital networks to phone lines.

For the information owners to make their data available over a server, they must be easy to start, inexpensive to operate, and profitable. One possible approach would be to provide software at a low price that will help those with information holdings to put their data on an electronic network. The power of the current personal workstations is enough to enable sophisticate information servicing capabilities. Charging for services can be done in a number of ways that do not entail setting up large billing operations. In this way, it is easy to set up, operate, and charge for information services.

The key ideas that the WAIS system are that information services should be easily and freely distributed, that the power of the current workstations can provide sophisticated tools as servers and consumers, and that electronic networks should be exploited to distribute information.

II. The Workstation's Role in WAIS

The personal workstation has grown to be a sophisticated computer that can store hundreds of books worth of information, multiprocess, and communicate over a variety of networks. The advanced capabilities of the workstation are used to find appropriate information for the user by contacting, probing, and negotiating with information servers. The explosion of available information may change the way we use computers since the usual approaches to information on workstations may not grow to make the new information environment understandable. The proposed mechanism involves finding information with one mechanism called "Content Navigation" whether the data is local or remote, available immediately or over time. This section details what a workstation might do to collect and present information from a variety of sources.

A. Accessing Documents with Content Navigation

Currently, the common way to find a document (or file) is the "Finder" on

the Macintosh or most other machines. This tree structure requires the user to remember where s/he has put each file. This approach works when a user is familiar with the file organization. It is also computationally efficient. To aid those that have forgotten the exact location many systems have some way to locate files anywhere in the structure based on the filename ("Find File" on the the Mac, and "find" on Unix machines). The number of potential files increases as the disk space become less expensive and networks let users access remote files. At some point, when the number of files becomes large, this organization can become unwieldy because of the amount the user has to remember. Another technique that is currently popular is to augment documents with static HyperText links 1,2. These links help users move through 500 Megabyte CD-ROMs of data without being overwhelmed. HyperText systems allows the author to provide "paths" through the document. The HyperCard system, from Apple, also has a simple content searching mechanism that helps navigate without those links. HyperText links give the author another tool to guide the user and augment the capabilities of the file system.

A different technique that would allow access to a large collection of documents based on document content and similarity can be called "Content Navigation." With this tool, documents are retrieved by starting with a question in English. A single line, or headline, would describe possible documents that are appropriate. These documents can be viewed, or used to further direct the search by asking for "more documents like that one". Each document on the disk (or some other source) is then scored on how well it answers the question and the top scoring documents are listed for the user. Since full natural language processing is currently impossible, each document type, be it and newspaper article or a spread sheet, must have some simple measure to determine how relevant it is to the question asked. For text documents a useful and powerful measure is to count the number of words in common between the question and the text. This well known technique of Information Retrieval1 can be augmented with different weighting schemes for different words or constructions. Other types of information might be retrieved with specific question formats.

Thus, documents can be found by asking the "navigator" for documents that contain a set of words. Those documents that share the most words with the question will come back at the top of the list (have the best "score"). In this system the "answer" to a question is not a single document, rather it is an ordered list of candidate documents.

Content navigation is not new; NeXT and Lotus have implemented systems for personal computers,2 many text database systems on mini-computers, and the DowQuest system using a super-computer. In general, there is no standardization yet on how these systems should be queried and used.

B. Dynamic Folders Find Information for the User

Content navigation takes a question and returns an ordered list of possibly relevant documents. The question can be further refined by giving feedback as to how relevant the documents were. The results of a question can be seen as cousin to the file folder in that it contains a list of documents. In reality, the answers to a questions might not be a "copy" of a document, but a "reference" or pointer to a document. These question and answer sessions can be saved just like a file folder can be saved. Saving a session also frees the machine to find answers when the user in not looking. This capability becomes important when some of the questions take time to answer because the data might be far away or difficult to answer. This section discusses one way to think of a saved question: a Dynamic Folder.

"Dynamic Folders" are a cross between a database query and a Macintosh folder that can give us great power in defining questions and probing databases. Text database queries respond with a list of pointers to "hit articles", in the form of titles or headlines, that might interest the user. At that point, the entire article can then be retrieved, if desired. A Dynamic Folder, similarly, has a question that is used to retrieve

headlines. Further a Dynamic Folder can be saved and viewed later. Since a folder is a also structure that holds documents so that they can be viewed later, a Dynamic Folder is a folder that has a question associated with it.. In that way a dynamic view acts like a database query in collecting pointers to interesting documents and like a folder in that it can be closed and opened at different times. A Dynamic Folder's question or "charter" acts as instructions to an active agent as to what what should be put in the folder. This charter gives the folder a mission to keep itself full of appropriate pointers to files or documents. This charter might be as simple as "all files on my personal disk that have a .c suffix", or all mail received in the last day. In some circumstances, it is important for a Dynamic Folder to contain pointers to a part of a file rather than to an entire file. Treating parts of files as first class documents is important in systems that group many independent documents in one file, such often done with e-mail or news articles. In this way, "documents" and "files" are slightly different.

A Dynamic Folder's contents will change when the charter has changed, at fixed intervals, or when external events happen. The user interface should indicate how current the folder is if it does not always appear up to date. Ideally, when a user changes the charter of a Dynamic Folder, the contents would reflect this instantly. This is possible for local searches and some remote searches. Sometimes, however, changes in the available documents can not be reflected immediately. This is the case when indexing the contents of new files can take a while and is done in the background. Some folders should be updated periodically to reflect new documents in remote databases. For example, a folder that uses the New York Times should be rechecked every day for new articles. Other updates to folders could be done based on events happening such as a new document being stored on the local disk. This could cause all appropriate folders to see if that file is appropriate to add to the contents.

C. Using Information Servers

Information servers sit on a network and answer questions. A server, whether local or remote, has some database that can be queried and retrieved from. These servers can be easily accessed by a workstation over a network with a standard protocol (see the Protocol section) using the Content Navigation tool to state queries and the Dynamic Folders to hold and coordinate the responses. In this way, a user's sources of information can be seamlessly expanded past the contents of the workstation without an extra conceptual burden on the user. Part of the "charter" of a Dynamic Folder, then, is the servers that it should use. This combination of tools extends the reach of the user while maintaining a consistent view of information. The capabilities of the servers will be discussed more in the server section, but it is important to see at this point that the workstation can be negotiating with a large number of local and remote servers.

D. Other User Interface Possibilities

The "Dynamic Folder" is just one way to portray the results of a question. Other visual and aural possibilities have been suggested including draw from newspapers, books, library shelves, and sound recordings. This section touches on these possibilities.

Presenting information in newspaper format has been tried at the MIT Media Lab (NewsPeek). This approach shows not only a one-line headline, but also the writer, date, place, and first few paragraphs of the article. This format expresses importance by the size of the headline typeface, the organization of the articles on the page, and the amount of text include on the first page. Advertisements also have a place in such a presentation.

Using a book or a loose-leaf binder metaphor has been explored by the Hearst group at Apple. In this model, the inside flap of the book is used to describe the charter of the book. A table of contents is the headlines

that can be retrieved. Further, the book can have sections to it separated by tabs. An index fits naturally into this model. The Dynamic Folder is a version of this idea.

Borrowing from e-mail programs, listing the possibilities in order of importance has been the technique used by Thinking Machines and NeXT for displaying candidates. Selecting an article brought the text to another window. This interface style allows the user to mark "good" documents to further refine the question. This approach is closely related to the Babyl, Rmail, and Zmail mail handler programs(ref?).

Showing the source of documents geographically was suggested by Tom Erickson of Apple. In this approach, a world map can be used to show areas of interest. This might be a good way to initiate browsing if geographical relevance is an important factor to the user. The number of articles concerning or originating from an area can be displayed conveniently.

Presenting documents like books on a shelf is a familiar metaphor to librarians. Information about the age of the book, how frequently it has been used, its size, if it is a picture book or monograph or pamphlet, when it was published (by the age of the font) are easily gathered with this presentation. Grabbing a book and looking at it, or looking on the shelves close by are natural reactions in this metaphor. I do not know of any attempts to display information in this way.

Generating a recording of a person reading the top articles can be useful for commuters. With simple skip forward and back capabilities, this might be an effective way to deliver a custom newspaper to someone driving a car. This ideally would be done with a CD player, but a cassette could be used.

The Dynamic Folder is just one possible presentation idea. This area will be an interesting area for research and prototypes.

E. Advantages of Remote and Local Filtering

When a user subscribes to a remote server, the user can get a complete copy of the database unfiltered, or can instruct the server to filter the documents remotely. Printed newspapers are delivered whole whether all of it is relevant or not. With electronic distribution, one can imagine a user asking for all sports articles but not the business articles. A query is a form of filter that works at the server. A broad query will retrieve a large number of documents that can be further filtered on the personal workstation. The system and protocols can handle filtering at either or both ends.

Local filtering can done by the content navigation on the local disk after the documents have been retrieved. The quality of this filtering will depend on the quality of the content navigator on the local workstation. The filtering might be able to use knowledge about the user that is impractical to deliver to a server. Local filtering gives the user the most flexibility, but it could entail too much communication or too much disk space. How much filtering will be done on the local workstation has tradeoffs that must be made on a server-by-server basis. If the filtering is done locally, then the workstation might have a subscription to a server that periodically retrieves the newest articles.

Remote filtering can reduce the communications bandwidth as well as possibly offer better filtering. A server can have better filtering capabilities because it can be database specific as opposed to the workstation's navigator that must be quite general. Remote filtering, just like an interactive query, in initiated by using a question.

As communications, storage, and local computation costs change relative to each other, different filtering structures might make sense.

F. Local Caching of Documents

Documents that have been retrieved from a server are stored locally on the personal workstation in a cache. A cache is a computer architecture term meaning fast, short term storage that helps speed up access by remembering commonly used entries. In this context, a cache would store documents that the user has seen or might want to see so that access to those documents would be faster and easier. A fundamental property of computer caches is that the use of the cache only makes access faster rather than changing any functionality. In certain circumstances, it might be useful to relax this constraint, but this will be seen below. Most interactive queries will only use the cache and local files because the cache will be up-to-date on its information subscriptions. The cache is very important to make queries interactive even though data may have come from remote servers.

The document cache would be stored locally but is shared between all Dynamic Folders. In this way, an article retrieved for one reason could be used in another folder without requiring two copies. A central repository would have to be managed carefully to keep the most relevant articles but not to overload the storage. A quota might be allocated to the cache, and a cache manager would make decisions about what should stay and what should go. Sometimes the user should be consulted, and other times it can be done automatically. The cache manager should keep header information on how each document in the cache such as: (1) what server the document came from, (2) how big it is, (3) if it was looked at by the user, (4) when it was retrieved, (5) what folders point to it, (6) if the user asked to keep it permanently, (7) what the user thought about it , (8) how hard is it to retrieve it again, (9) how to retrieve it again, if at all. If a document has been deleted from the cache, but it is still being referenced by a Dynamic Folder, the header information should be preserved enough to be able to retrieve the document again. In this way, deleting a document is not a catastrophe.

Since a cache can hold many of the articles seen by a user, the cache is useful in answering retrieving documents based on "I read an article once about..." (In a study of libraries users of scientific journals, about 60% of the articles read were found by browsing, and about 30% were from remembering that they saw it before and they wanted to know more). Supporting this type of question is important for a WAIS interface. The cache can help here by storing all the documents that the user has read. If the cache can not store all of them then it can be instructed as to what type of documents it should keep on hand.

G. Local Scoring of Competing Servers

Since a Dynamic Folder can get its data from many servers, it must merge this data and present it in a meaningful way to the user. While servers that rate other servers can help determine which server's answers should be valued (see the ***ratings section), these servers only rate the server as a whole and not the individual documents. Furthermore, the article could be very good, just not appropriate to the question. One way to order the responses presented to the user could be based on a "score" that is assigned to each response by the server. Each server might, for instance, judge the appropriateness of its response to the question on a scale of 1-10. These lists from multiple sources could be merged in that order (weighted by the ratings of the servers) and presented to the user. Unfortunately, since a server would want its data to be used, it has every incentive to rate all articles with at 10. Thus, determining how much to trust the server's scores will improve the selection of documents presented to the user.

One possible solution to this problem is to have local scores for servers to augment what the server says. Therefore, if a server always says "this answer is worth 10" and the user never finds it useful, then the personal workstation can lower the trustworthiness of that server's estimation of

itself. Saying 10 all the time is the equivalent to crying wolf; if it does it too often, then users will stop listening. In such a scenario, then, all responses from that server could be degraded by 30% before it is used to merge in with the other database's responses. On the other hand, other databases may underrate themselves and should be boosted. This local scoring can be used to indicate a user's satisfaction with a database and could be used by others to help in rating it. Further, this local score could be used to determine if the server is worth subscribing to or keeping its articles in the cache.

H. Budgeting the User's Time and Money

Since the users workstation will be spending the users money to contact some servers, a system of accounting and budgeting must be installed so that users get the most value for their money. The trade-offs of time and money can be tricky to try to represent, so a simple system should be attempted first. The underlying premise is that the computer knows how much it cost to use different services. This can be easy if a service charges for connect time. If a service is reached with a long distance phone call, however this rate could be difficult. (Maybe a server should be set up that knows how much the phone companies charge for different calls.) Further, if a server charges based on the question, there must be a way for the protocol for limiting the amount spent.

Some queries are going to be very important to happen quickly or they are of no use. Working this into the interface can be tricky.

Ideas towards automatic budgeting are still quite primitive. They involve global limits per month, or limits per Dynamic Folder, etc. Should the workstation enforce the limits? Who can override the limits? We need ideas on this one.

III. The Server's Role in WAIS

Servers sit on networks and answer questions. Successful servers will have some expertise or service that others find useful whether it is primary information, information about other servers, or a service. A file server, a printer, and a human travel agent can all be viewed as forms of servers. This section describes how servers might be used in a Wide Area Information Servers system.

A. Probing Information Servers

Finding documents (or more generally, information) on one's personal disk is important, but finding relevant information on remote systems would extend the usefulness of personal computers. Currently, most remote database accesses are not integrated with the workstation model using a "glass terminal" interface which does not use the power of the workstation. Some servers look like extensions of the file system and do integrate naturally (such as Sun NFS and AppleShare) but do not provide ways documents based on content. One of the major goals of the WAIS project is to integrate wide area requests in a natural way with local area requests. This section will describe how different information servers could be integrated into this model.

Using the Dynamic Folder, the user creates lasting questions that can collect answers over time from a variety of sources. The charter of a Dynamic Folder includes what sources should be used, which might include the local disk, local special purpose information servers (such as dictionaries etc), AppleShare file servers, and remote databases or WAIS (see the Examples of Information Servers section).

A wide area information server is a computer which provides information on a particular theme to other computers. Servers sit on a network, such as

the phone system, the Internet, or X.25, accept connections from other servers or users in order to answer questions in a standard format.

Each information server can be queried at the time the charter is updated, or it can be periodically polled for new information. Newspaper servers, for instance, should be polled to find new articles, while dictionary servers should only be queried once because repeatedly asking the same question is pointless. Thus, the user's workstation keeps information about each server.

While a map, a spread sheet, an airline ticket, or music might be the appropriate reply to a specific query, the initial question is stated in English. A charter (or question) about "Beethoven's choral works" might result in an article from the encyclopedia server, a schedule of concerts from the newspaper server, and recordings from a music server. Depending on the networks used, some responses might be impractical to retrieve, but the architecture allows for any type of information exchange.

A Dynamic Folder can also be used as an information server to other workstations. This simple form of server can enable others to share information easily. This capability should be put into the user interface to encourage people to exchange information. A Dynamic Folder could be "exported" or made available to those that know about it, or "advertised" by adding it to a directory of services. If it is entered into a directory (which is just another information server) then an English description of the folder should be included.

An information server is probed by putting it in the sources section of the folder's charter. These servers can be varied in size, content, and location. Using content navigation and Dynamic Folders we have an metaphor for accessing many types of information servers.

B. Examples of Information Servers

Information servers, in the broadest sense, answer questions on a particular subject on some network. Electronic networks have been used for years to distribute information in this way. Some of the servers that are available on local area networks have been:

File serving
Printers
Compute servers (such as supercomputers)
FAX
Mail services and archives
Bboard services
Modem pools
Shared databases
Text searching and automatic indexing
CD-ROM servers
Conferencing
Dictionary lookup
User's locations (finger)
Scanners/OCR
35mm Slide output

Wide area networks open up other possibilities for other services. Some services will be offered because they are expensive to offer on a local basis, because it requires some special expertise or machinery, or because it is used infrequently on a local basis. Examples of wide area services that could be offered: Current newspapers and periodicals Movie and TV schedules with reviews Bulletin boards and chat lines Archive searching through public databases Hobby specific information (ie sports scores or newletters) Mail order shopping services Banking services Talk services, bboard, and party line styles Directory information (both online sources and Yellow Pages) Scientific papers Government databases, such as patents, congressional record, and laws.

Library catalogs (eg. OCLC)
Weather predictions and maps
Usenet and Arpanet articles
Maps with driving directions included
Software distribution
Remote conferencing
Voice mail
Music and video archives
Pizza ordering

What services will be popular or commercially successful can only be guessed.

C. Navigating through the "Directory of Services"

The Directory of Servers is an information server maintains a database of available servers and how they are contacted. Like the white pages of the phone system the directory should be easy and cheap to use and include everyone. Equally important, this directory is easy to add to. Thus, people with something interesting to offer are encouraged to add their service to the directory.

A directory entry, however, should give enough information to understand what the service is and how to connect to it. This entry is similar to a yellow-pages entry in the phone book since the goal is to advertise the service. A directory entry includes: (1) Description of server in English, (2) the parent server if it is a subsidiary of a larger server, (3) related servers, (4) public encryption key, and (5) contact information including networks and contact points, (6) cost information. A local workstation would keep extra information such as: (1) locally determined "score" reflecting usefulness (2) subscription information (if any), (3) user comments, and (4) time of last contact.

This information would be used to help determine when and if the server should be contacted, and how the responses should be handled.

Navigating in the sea of servers to find new servers can be done using the content navigation technique. In this way a question on classical music would retrieve documents as well as directory entries. This could be done by storing the directory entries on the local disk (in the cache) and accessing it just like local documents based on the appropriateness of the description. Thus retrieving the document would show all the directory information. In that way, a user that is unaware of a certain server would be presented with a description of that server with a listing of its hits for the current question so that s/he could effectively evaluate its potential value of the server. If the server is added to the list of servers for that viewer, then it would be queried in the future. Maintaining an up-to-date list of services in the cache naturally falls out of content navigation and Dynamic Folders model because a directory of services viewer would have the charter to keep itself up-to-date on directory changes, and can be probed using content navigation. The directory of services viewer would list the remote directory server or servers in the sources slot. That way, the directory is kept locally and is fast to access.

Cost and availability information can help guide the workstation to alert its user to new choices of databases. If a new server appears in the directory that is cheaper than the current server, then it could be suggested as an alternative server. This can be complicated to do well, but the benefits of not having the user cull through new directory listings can warrant work in this direction. As Stewart Brand said, "One of the problems with a market based system is that you are always shopping!" Hopefully, the workstation can do some of the mindless part of comparing servers.

Directories are classically owned and serviced by the communications companies. In this role, the communications company is an unbiased party

that profits from the use of the system as a whole. Further, communications companies generally take on a teaching role to get users familiar with the system and aid those with problems. This has been true with AT&T with the telephone, the different phone companies with the 900 numbers, and the Network Information Center for the Arpanet. Whether the communications companies take over this role or not, the directory must be supported by some organization or organizations that profit from the use of the system.

D. Servers that Rate other Servers

With a large number of servers, it would be nice to know which ones are sponsored by crooks, and which ones are gems. The directory of information servers necessarily accepts all applications for inclusion, just as the white pages do. Unlike the white pages, however, is a description (or advertisement) of the server is included which can be misleading with the result that users are charged for contacting fraudulent servers. Some protection can be offered by independent servers that rate or grade other servers. These servers can serve somewhat the same roles as Consumer Reports, Better Business Bureau, and movie reviewers. This section describes what rating services might do within the WAIS system.

Just as people use movie reviewers to help them select what movies to see, rating services can help in the selection of quality servers. Servers that provide "grades" or reviews of other servers will become useful as the number of servers grow. These ratings can come in many forms such as a numeric grade, formatted reviews that can be used with filters, or a free form discussion. Thresholds can be used by different users to ensure that a server is proven before it is used. This threshold might best be used in conjunction with the cost so that even worthless, but free databases might be tried.

These rating services can come from professional servers or from friends. A user does not have to subscribe to just one rating service, since a combination might be more useful. Combining information from multiple ratings is an interesting topic for exploration. Creating the ratings server with personal ratings could also be automated somewhat since, each user's workstation keeps track of how frequently a server has been found useful. This information, or any other, can be exported so that other people can select servers that are commonly used.

Numeric ratings of servers can be merged into the user interface by helping order the documents suggested to the user. Therefore, for some user, articles from the Wall Street Journal might get better scores than a similar article in the People's Enquirer. This information could also be displayed by the color of the headline, for instance, so that unrated services would not be overly penalized.

Just as movie goers start to trust a reviewer that has agrees with them on past movies, users will trust rating services that they agree with. Selecting a rating service based on this criteria can have some interesting effects. The rating services that a user has agreed with the most will single themselves out automatically. Users with similar tastes would then find each other. With such an arrangement, one could be lead to find other servers just because other users have liked it whether it is logically related to the common servers or not. This is an automated form of the "if you like this book, then you will like this other book" system. Further, if two users like many of the same things, then they might want to meet.

A generation of server speculators can also arise. Since servers are paid based on people using them, a ratings server will want people to use them often. If agreeing with user's past evaluations is criteria for using a ratings service, then predicting what people will like will be a lucrative business. If a server turns out to be right, then it will be used more. This type of speculation is closely related to the stock market advisers that have become notable of late. A difference would be that this form of speculation is trying to predict what will be interesting to people.

E. The Role of Editors

One of the conclusions from the NewsPeek personal newspaper project at MIT (I hear) was that editors still had a place in the electronic age by reviewing and selecting certain articles as important. Unlike the rating services, an editor grades specific articles as whether they are important. These grades are similar in many ways to the rating services and might be able to be merged.

A Dynamic Folder might have a charter like: "any article from the front page of the New York Times" which is a command to use what the editor suggests the top articles are. Like the rating services, this can be independent of the sources of the articles and combine the information from multiple sources.

A form of editor server would be if users kept track of their favorite articles and put them in a Dynamic Folder and exported it for others. This way, many favorite servers might emerge and articles could be selected based on friend's suggestions.

Automatically figuring out what the user thought of a document is tricky. Clues as to what the user thought of it are: (1) how many folders point to it, (2) if the user read it, how much of it, and for how long, (3) has the user ever taken any information from it to be used in other documents, (4) has the user ever referenced it.

This type of information could greatly improve users ability to deal with the flood of available information. Furthermore, throwing away all the thoughts a user has about a document is denying others of that mental effort.

F. Markets and Hierarchies: Using Silicon Valley

Currently there are several online information providers and many online information "brokers". Brokers provide the connections between the workstations and the information providers (such as PC-link and Compuserve). Sometimes these brokers have services of their own such as electronic mail and bulletin board services. These brokers try provide a complete information environment by providing access to servers. This structure forces a new information server to be connected to many brokers to have their product used since many users only use a few brokers.. The airline reservation program Eaasy Sabre, for example, is available on 20 of these broker networks. The approach of WAIS is to have an open system of interconnection between users and servers where the brokers can act as a server, but is not an all encompassing information environment. With an open system we have a "market" of information servers rather than a controlled environment or a "hierarchy"1 . Such a structure could open up the field to many more servers and more sophisticated front-ends.

A market based approach would only standardize on the interchange formats leaving different companies free to store and service queries in any way deemed efficient. The user interfaces, similarly, are free to evolve to fit users needs. Since the protocol is not "terminal oriented" (as most systems are today), it frees the computers on either side to be sophisticated in serving the user.

Rapid evolution of a technology can happen in a market system if the structure is designed well. As long as the protocols are flexible enough to start with, and a procedure for changing the protocol is established, then the components will evolve independently by companies seeking to gain a competitive edge.

Silicon valley is an example of a market based system that led to rapid evolution of hardware in the 1970's and software in the 1980's. As the needs of the customers became understood and defined, larger companies that

had good marketing and service reputations could make the profitable components without the help of the plethora of small companies. Information servers is an innately niche-based market given the diverse information needs of the population. Furthermore, the industry is more like a service industry than a manufacturing one because of the continual need for updates and new information. For these reasons, the silicon valley structure can help in the rapid evolution of this market.

The key is to have enough users to make the servers profitable. Since, small companies can not wait long before investment turns to profit, achieving early income is important to get the system started. A "critical mass" of users might form if the first interfaces were inexpensive or free, and a few useful servers were available.

G. How Server Companies Can Make Money

If the WAIS system is to take off, then server companies must be able to make money. Companies that offer servers can make money by billing users directly, using credit cards, or by using 900 numbers to have the phone system bill the users. Direct billing is difficult to set up and can be expensive to operate, but large providers might want to do this. Credit card billing has been a popular one for information providers. This enables any network to connect the user to the server and then the user is charged for use of the server. Typically, the first transaction with a server is a negotiation of how payment will occur and the allocation of a password for future transactions. This could be automated in the WAIS system so that the workstation could know how much the costs will be and keep a total of everything spent. A risk with the credit card system is that a credit card number in the hands of a crook can enable him to make fraudulent charges. With the potentially large number of WAIS systems, this might prove dangerous. Ratings services might be able to help weed out the fraudulent information providers (if any).

Another approach is to use a phone company service over 900 numbers. When a company is assigned one of these numbers, callers are charged per minute of phone conversation and these charges appear on the phone bill every month. Typically the phone company gets 50% of the revenue from this and the charges range from $.10 to $2 per minute (PacBell gets $.25 for the first minute and $.20 thereafter). This approach eliminates the need to have a negotiation of credit card information and limits some of the risks of disclosing a credit card number. On the other hand, the charge for billing is high. Another limitation is that one must use the phone system to connect with the server.

In any case, there is very low overhead in starting a server and earning money. All one needs is a phone, a computer, and some desirable information. This is crucial to the success of the system.

All methods of billing are likely to be used and should be supported by the WAIS interfaces.

IV. The Protocol's Role in WAIS

"... they have all one language; and this is only the beginning of what they will do; and nothing that they propose to do will now be impossible for them"

Genesis 11:6

To connect a workstation to a server requires a communication network and a language to talk. The communications network can be anything that allows computers to communicate such as modems, Internet, or digital phone networks. A protocol is the language used to relate questions and receive answers between the workstations and servers. This section describes some of the issues involved in this protocol.

A. Open Protocols Promotes Wider Acceptance

It is important to the success of this system to have an open protocol that allows users to connect with servers. Several models for how to create an open standard have been tried, such as: have a company own it and license it (Adobe, for instance), have a university develop it (X Windows, for instance), have a standards organization bless it (Common Lisp, for instance), and simply make the specification available and declare is open (IBM PC, for instance). Each approach has advantages and disadvantages. The key point is that certain attributes be adhered to.

1. The companies that are developing the protocol must be open to using existing standards, and not feeling that new protocols should be protected.

2. A system for enhancements to the standard should be set up. Standards committees are often used for this.

3. The standard should be able to transmit data in a variety of formats. There are many emerging multi-media standards. A good standard will be able to transmit these information standards.

4. The query part of the protocol should be able to accept different formats of queries. Queries might, eventually, have multimedia expressions. These should be free to evolve with periodic standardization.

5. The query must have some method to transmit cost restrictions and time-outs. It should also be able to handle query forwarding while avoiding circularities.

An idea for a query language is to use English that is restricted by the constructs that are understood by the servers. As systems become more complicated, they can handle more English constructs. In this way, future server systems can get more information from a query and produce more appropriate responses, simpler systems might use the words in the query without parsing the structure of the query. This approach would allow the servers to change, while the not changing the human interface and the protocols. The English language approach has been very successful for untrained users of the Dow Jones DowQuest system.

The overall success of this system largely depends on how well these protocols work and how they are made available. There is a standard that could solve part of the problem: NISO Z39.50-1988. This standard can help with connecting to servers, delivering queries, and getting responses back. It does not specify the query language or the format of the retrieved records. Other standards may be able to aid other communications needs.

B. Hardware Independence

Since this system depends on an open protocol rather than a particular implementation, the workstation, servers, and communications systems can all be made up of various hardware technologies that would evolve in time. This independence fosters an appropriate use of all hardware pieces, and a freedom to compete to produce the best components.

Each personal workstation platform has attributes that are appropriate to exploit differently. These can be used to make tailored user interfaces. Further, a competition for the best caching and selection criteria should emerge which will hopefully settle into a good general standard. As personal workstations start to handle audio and video, these can be retrieved with the WAIS system if the bandwidth is available.

Nintendo, for instance, makes a home computer that connects to the television that is installed about 25% of all American homes. They are providing information services to 150,000 Japanese households using this technology. This might be an attractive front-end to a WAIS system.

The server computers will range from personal workstations to

supercomputers. Most databases are under 1 gigabyte so they can be stored and processed with a personal workstation unless there are a very large number of users. Supercomputers will be used in applications where there is a large amount of data or there are a very large number of users. Supercomputers can offer superior query handling by doing extensive work on each query.

The communications systems used should be any that are locally available. The bandwidth requirements for text can be satisfied with current phone systems using modems. As advances in bandwidth and connectivity emerge, such as X.25, ISDN, and InterNet; then the range of offerings from the information providers should go up.

Since no component is centralized, this system is free to be established anywhere in the world. Other more centralized systems, such as Minitel, have had difficulty in expanding outside of France. This system should encourage independent regions to set up a compatible system because of the availability of software for servers and workstations.

C. Protecting the User's Privacy

"Electrical information devices for universal, tyrannical womb-to-tomb surveillance are causing a very serious dilemma between our claim to privacy and the community's need to know."

Marshall McLuhan, Media is the Message

To encourage users to trust their personal machines with their data and interests, we must be sure to protect people's sense of privacy. As machines start to learn more about their users and start to contact other machines on their user's behalf, the dangers to privacy are significant. There are technical as well as legal issues involved. This section will cover the technical issues in protecting privacy (any good ref for the legal side?).

There is no easy way to protect a personal workstation if an intruder can get at the keyboard. Since the workstation acts on behalf of the user the potential damage that could be done by a crook at the controls would be worse than is currently possible. Since users will be leaving their computer on all the time so that it can contact servers and be used by other servers, we lose the security of the computer being off at night. One way around this might be to able to turn off input from the user while leaving the computer on to contact servers over the network. If a user knows that she is never around at night or on weekends, then this profile might help lead the system to not trust off hour use and require a password. The assumption so far in personal computers is that the machine stays in a secure physical environment and all protection must be directed to network connections. This is not a safe long term solution, and should be thought through carefully.

Other risks are involved when dealing with networks. There are problems with intruders, spies, and forgers. An intruder will try to read, modify, or destroy data that the user did not intend to leave accessible. Spies will watch the traffic from a user to determine the servers contacted and the content of the messages. A forger will copy password information to act like a different user.

Network intruders can be prevented from reading unwanted data by the user only exporting certain Dynamic Folders to become servers for the outside world. A question is whether we want "group" access as well as "world" access as in the Unix file system or some other layered approach. A Dynamic Folder only contains pointers to information. If the information is on the local disk, should that be accessible by a remote machine? Should those files be protected from being read? If the information came from a remote database, should the requester be required to get it from the source even if a copy is on site? What are the copyright issues here? Spies can watch communications networks and collect passwords and credit

card data if this information is sent in clear text (not encrypted) as well as read the data. A public key system makes sense in this application because the directory information can include a key. Public key systems are those that everyone can lock a message (encrypt) for a recipient, but only the recipient can read it. Presumably the public key system would be used in establishing a connection and a special key for the conversation would be established. Current public key systems are too compute intensive to be used for large volumes of data. A conversation key could be used with DES or some other encryption system that is easier to compute (usrEZ software has a product that runs at 30k characters/second on a MacII). Adoption of such a system early in the WAIS development would ensure that this type of protection is assumed in modern information systems.

Forgers can be foiled with a system of authentication. Authentication is important when the charges are high or when the system is used for ordering goods. One solution is to use a public key signature system that is easy to implement using the public key system (ref the Public Key papers). A signature is passed so that only the sender could have created it.

V. Conclusion: Why WAIS will Change the World

Historically, when the distribution of information became easier or less expensive, and explosive growth in learning occurred. Wide area information servers are a new way to distribute information. Since anyone with a personal computer, a phone, and some information can be a server, people are free to create and distribute their work in ways that paper distribution made impractical. The current electronic databases, in general, do not have a standard for interchange. Just as the railroads were owned and controlled by relatively few people current database brokers control access and hence the production of data. The highway system was not owned by anyone and the incremental cost to start a new business was very low. Small businesses flourished partly because of this. WAIS systems, similarly, have very low initial costs and low distribution costs which can pave the way to many servers in a short time.

Since the WAIS system is founded on computer to computer communications, new servers that just learn from other servers and produce useful information or analysis can become profitable. Such a server could be thought of as "smart" and the better servers will learn from other servers and from its own mistakes. Thus a distributed "smart" intelligence can be formed.

BBoard systems have not produced any astounding works of literature, I suggest, because it is difficult to reference older works. If older works were easy to find and reference, then people would be more inclined to make better entries. Better entries would get more references and be used more. No BBoard systems, that I know of, make this easy. Since editors, content searching, and archiving are all fundamental parts of the WAIS architecture, we stand a better chance of high quality works being produced.

A large server, or sage, has a role in this distributed system because it can infer correspondences between many pieces of information. Further, large servers will have many users that it can learn from. Users will teach a server what is important just by using the server. Thus a large server will be the place that great new ideas will be created based on lots of existing information. This new form of intelligence, that is formed out of many participating people and machines, is an exciting prospect.

VI. Related Documents

Blip Culture Hypermedia, Harry Chesley, Apple.

Catalyzing a Market of Wide Area Information Servers, Brewster Kahle.

Wide Area Information Server Demonstration, Brewster Kahle and Charlie Bedard.

Electronic Markets and Electronic Hierarchies, Thomas Malone CACM June 1987.

Introduction to Modern Information Retrieval, Gerald Salton, Cornell. McGraw Hill.

Parallel Free-text search on the Connection Machine, Stanfill and Kahle CACM Dec 1986.

VII. Appendix: Comparisons to Existing Systems

There are always precedents to any system, this one included. Some are academic and some are commercial; some are computer oriented and some are human services; some are special purpose and some are generally useful.

A. Compuserve;(of Columbus Ohio, 1-800-848-8199) is a phone based service with about 1000 services with 500,000 PC subscribers. It includes BBoards, hobby services, home shopping, email, multiuser online games, etc. Interestingly, they have contracted with the government to accept Export License Application transactions and other user interface functions. They have "Personal Newspaper" products and deliver data from many publishers. They own a lot of the underlying communication system, but are afraid of ATT and Baby Bells. They are building sophisticated user interfaces for the PCs and MACs.

Compuserve is owned by H&R Block and charges by the minute. They handle their own billing. They have recently bought most of their competitors (The Source, Access, Software House of Cambridge, and Collier-Jackson of Tampa Florida) and are making a fortune. They turned a profit in 4th quarter fiscal 1985 and by the end of fiscal 1986 it recorded a profit of $1.7 million on $100 million revenues and 300,000 users.

Compuserve is the closest model and can be easily accessed with the WAIS system. On the other hand, WAIS helps you find the database you are interested in, does not use a terminal interface (you use your PC with all of its speed), and WAIS offers subscriptions to services where your PC will keep itself informed automatically. Most importantly, WAIS is not "owned" by anyone and is free to grow independently from a centralized company.

(For more technical information I have a book of their services, Thinking Machines has an account, and I have a series of articles describing their business activities.) B. Minitel; in France is an outgrowth of the phone company. As an alternative to phone books, users were offered terminals for their homes. Many people took the terminal. By all reports it has been a very popular system. A 1986 news report said: "The directory for Minitel services is now the size of a phone directory for a small city, evidence that Minitel is a success." George Nahon, managing directory of Intelmatique: "Then need to create a market of users emerged as a prerequisite for a service." One reports speculated that France has put about $500 million into the system by 1986.

Their interface is a terminal type interface and the servers are both human and machine. [Europe is the most exciting continent for information services. It seems that they take this very seriously, while the US

government has yet to take the bold steps of investment and standardization.] C. NetLib; is a free Unix utility for distributing files through the email. Anyone that has access to the servers via electronic mail can make inquiries and file requests. This system currently has about 100 (a guess) collections world-wide and is growing. In 1987, about 10,000 requests per month were serviced. The bulk of the offerings are software programs rather than raw data. Since no charges are made for queries or requests this system is used by academics and researchers. ATT and Argonne labs are supporting this work.

The automatic reply system (remote-machine-to-local-machine rather than remote-machine-to-local-human interface) in NetLib is similar to the WAIS system. WAIS, however, is not centered solely around EMail as a transport layer; it uses the phone system as well for interactive use. Also, WAIS would help find databases that are relevant and handle the queries and requests through a more "user friendly" interface. (For more on NetLib see Distribution of Mathematical Software via Electronic Mail in Communications of the ACM May 1987) D. Switzerland system; Still assessing this system.

E. Lotus and NeXT text system Both Lotus and NeXT have text searching systems that are similar to Thinking Machine's Dow Jones system, but are based on local data (LAN based). Since disks hold close to 1 gigabyte these days, and the entire CM at Dow Jones holds 1 gigabyte, we are close in scope but not performance. On the other hand, a PC will serve its 20 users adequately and the new daily information can be effectively distributed from Dow Jones and other places. Lotus seems to be getting into the information distribution business and is writing software to process that data locally.

These companies see themselves as critically involved in this area. I believe cooperating with them is in our best interest.

F. Information Brokers Many companies act as brokers to other information providers. Often these services will offer electronic mail and bulletin boards. These private systems rarely communicate with each other. The systems that I know of are listed below. If anyone has any information on these or other companies, please tell me.

AppleLink(Personal Edition)	1-800-227-6364	getting info
Delphi	1-800-544-4005	getting info
Dialcom, Inc.	1-800-435-7342	
GE Information Services	1-800-433-3683	getting info

This company services the fortune 500 companies with network and processing services using Honeywell and IBM mainframes. They lease lines from ATT and provide an environment for their customers including network services and value added filtering and massaging of data.

GEnie	1-800-638-9636	getting info
IBM Information Network	1-800-IBM-2468 ext 100	
INet 2000/TravelNet	1-800-267-8480	bad number
Inet	1-800-322-INET	
NWI	1-800-624-5916	

Quantum Computer Services since 1985, privately held, "multimillion dollars" official commodore info service. Has been supported by commodore.

PC-link	1-800-458-8532	IBM PC product
Q-Link	1-800-392-8200	Commodore product
America online		Mac product

Snet	1-800-272-SNET	Dept AA
The Source	1-800-336-3366	
StarText	1-817-390-7905	
Travel+Plus	1-800-544-4005	
US videotel	1-713-323-3000	
Western Union EasyLink	1-800-779-1111	Dept 31

Minitel Services	1-914-694-6266
Omnet/SCIENCEnet	1-617-265-9230

Other systems that I would like to find out more about: Holland system, Prodigy, Knight Ridder, Audio Tex, Airline Reservations system, Hospital Ordering System, Verity, Personal Newspaper (Media lab), Information Lens (Media Lab), SuperText.

G. Hypertext Hypertext and WAIS share many attributes for accessing textual information. In some sense, WAIS is an attempt at a large-scale hypertext system by allowing links to be deduced at run-time and across many databases stored in many places. Since servers provide pointers to documents, a pointer to a document can be put in a document and retrieved at a later time. Thus document pointers can be thought of as a crude form of hypertext link. This form of deducing hypertext links through content navigation might lead to interesting paths that are tailored to a particular user. Automatic systems will never replace the value of having users suggesting links. Suggested links can be added directly to the documents (as in most hypertext systems) or then can be made available in a distributed manner through the favorites databases. In this way, users that found certain articles to be similar or usefully viewed together can put them in a folder and export it as a database. One might ask, "Does anyone have these documents grouped in a server, and if so, what other documents are in that server?" These databases could then be used by others as evidence that they belong together. By combining many people's groupings, one can navigate through large number of documents in potentially interesting ways in a hypertext style.

1 Nelson, Ted. Literary Machines.

2 HyperCard by Apple (ref?)

1 Salton, Gerald. Introduction to Modern Information Retrieval, McGraw Hill. 1989.

2 NeXT calls theirs the Digital Librarian, and Lotus calls theirs Megellan (sp?).

1 Malone, Thomas, et al. Electronic Markets Electronic Hierarchies, CACM June 1987 Volume 30, number 6.

Open Digital Rights Language (ODRL)

Version: 0.8
Date: 2000-11-21
This Version: <http://odrl.net/ODRL-08.pdf>
Previous Versions: <http://odrl.net/ODRL-07.pdf>
Editor: Renato Iannella <renato@iprsystems.com>

0 Status

This document is an early draft and a work-in-progress and may be updated and/or replaced by other documents at any time.

The intention is to promote this draft document amongst multiple communities interested in the expression of Digital Rights Management statements and semantic interoperability across these communities.

ODRL will be standardised via an appropriate, open, and non-competitive organisation with an open process for the future maintenance of the standard. ***ODRL*** has no license requirements and is available in the spirit of "open source" software.

Comments are welcome to the editors from all interested parties.

Change Bars indicate modifications from Version 0.7

1 Overview

Digital Rights Management (DRM) involves the description, layering, analysis, valuation, trading and monitoring of the rights over an enterprise's assets; both in physical and digital form; and of tangible and intangible value. DRM covers the digital management of rights - be they rights in a physical manifestation of a work (eg a book), or be they rights in a digital manifestation of a work (eg an ebook). Current methods of managing, trading and protecting such assets are inefficient, proprietary, or else often require the information to be wrapped or embedded in a physical format [HIGGS].

A key feature of managing online rights will be the substantial increase in re-use of digital material on the Web as well as the increased efficiency for physical material. The pervasive Internet is changing the nature of distribution of digital media from a passive one way flow (from Publisher to the End User) to a much more interactive cycle where creations are re-used, combined and extended ad infinitum. At all stages, the Rights need to be managed and honoured with trusted services.

Current Rights management technologies include languages for describing the terms and conditions, tracking asset usages by enforcing controlled environments or encoded asset manifestations, and closed architectures for the overall management of rights.

The Open Digital Rights Language (***ODRL***) provides the semantics for DRM in open and trusted environments whilst being agnostic to mechanisms to achieve the secure architectures.

1.1 The Bigger Picture

It is envisaged that ***ODRL*** will "plug into" an open framework that enables peer-to-peer interoperability for DRM services. (See [ERICKSON] for an overview of this area). However, ***ODRL*** can also be used as an mechanism to express rights statements on its own and to plug into existing DRM architectures, for example, the Electronic Book Exchange [EBX] framework.

The editors consider that traditional DRM (even though it is still a new discipline) has taken a closed approach to solving problems. That is, the DRM has focused on the *content protection* issues more than the *rights management* issues. Hence, we see a movement towards "Open Digital Rights Management" (ODRM) with clear principles focused on interoperability across multiple sectors and support for fair-use doctrines.

The ODRM Framework consists of Technical, Business, Social, and Legal streams as shown in Figure 1.

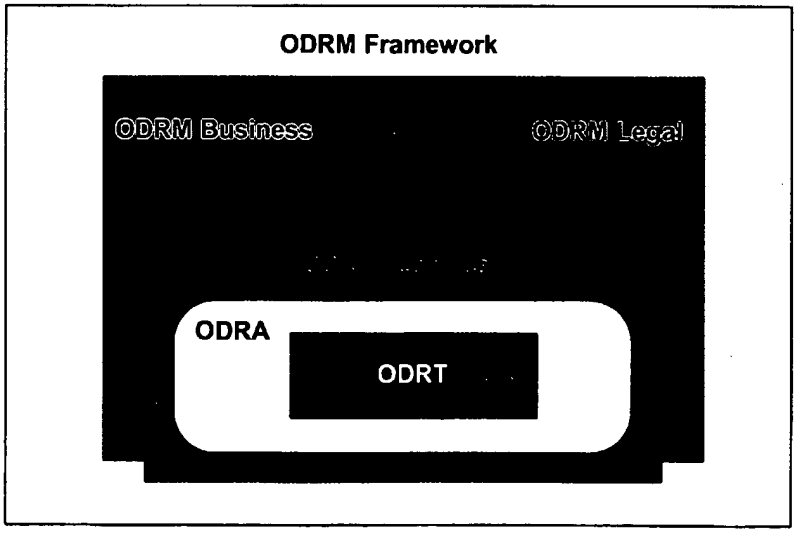


Figure 1. ODRM Framework

The ODRM Technical stream consists of an Architecture (ODRA), Trading Protocol (ODRT) and Protection (ODRP) mechanisms with ***ODRL*** clearly focused on solving a common and extendable way of expressing Rights assertions within this Architecture.

The ODRM Architecture exists in other forms that are specific to other communities needs, such as Privacy metadata. Hence, ODRA can be

achieved by abstracting and reusing such architectures to enable trusted metadata expressions about digital assets.

1.2 About this Specification

This document, along with its normative references, includes all the specification necessary for the implementation of interoperable ***ODRL*** applications.

The key words *must, must not, required, shall, shall not, should, should not, recommended, may,* and *optional* in this specification are to be interpreted as described in [RFC2119] which defines the significance of each particular requirement.

Examples used in this document are for demonstration purposes only.

2 ODRL

ODRL complements existing analogue rights management standards by providing digital equivalents, and supports an expandible range of new services that can be afforded by the digital nature of the assets in the Web environment. In the physical environment, ***ODRL*** can be used to enable machine-based processing for Rights management.

ODRL is a standard vocabulary for the expression of terms and conditions over assets. ***ODRL*** covers a core set of semantics for these purposes including the rights holders and the expression of permissible usages for asset manifestations. Rights can be specified for a specific asset manifestation (format) or could be applied to a range of manifestations of the asset.

2.1 Scope

ODRL is focused on the semantics of expressing rights languages. ***ODRL*** can be used within trusted or untrusted systems for both digital and physical assets. However, ***ODRL*** does not determine the capabilities nor requirements of any trusted services (eg for content protection, digital/physical delivery, and payment negotiation) that utilises its language. Clearly, however, ***ODRL*** will benefit rights transactions over digital assets as these can be captured and managed as a single transaction. In the physical world, ***ODRL*** expressions would need an accompanying system with the distribution of the physical asset.

ODRL defines a core set of semantics. Additional semantics can be layered on top of ***ODRL*** for third-party value added services.

ODRL does not enforce or mandate any policies for DRM, but provides the mechanisms to express such policies. Communities or organisations, that establish such policies based on ***ODRL***, do so based on their specific business or public access requirements.

ODRL depends on the use of unique identification of assets. This is a very difficult problem to address and to have agreement across many sectors and is why identification mechanisms and policies of the assets

is outside the scope of ***ODRL***. Sector-specific versions of ***ODRL*** may address the need to infer information about the asset manifestation from its unique identifier.

ODRL model is based on an analysis and survey of sector specific requirements (models and semantics), and as such, aims to be compatible with a broad community base. ***ODRL*** aims to meet the common requirements for many sectors and has been influenced by the ongoing work and specifications/models of the following groups:

- <indecs> [INDECS]
- Electronic Book Exchange [EBX]
- IFLA
- DOI Foundation [DOI]
- ONIX International [ONIX]
- MPEG
- IMS
- Dublin Core Metadata Initiative [DCMI]
- Propagate Project [PROPAGATE]

ODRL proposes to be compatible with the above groups by defining an independent and extensible set of semantics. ***ODRL*** does not depend on any media types as it is aimed for cross-sector interoperability.

2.2 Foundation Model

ODRL is based on a simple, yet extensible, model for rights management which involves the clear separation of Parties, Assets, and Rights descriptions. This is shown in Figure 2.

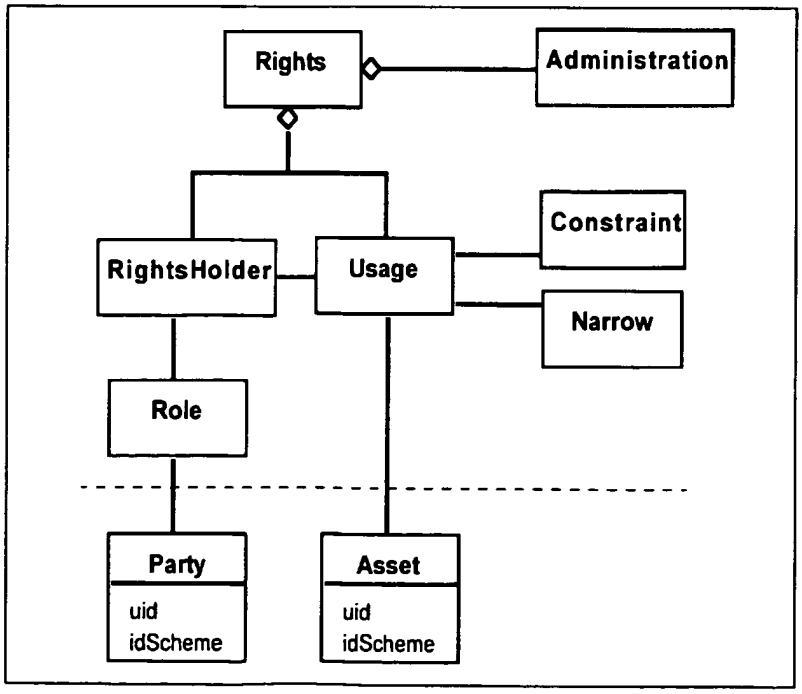


Figure 2. ODRL Foundation Model

The Rights entity consists of Usage, Constraint, Narrow, and RightsHolder which together enable the expression of digital rights over the identified Asset and their Rights Holders (parties). The Parties' Role with respect to their entitlements can also be expressed.

The description of the Party and Asset entities is outside the scope of ***ODRL***. What is in scope is that these entities must be referenced by using unique identification mechanisms (such as [URI], [DOI], [ISBN] etc).

The Asset entity (sometimes referred to as a Work, Content, Creation, or Intellectual Property), is viewed as a whole entity. If the Rights are assigned at the Asset's subpart level, then such parts would require to also be uniquely identifiable. However, ***ODRL*** can specify constraints on subparts of the asset.

The Rights entity also consists of an Administration entity that captures the responsible parties and valid dates of the Rights expression.

Complete and formal semantics for the ***ODRL*** Foundation Model properties and attributes are specified in Section 3.1 "Foundation Semantics" on page 12.

2.2.1 Example

The ***ODRL*** Foundation Model can be expressed using XML. A pseudo-example is shown below:

```
<rights>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <usage>
      <usage-type>
         ...
         <constraint> ... </constraint>
      </usage-type>
      <usage-type>
         ...
         <constraint> ... </constraint>
      </usage-type>
      ...
   </usage>
   <narrow> ... </narrow>
   <rightsholder>
      <party>
         ...
         <role> ... </role>
      </party>
      ...
   </rightsholder>
   <admin>
      <party> ... </party>
      <datetime> .. </datetime>
   </admin>
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.3 Rights Usage Model

ODRL supports the expression of Rights Usages. This is the recognised set of allowable usage rights over the Asset. This is shown in Figure 3.

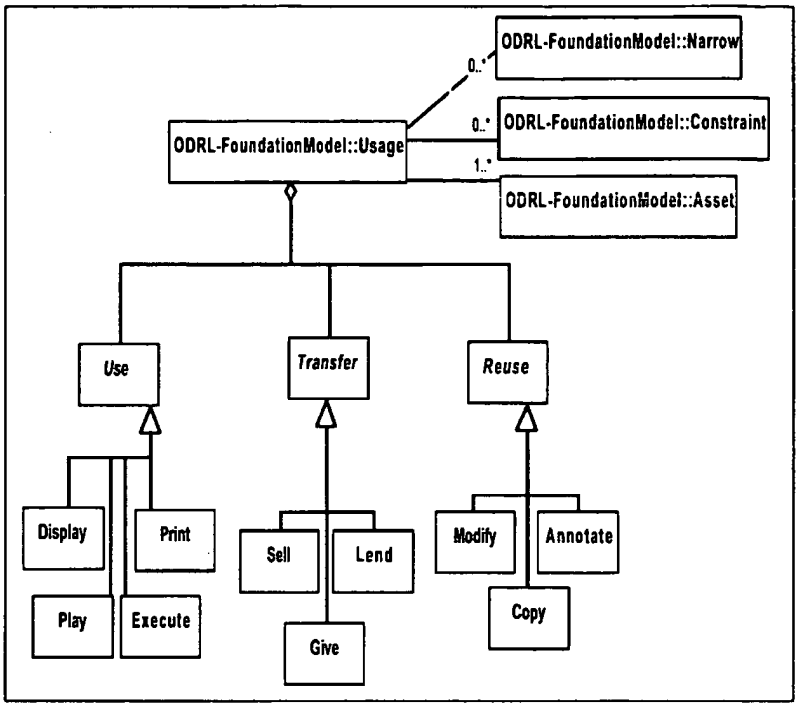


Figure 3. ODRL Usages Model

The Usage entity consists of an aggregation of three abstract entities:

- Use - indicates a set of usages in which the Asset can be consumed (realised with: Display, Print, Play, Execute).
- Transfer - indicates a set of usages in which the rights over the Asset can be transferred (realised with: Sell, Lend, Give).
- Reuse - indicates a set of usages in which the Asset (or portions of it) can be re-utilised (realised with: Modify, Copy, Annotate).

A Usage must be associated with one or more Assets. A Usage can be associated with zero or more Constraints. For any rights expression, all Usages are "and-ed" together including their constraints.

Important Note

A Usage Right that is not specified in any Rights Expressions is not granted. That is, no assumptions should be made in regard to Usage Rights if they are not explicitly mentioned.

Additionally, all Usages can be subject to an "Exclusivity" attribute that indicates if the constraint is exclusive or not.

Complete and formal semantics for the ***ODRL*** Usage Model properties and attributes are specified in Section 3.2 "Usage Semantics" on page 13.

2.3.1 Example

The ***ODRL*** Usage Model can be expressed using XML. A pseudo-example is shown below in which the identified asset has display, print (with constraints), and annotate rights.

```
<usage>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <display/>
   <print>
      <constraint> ... </constraint>
   </print>
   <annotate/>
   ...
</usage>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.4 Rights Constraint Model

ODRL supports the expression of Rights Constraints. This is the recognised set of restrictions on the usage rights over the Asset. This is shown in Figure 4.

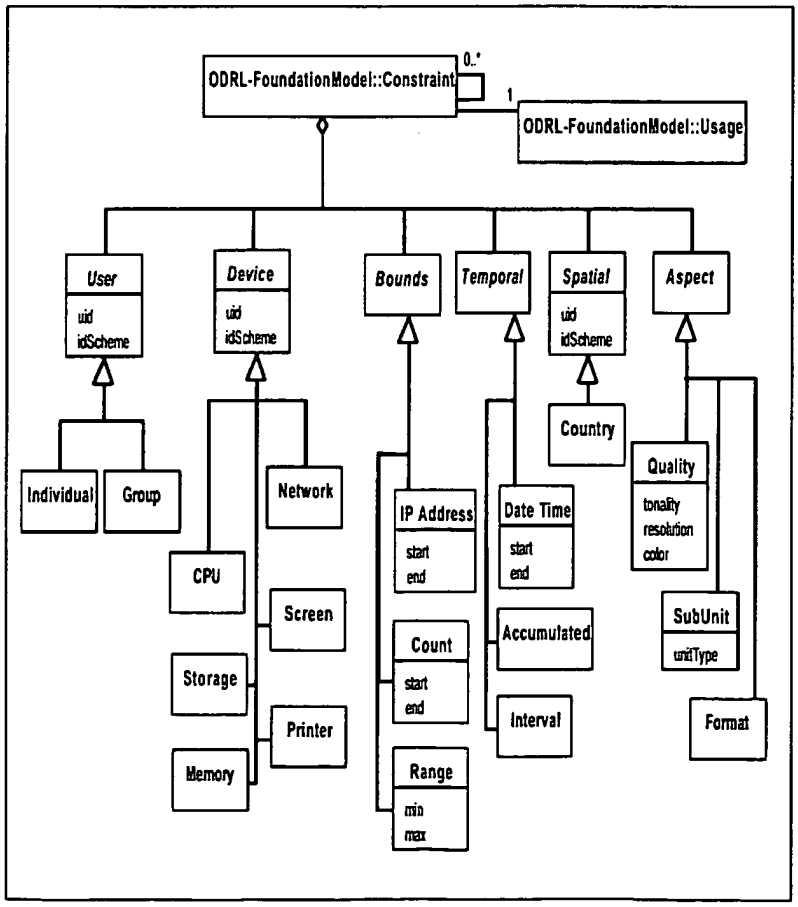


Figure 4. ODRL Constraints Model

The Constraint entity consists of an aggregation of six abstract entities:

- User - indicates a set of constraints which limits usage to identified user(s) (realised with: Individual, Group).
- Device - indicates a set of constraints which limits usage to physical devices (realised with: Network, CPU, Screen, Storage, Printer, Memory).
- Bounds - indicates a set of constraints which limits usage to a fixed number or extent (realised with: Count, Range, IP Address).
- Temporal - indicates a set of constraints which limits usage to temporal boundaries (realised with: Date Time, Accumulated, Interval).
- Spatial - indicates a set of constraints which limits usage to spatial boundaries (realised with: Country).
- Aspect - indicates a set of constraints which limits usage to distinct features of the asset (realised with: Quality, SubUnit, Format).

Additionally, all Constraints can be subject to an "Exclusivity" attribute that indicates if the constraint is exclusive or not.

A Constraint is associated with one Usage. If a Constraint appears at the same level as a number of Usages, then the Constraint applies to all of the Usages. Constraints can also have zero or more other Constraints. For Usages with multiple constraints, all contraints must be "and-ed" together and no conflicts should arise. An error must be generated if the latter is true.

Important Note

Any Constraint that is expressed but can not be performed by the consuming system, must not be granted. That is, if a system does not understand how to guarantee that a specified constraint be honoured it must not grant the Usage right at all.

Complete and formal semantics for the ***ODRL*** Constraint Model properties and attributes are specified in Section 3.3 "Constraint Semantics" on page 16.

2.4.1 Example

The ***ODRL*** Constraint Model can be expressed using XML. A pseudo-example is shown below in which the display usage right is constrained to a particular network within an identified IP address range.

```
<display>
   <constraint>
      <network>
         <constraint>
            <ipaddress start="111.222.333.1" end ="111.222.333.255" />
         <constraint>
      </network>
   </constraint>
</display>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.5 Rights Narrow Model

ODRL supports the expression of Narrowing of Rights. This is the ability to specify if the current Rights can be modified (narrowed or removed) when re-issuing the Rights expression. This is shown in Figure 5.

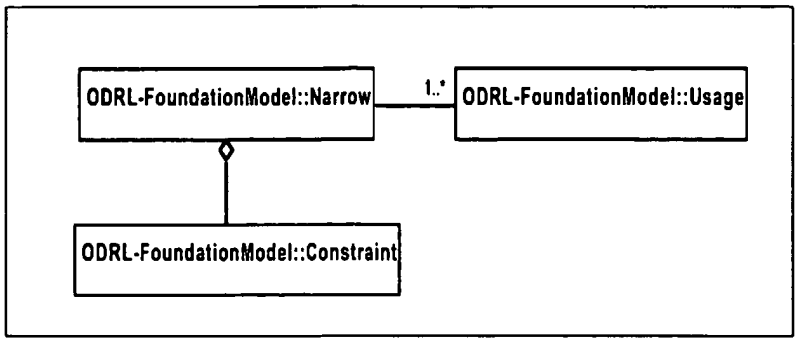


Figure 5. ODRL Narrow Model

The Narrow entity is an aggregation of one other existing entity:

- Constraint - indicates any constraints that the Narrow rights must conform to.

Complete and formal semantics for the ***ODRL*** Narrow Model properties and attributes are specified in Section 3.4 "Narrow Semantics" on page 20.

2.5.1 Example

The ***ODRL*** Narrow Model can be expressed using XML. A pseudo-example is shown below in which sell and lend transfer rights exist for the identified asset and narrow rights are applicable and are also constrained to a particular country.

```
<rights>
   <asset>
      <uid idscheme="URI">http://byeme.com/myasset.pdf</uid>
   </asset>
   <usage>
      <sell/>
      <lend/>
      <narrow>
         <constraint>
            <country>
               <uid idscheme="ISO3166"> AU </uid>
            </country>
         </constraint>
      </narrow>
   </usage>
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.6 RightsHolder Model

ODRL supports the identification of Rights Holders. This is the recognised Party, their (optional) Role, and any set of rewarding

mechanisms for the usage of the Asset for the Rights Holder. This is shown in Figure 6.

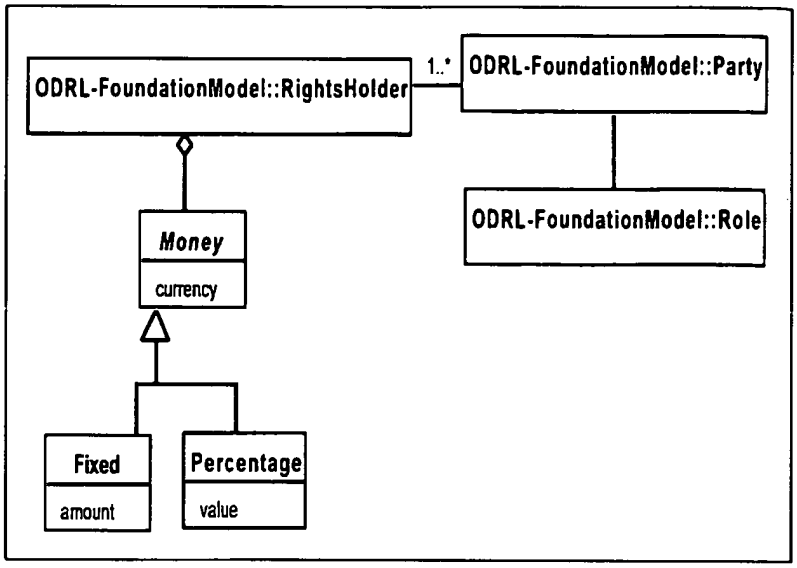


Figure 6. ODRL Rights Holder Model

The RightsHolder entity is an aggregation of one abstract and one existing entity:

- Money - indicates a set of financial rewards associated with the usage of an Asset (realised with: Fixed, Percentage).
- Party - indicates the Rights Holder and the role they play.

One or more Parties must be identified with the RightsHolder expression. The Role of the Party may also be indicated.

Complete and formal semantics for the ***ODRL*** RightsHolder Model properties and attributes are specified in Section 3.5 "RightsHolder Semantics" on page 21.

2.6.1 Example

The ***ODRL*** RightsHolder Model can be expressed using XML. A pseudo-example is shown below in which two identified Rights Holders (parties) share the financial rewards with 90% to the Author and 10% to the Publisher.

```
<rightsholder>
   <party>
      <uid idscheme="X500">c=FOO;o=Federal Library;ou=Registry;
                                    cn=Maria Brown</uid>
      <role>Author</role>
      <percentage value="90" currency="AUD"/>
   </party>
   <party>
      <uid idscheme="X500"> c=FOO;o=Federal Library;ou=Registry;
                                    cn=Bye Me Inc</uid>
      <role>Publisher</role>
      <percentage value="10" currency="AUD"/>
   </party>
</rightsholder>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

2.7 Rights Administration Model

ODRL supports the Administrative information about the Rights expression. This is shown in Figure 7.

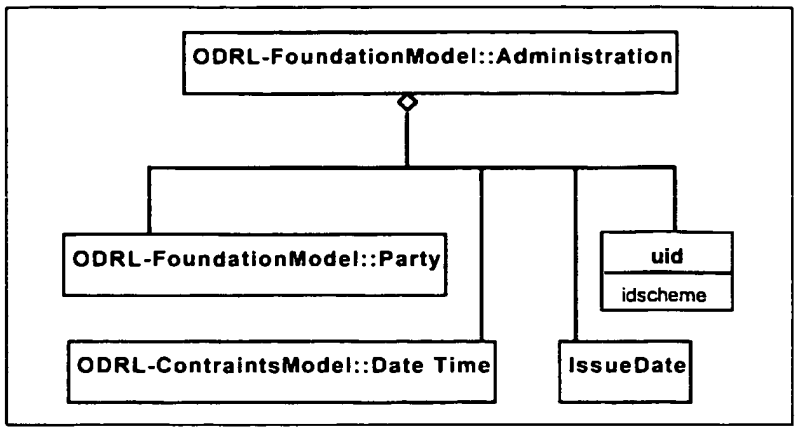


Figure 7. ODRL Administration Model

The Administration entity is an aggregation of three other existing entities and one new entity:

- Party - indicates who is responsible for maintenance of this Rights expression and the (optional) role they play.
- Date Time - indicates the valid date range for the Rights expression.
- Issue Date - indicates the date/time that the Rights expression was issued.
- UID - a unique identification number for the Rights expression

Complete and formal semantics for the ***ODRL*** Administration Model properties and attributes are specified in Section 3.6 "Administration Semantics" on page 21.

2.7.1 Example

The ***ODRL*** Administration Model can be expressed using XML. A pseudo-example is shown below in which the Rights expression is managed by the identified party (the Rights Cataloguer) and is valid for a two year period.

```
<rights>
   <admin>
      <party>
         <uid idscheme="X500"> c=FOO;o=Federal Library;ou=Registry;
                                    cn=Maria Brown</uid>
         <role>Rights Cataloguer</role>
      </party>
      <issuedate> 2000-12-31 </issuedate>
      <datetime start="2001-01-01" end="2001-12-31"/>
      <uid idscheme="URI"> http://byeme.com/mybook-rights.xml</uid>
   </admin>
   ...
</rights>
```

Complete and formal syntactical examples are given in Section 4 "Syntax" on page 22.

3 Semantics

This section details the semantics of all the properties and attributes used in the ***ODRL*** Models.

3.1 Foundation Semantics

Rights

Identifier	rights
Definition	The digital expression of intellectual property rights over an asset
Cardinality	mandatory
Content (entities)	usage rightsholder administration asset narrow

Usage Rights

Identifier	usage
Definition	A defined set of actions or operations allowed over an asset
Cardinality	mandatory
Content (entities)	use transfer reuse

Rights Holder

Identifier	rightsholder
Definition	Any party that holds any form of Rights over the asset
Cardinality	optional
Content (entities)	money party role

Asset

Identifier	asset
Definition	Any object (digital or physical) of value which rights can be assigned
Comment	Must be uniquely identifiable
Cardinality	mandatory
Content (entity)	uid - unique identifier

Party

Identifier	party
Definition	An identifiable person or organisation to which rights may be assigned over assets
Comment	Must be uniquely identifiable
Cardinality	optional
Content (entities)	uid - unique identifier role - role played by the party

UID

Identifier	uid
Definition	The unique identification number/code for the entity
Comment	The uid may be applied to assets, parties, constraints and admin entities.
Cardinality	optional
Content (attribute)	idScheme - encoding scheme used for the unique identifier value

Role

Identifier	role
Definition	The role played by the Party
Comment	The role values may be selected from existing vocabulary schemes. For example: • marc - MARC Code List for Relators [MARC] • onix - ONIX International Contributor Role Code List [ONIX]
Cardinality	optional
Content (attribute)	idScheme - identifies the vocabulary scheme used for the role value

3.2 Usage Semantics

Use

Identifier	use
Definition	A set of Usage rights pertaining to the end use of an asset
Comment	This entity is abstract and used to group common Rights Usages.
Cardinality	optional
Content (entities)	display print play execute

Use: Display

Identifier	display
Definition	The act of rendering the asset onto a screen or visual device
Cardinality	optional
Content (entities)	constraint

Use: Print

Identifier	print
Definition	The act of rendering the asset onto paper or hard copy form
Cardinality	optional
Content (entities)	constraint

Use: Play

Identifier	play
Definition	The act of rendering the asset into audio/video form
Cardinality	optional
Content (entities)	constraint

Use: Execute

Identifier	execute
Definition	The act of rendering the asset into machine-readable form
Cardinality	optional
Content (entities)	constraint

Transfer

Identifier	transfer
Definition	A set of Usage rights pertaining to the transfer of ownership of an asset
Comment	This entity is abstract and used to group common Rights Usages
Cardinality	optional
Content (entities)	sell lend give

Transfer: Sell

Identifier	sell
Definition	The act of allowing the asset to be sold for exchange of value
Cardinality	optional
Content (entities)	constraint

Transfer: Lend

Identifier	lend
Definition	The act of allowing the asset to be available for temporary use then returned
Comment	Time-based constraints are required
Cardinality	optional
Content (entities)	constraint (mandatory)

Transfer: Give

Identifier	give
Definition	The act of allowing the asset to be given away (without exchange of value)
Cardinality	optional
Content (entities)	constraint

Reuse

Identifier	reuse
Definition	A set of Usage rights pertaining to the re-utilisation of an asset
Comment	This entity is abstract and used to group common Rights Usages.
Cardinality	optional
Content (entities)	modify copy annotate

Reuse: Modify

Identifier	modify
Definition	The act of changing parts of the asset creating a new asset
Cardinality	optional
Content (entities)	constraint

Reuse: Copy

Identifier	copy
Definition	The act of extracting parts (or all) of the asset for reuse into another asset
Cardinality	optional
Content (entities)	constraint

Reuse: Annotate

Identifier	annotate
Definition	The act of adding notations/commentaries to the asset creating a new asset
Cardinality	optional
Content (entities)	constraint

3.3 Constraint Semantics

Constraint

Identifier	constraint
Definition	A restriction that applies to the Usage of an asset
Cardinality	optional
Content (entities)	user device bounds temporal spatial aspect

User

Identifier	user
Definition	Any human or organisation
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	individual group

User: Individual

Identifier	individual
Definition	An identifiable party acting as an individual
Cardinality	optional
Content (entity)	uid - unique identifier

User: Group

Identifier	group
Definition	A number of identifiable party acting as a collection of individuals
Cardinality	optional
Content (entity)	uid - unique identifier

Device

Identifier	device
Definition	Any electronic or digital equipment
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	network cpu screen storage printer memory

Device: Network

Identifier	network
Definition	An identifiable data network
Comment	If below attributes are not sufficient, then IP Address Range can also be used to limit the network.
Cardinality	optional
Content (entity)	uid - unique identifier

Device: CPU

Identifier	cpu
Definition	An identifiable system with a central processing unit (CPU)
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Screen

Identifier	screen
Definition	An identifiable display output screen device
Comment	For example, a screen reader or braille device
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Storage

Identifier	storage
Definition	An identifiable storage media device
Comment	For example, a hard disk or removable cartridge
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Printer

Identifier	printer
Definition	An identifiable hard copy printer
Cardinality	optional
Content (entity)	uid - unique identifier

Device: Memory

Identifier	memory
Definition	An identifiable memory device
Comment	For example, the clipboard
Cardinality	optional
Content (entity)	uid - unique identifier

Bounds

Identifier	bounds
Definition	The numeric limits within which any entity can function
Comment	This entity is abstract and used to group common Constraints.
Cardinality	optional
Content (entities)	Count Range IP Address

Bounds: Count

Identifier	count
Definition	A numeric count indicating the number of times the corresponding entity may be exercised
Comment	For example, the Print usage may be constraint with a count of 1 to 10 meaning that the asset can be printed once or up to 10 times. If there is no "start" or "end" value, then the count is open-ended. Integer, Floats must be supported. Note, "start" must always be less than or equal to "end" and one must always be present.
Cardinality	optional
Content (attributes)	start - the beginning of the count (inclusive) end - the end of the count (inclusive)

Bounds: Range

Identifier	range
Definition	A numeric range indicating the min/max values of the corresponding entity that the constraint applies to
Comment	For example, this is used to specify that only pages numbered 1 to 10 may be printed (using the subunit entity). If there is no "min" or "max" value, then the range is open-ended. Integer, Floats and negative numbers must be supported. Note, "min" must always be less than or equal to "max" and one must always be present.
Cardinality	optional
Content (attributes)	min - the beginning of the range (inclusive) max - the end of the range (inclusive)

Bounds: IP Address

Identifier	ipaddress
Definition	A network IP address range
Comment	There must be "start" and "end" values specified. The IP address format must be supported (Eg xxx.xxx.xxx.xxx).
Cardinality	optional
Content (attributes)	start - the beginning of the range (inclusive) end - the end of the range (inclusive)

Temporal

Identifier	temporal
Definition	The time limits within which any entity can function
Comment	This entity is abstract and used to group common Constraints. [ISO8601] Date format must be supported for all values.
Cardinality	optional
Content (entities)	Date Time Accumulated Interval

Temporal: Date Time

Identifier	datetime
Definition	A date/time-based range
Comment	If there is no "start" and/or "end" value, then the range is open-ended.
Cardinality	optional
Content (attributes)	start - the beginning of the range (inclusive) end - the end of the range (inclusive)

Temporal: Accumulated

Identifier	accumulated
Definition	The maximum amount of metered usage time
Cardinality	optional
Content	data value

Temporal: Interval

Identifier	interval
Definition	Recurring period of time in which rights can be exercised
Cardinality	optional
Content	data value

Spatial

Identifier	spatial
Definition	Any geographical range or extent
Comment	This entity is abstract and used to group common Constraints.
Cardinality	optional
Content (entities)	Country

Spatial: Country

Identifier	country
Definition	Specification of a Country code
Comment	Recommended best practice is to use the codes specified by the [ISO3166] Scheme.
Cardinality	optional
Content (entity)	uid - unique identifier

Aspect

Identifier	aspect
Definition	Any distinct feature of the Asset
Comment	This entity is abstract and used to group common Constraints
Cardinality	optional
Content (entities)	Quality SubUnit Format

Aspect: Quality

Identifier	quality
Definition	Specification of quality aspects of the asset
Cardinality	optional
Content (attributes)	tonality - the bit-depth resolution - the pixel size color - the number of colors

Aspect: SubUnit

Identifier	subunit
Definition	Specification of any sub-part of the asset
Comment	The values for the unittype attribute should be from a well known vocabulary and the source clearly identified.
Cardinality	optional
Content	unittype (attribute) constraint (entity)

Aspect: Format

Identifier	format
Definition	Specification of format(s) of the asset
Comment	The values are taken from the Internet Media Type [IMT] list.
Cardinality	optional
Content	data value

3.4 Narrow Semantics

Narrow

Identifier	narrow
Definition	Specifies modification of down-stream Rights
Cardinality	optional
Content (entities)	constraint

3.5 RightsHolder Semantics

Money

Identifier	money
Definition	Rewards in the form of financial payments
Comment	This entity is abstract and used to group common Reward types for Rights Holders
Cardinality	optional
Content (entities)	Fixed Percentage

Money: Fixed

Identifier	fixed
Definition	A fixed monetary value
Comment	The total of the Fixed values for a single asset must not exceed the Retail Price.
Cardinality	optional
Content (attributes)	amount - the value of the payment (an positive integer to two decimal places) currency - the currency for the amount (use [ISO4217] codes)

Money: Percentage

Identifier	percentage
Definition	A proportion of the value of the asset
Comment	The total of the Percentage values for a single asset must not exceed 100%.
Cardinality	optional
Content (attributes)	value - a number from 0 to 100 inclusive currency - the currency for the amount (use [ISO4217] codes)

3.6 Administration Semantics

Administration

Identifier	admin
Definition	Administrative information about the Rights expression
Cardinality	optional
Content (entities)	party datetime uid issuedate

Administration:
Issue Date

Identifier	issuedate
Definition	The date the Rights expression was issued/released
Comment	[ISO8601] Date format must be supported for all values.
Cardinality	optional
Content	data value

4 Syntax

ODRL can be expressed in [XML] (see [DTD] in Appendix A and [XML SCHEMA] in Appendix B for formal definitions). However, it is also conceivable that ***ODRL*** could be expressed in other syntaxes.

ODRL is XML Namespace aware as its primary target is use with other content description and management systems. The ***ODRL*** XML Namespace URI for *this version* is:

http://odrl.net/0.8/

The final Version 1.0 ***ODRL*** XML Namespace URI will be:

http://odrl.net/1.0/

NOTE: These URIs should be considered *experimental* until the ***ODRL*** specification is formalised by an appropriate body and the new URI is assigned.

ODRL uses XML XLink [XLINK] to refer from XML fragments to other fragments. This is used to express the relationship between the core ***ODRL*** entities such as Asset, Reward, and Usage. Such elements can be identified with the standard ID attribute then referred to via XLink's href attribute. Note, only the "xlink:href" attribute is required to be recognised to support ***ODRL*** expressions.

It is important to recognise that as the ***ODRL*** expressions become more complicated, the need to partition and express linkages (using XLink) becomes paramount in order to have manageable and reusable rights expressions. The linking mechanism allows for quite complex expressions to be generated whilst preserving the interpretability of the overall rights language.

All elements can also have optional Name and Remark elements for human-readable documentation. If the human language needs to be specified for any elements, then the use of the "xml:lang" attribute is recommended.

The XML syntax will be explained via a serious of Use Cases covering different content sectors (ebooks, image, audio, video).

4.1 Ebook Use Case #1

Corky Rossi (an author) and Addison Rossi (an illustrator) publish their ebook via "EBooksRUS Publishers". They wish to allow consumers to purchase the ebook which is restricted to a single CPU only and they are allowed to print a maximum of 2 copies. They will

also allow the first 5 pages (SubUnits) of the ebook to be viewed online for free.

The revenue split is $AUD 10.00 to the Author, $AUD 2.00 to the Illustrator and $AUD 8.00 to the Publisher.

Massimo DiAngelo from "EBooksRUS Publishers" is responsible for maintaining the Rights metadata which has a policy of one year validity on all its metadata.

The XML encoding of this in ***ODRL*** would be:

```
<?xml version="1.0"?>
<rights   xmlns="http://odrl.net/0.8/"
          xmlns:xlink="http://www.w3.org/1999/xlink">

   <admin>
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/mdiangelo-001</uid>
         <role>Rights Manager</role>
      </party>
      <datetime start="2000-07-01" end="2001-06-30"/>
   </admin>

   <asset ID="001">
      <uid idscheme="DOI">doi://10.9999/EB/rossi-0001</uid>
      <name> How to Wash Cats </name>
   </asset>

   <usage ID="002">
      <asset xlink:href="#001"/>
      <rightsholder xlink:href="#003"/>
      <display>
         <remark> Constrain to a particular CPU only </remark>
         <constraint>
            <cpu/>
         </constraint>
      </display>
      <print>
         <remark> Can only Print 2 Copies </remark>
         <constraint>
            <count start="0" end="2"/>
         </constraint>
      </print>
   </usage>

   <rightsholder ID="003">
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/crossi-001</uid>
         <role>Author</role>
         <fixed amount="10.00" currency="AUD"/>
      </party>
      <party>
         <uid idscheme="DOI">doi://10.9999/EP/arossi-001</uid>
         <role>Illustrator</role>
```

```
            <fixed amount="2.00" currency="AUD"/>
        </party>
        <party>
            <uid idscheme="DOI">doi://10.9999/EP/ebooksrus-01</uid>
            <role>Publisher</role>
            <fixed amout="8.00" currency="AUD"/>
        </party>
    </rightsholder>

    <usage ID="004">
        <asset xlink:href="#001"/>
        <remark> Allow the first 5 pages to be viewable </remark>
        <display>
            <constraint>
                <subunit unittype="page">
                    <constraint>
                        <range start="1" end ="5"/>
                    </constraint>
                </subunit>
            </constraint>
        </display>
    </usage>

</rights>
```

4.2 Ebook Use Case #2

ByeMe.Com is a distributor of ebooks. The ***ODRL*** expression below indicates that they have Sell rights for the identified ebook assets. The next Usage right is constrained to a particular individual (Mary Smith). Mary can also only print the HTML format of the asset for one to a maximum of 100 times. Mary is also limited to a maximum accumulated time of 10 hours of Display rights every 4 days.

```
<?xml version="1.0"?>
<rights    xmlns="http://odrl.net/0.8/"
           xmlns:xlink="http://www.w3.org/1999/xlink">

    <asset ID="001">
        <uid idscheme="URI">http://byeme.com/mybook.pdf</uid>
        <uid idscheme="URI">http://byeme.com/mybook.html</uid>
    </asset>

    <rightsholder ID="002">
        <party>
            <uid idscheme="X500">c=ZZ;o=Bye Me;cn=R Owner</uid>
            <role idscheme="marc"> dst </role>
        </party>
    </rightsholder>

    <usage>
        <remark> This usage associates the Distributor with the Sell rights
                 of the assets </remark>
        <asset xlink:href="#001"/>
        <rightsholder xlink:href="#002"/>
```

```
        <sell/>
    </usage>

    <usage ID="003>
        <asset xlink:href="#001"/>
        <display>
            <constraint>
                <individual>
                    <uid idscheme="X500" >c=ZZ;o=People Directory;
                                        cn=Mary Smith</uid>
                </individual>
                <accumulated> P10H </accumulated>
                <interval> P4D </interval>
            </constraint>
        </display>
        <print>
            <constraint>
                <format>text/html</format>
                <count start="1" end="100"/>
            </constraint>
        </print>
    </usage>

</rights>
```

4.3 Ebook Use Case #3

The ebook for an "Electronic Book Exchange" voucher is entitled "XML: A Manager's Guide" by "Kevin Dick". The rights owner is Addison-Wesley.

The Distributor of this book is a company called "XYZ". They have rights to Sell up to 5000 copies of the book. The have "Narrow" rights for Sell.

The licensed end user for this book is "John Doe". He has rights to view the book for 30 days before the end of 2004. He can print up to 5 copies on a "trusted printer" before the end of the year 2004. He can print up to 5 pages between page 1 and 100 every week - up to a total of 100 pages - on a "conventional printer" - before the end of 2004. He can also extract 5000 bytes every week up to a total of 1,000,000 bytes onto the Clipboard before the end of 2004. He has a right to Give the book away after one year of the usage starting.

```
<?xml version="1.0"?>
<rights    xmlns="http://odrl.net/0.8/"
           xmlns:xlink="http://www.w3.org/1999/xlink"
           xmlns:ebx="http://ebxwg.org/voucher/1.0/">

    <admin>
        <remark> Info about the Voucher </remark>
        <datetime start="2000-06-07" end="2001-06-07"/>
    </admin>

    <asset ID="Ebook-0001">
        <remark> The product ID info </remark>
```

```
    <uid idscheme="ISBN"> 0201433354 </uid>
    <name> XML: A Managers Guide </name>
</asset>

<rightsholder ID="RH-PUB-1">
    <remark> The Rights Holder (publisher) info </remark>
    <party>
        <uid idscheme="URL"> http://www.awl.com/ </uid>
        <name> Addison-Wesley </name>
        <role> Publisher </role>
    </party>
</rightsholder>

<usage ID="USE-DIST-1">
    <remark> Usage Rights for the Distributor </remark>
    <asset xlink:href="#Ebook-0001"/>

    <rightsholder>
        <remark> The Rights Holder (distributor) info </remark>
        <party>
            <uid idscheme="CG-ID"> ABDC-1234 </uid>
            <name> XYZ </name>
            <role> Distributor </role>
        </party>
    </rightsholder>

    <sell>
        <constraint> <count start="0" end ="5000"/> </constraint>
        <narrow/>
    </sell>
    <remark> Distributor also has Narrow rights over the End User
            rights </remark>
    <narrow xlink:href="EU-00001"/>
</usage>

<usage ID="EU-00001">
    <remark> Usage Rights for a typical End User</remark>
    <asset xlink:href="#Ebook-0001"/>

    <constraint>
        <remark> All usages are Licensed to Mr Doe </remark>
        <datetime start="1999-10-13"/>
        <individual>
            <uid idscheme="Lic-ID"> 92840-AA9-39849-00 </uid>
            <name> John Doe</name>
        </individual>
    </constraint>

    <display>
        <remark> View the work for 30 day period until 2004 </remark>
        <constraint>
            <accumulated> P30D </accumulated>
            <datetime end="2004-12-31"/>
```

```
    </constraint>
</display>

<print>
    <remark> Print the work up to 5 times on a trusted printer
                until 2004 </remark>
    <constraint>
        <count start="0" end ="5"/>
        <printer>
            <uid idscheme="ABC"> MyTrustedPrinterID </uid>
        </printer>
        <datetime end="2004-12-31"/>
    </constraint>
</print>

<print>
    <remark> Print up to 5 pages in any week period - between the
                pages 1 and 100 - up to a total of 100 pages - on a
                conventional printer - until 2004 </remark>
    <constraint>
        <subunit unittype="ebx:page">
            <constraint> <count end = "100"/> </constraint>
        </subunit>
        <subunit unittype="ebx:page">
            <constraint>
                <count end = "5"/>
                <range min= "1" max = "100"/>
                <interval> P7D </interval>
            </constraint>
        </subunit>
        <printer>
            <uid idscheme = "ABC"> AnyPrinterID </uid>
        </printer>
        <datetime end="2004-12-31"/>
    </constraint>
</print>

<copy>
    <remark> Extract 5000 Bytes onto the Clipboard every
        week - up to a total of 1,000,000 Bytes - until 2004 </remark>
    <constraint>
        <memory/>
        <subunit unittype="ebx:byte">
            <constraint>
                <count end ="5000"/>
                <interval> P7D </interval>
            </constraint>
        </subunit>
        <subunit unittype="ebx:byte">
            <constraint>
                <count end ="1,000,000"/>
            </constraint>
        </subunit>
```

```
            <datetime end="2004-12-31"/>
        </constraint>
    </copy>

    <remark> All the ebook to be given away (after one year) </remark>
    <give>
        <constraint>
            <datetime start="2000-10-13"/>
        </constraint>
    </give>

</usage>

</rights>
```

4.4 Image Use Case	To do...
4.5 Video Use Case	To do...
4.6 Audio Use Case	To do...

5 References

Technical Standards:

- [DCMI] Dublin Core Metadata Initiative
  <http://purl.org/DC/>
- [DOI] Digital Object Identifier
  <http://www.doi.org/>
- [DTD] Document Type Definition
- [EBX] Electronic Book Exchange
  <http://www.ebxwg.org/>
- [IFLA] Functional Requirements for Bibliographic Records
  <http://www.ifla.org/VII/s13/frbr/frbr.htm>
- [IMS] Instructional Management Systems
  <http://www.imsproject.org/>
- [IMT] Internet Media Types
  <http://www.isi.edu/in-notes/iana/assignments/media-types/media-types>
- [INDECS] Interoperability of Data in Ecommerce Systems
  <http://www.indecs.org/>
- [ISBN] International Standard Book Number
- [ISO3166] Country Names and Code Elements
  <http://www.din.de/gremien/nas/nabd/iso3166ma/codlstp1/>
- [ISO4217] Currency Names
  <http://www.xe.net/gen/iso4217.htm>

- [ISO8601] ISO (International Organization for Standardization). Representations of dates and times
  <http://www.iso.ch/markete/8601.pdf>
- [MARC] MARC Code List for Relators
  <http://www.loc.gov/marc/relators/re0002r1.html>
- [MPEG] Moving Picture Experts Group (WG 4,7,21)
  <http://www.cselt.it/leonardo/mpeg/>
- [ONIX] ONIC International V1.1
  <http://www.editeur.org/onix.html>
- [PROPAGATE] Propagate Project
  <http://www.propagate.net/>
- [RFC2119] Key words for use in RFCs to Indicate Requirement Levels
  <http://www.ietf.org/rfc/rfc2119.txt>
- [URI] Uniform Resource Identifiers (URI): Generic Syntax
  <http://www.ietf.org/rfc/rfc2396.txt>
- [VCARD] vCard MIME Directory Profile
  <http://www.ietf.org/rfc/rfc2426.txt>
- [XLINK] XML Linking Language (XLink) Version 1.0
  <http://www.w3.org/TR/xlink/>
- [XML] Extensible Markup Language 1.0
  <http://www.w3.org/TR/REC-xml>
- [XML NAMESPACE] Namespaces in XML
  <http://www.w3.org/TR/REC-xml-names/>
- [XML SCHEMA] XMl Schemas Part 1: Structures
  <http://www.w3.org/TR/xmlschema-1/>

Position Papers:

- [ERICKSON] Toward an Open Rights Management Interoperability Framework, John S Erickson.
  <http://www.oasis-open.org/cover/ericksonRT19990624.pdf>
- [HIGGS] The Nature of Knowledge and Rights Management Systems, Peter Higgs.
  <http://www.iprsystems.com/html/rights_management.html>

6 Acknowledgements

The editors wish to acknowledge contributions and feedback to this document from:

- Peter Higgs, IPR Systems
- David Parrott, Reuters
- Robert Bolick, McGraw-Hill

Appendix: A ODRL DTD (Normative)

```
<!ELEMENT rights (admin? | asset+ | usage+ | rightsholder* | name? | remark? |
narrow* )>
<!ATTLIST   rights      xmlns:xlink   CDATA   #REQUIRED
                        xmlns         CDATA   #REQUIRED >
```

```
<!ELEMENT name (#PCDATA )>
<!ELEMENT remark (#PCDATA )>

<!ELEMENT admin (name? | remark? | party* | datetime? | issuedate? | uid?)>

<!ELEMENT party (name? | remark? | uid+ | role? | percentage? | fixed? )>

<!ELEMENT uid (#PCDATA )>
<!ATTLIST uid idscheme CDATA #REQUIRED >

<!ELEMENT role (#PCDATA )>
<!ATTLIST role idscheme CDATA #IMPLIED >

<!ELEMENT issuedate (#PCDATA )>

<!ELEMENT asset (uid+ | name? | remark? )>
<!ATTLIST asset     xlink:href  CDATA    #IMPLIED
                    ID          CDATA    #IMPLIED >

<!ELEMENT usage (asset* | display* | rightsholder* | print* | play* | execute* | sell*
| lend* | give* | modify* | annotate* | copy* | constraint* | name? | remark? )>
<!ATTLIST usage     xlink:href  CDATA    #IMPLIED
                    ID          CDATA    #IMPLIED >


<!ELEMENT print (name? | remark? | constraint* )>

<!ELEMENT display (name? | remark? | constraint* )>

<!ELEMENT play (name? | remark? | constraint* )>

<!ELEMENT execute (name? | remark? | constraint* )>

<!ELEMENT sell (name? | remark? | constraint* )>

<!ELEMENT lend (name? | remark? | constraint* )>

<!ELEMENT give (name? | remark? | constraint* )>

<!ELEMENT modify (name? | remark? | constraint* )>

<!ELEMENT annotate (name? | remark? | constraint* )>

<!ELEMENT copy (name? | remark? | constraint* )>

<!ELEMENT constraint (accumulated* | interval* | datetime* | country* | quality* |
count* | range* | ipaddress* | subunit* | individual* | group* | format* | cpu* |
network* | screen* | storage* | memory* | printer* | name? | remark? )>

<!ELEMENT individual (uid+ | name? | remark? | constraint* )>

<!ELEMENT group (uid+ | name? | remark? | constraint* )>

<!ELEMENT cpu (uid+ | name? | remark? | constraint* )>

<!ELEMENT network (uid+ | name? | remark? | constraint* )>

<!ELEMENT screen (uid+ | name? | remark? | constraint* )>

<!ELEMENT storage (uid+ | name? | remark? | constraint* )>

<!ELEMENT memory (uid+ | name? | remark? | constraint* )>
```

```
<!ELEMENT printer (uid+ | name? | remark? | constraint* )>

<!ELEMENT count EMPTY>
<!ATTLIST count end   CDATA   #IMPLIED
                start CDATA   #IMPLIED >

<!ELEMENT range EMPTY>
<!ATTLIST range min   CDATA   #IMPLIED
                max   CDATA   #IMPLIED >

<!ELEMENT ipaddress EMPTY>
<!ATTLIST ipaddress   end   CDATA   #REQUIRED
                      start CDATA   #REQUIRED >

<!ELEMENT datetime EMPTY>
<!ATTLIST datetime end   CDATA   #IMPLIED
                   start CDATA   #IMPLIED >

<!ELEMENT accumulated (#PCDATA )>

<!ELEMENT interval (#PCDATA )>

<!ELEMENT country (uid+ | name? | remark? | constraint* )>

<!ELEMENT quality EMPTY>
<!ATTLIST quality   resolution CDATA   #IMPLIED
                    color      CDATA   #IMPLIED
                    tonality   CDATA   #IMPLIED >

<!ELEMENT subunit (name? | remark? | constraint* )>
<!ATTLIST subunit unittype CDATA #REQUIRED >

<!ELEMENT format (#PCDATA )>

<!ELEMENT rightsholder (party+ | fixed* | percentage* | name? | remark? )>
<!ATTLIST rightsholder ID         CDATA   #IMPLIED
                       xlink:href CDATA   #IMPLIED >

<!ELEMENT fixed (name? | remark? | party+ )>
<!ATTLIST fixed currency CDATA   #REQUIRED
                amount   CDATA   #REQUIRED >

<!ELEMENT percentage (name? | remark? | party+ )>
<!ATTLIST percentage currency CDATA   #REQUIRED
                     value    CDATA   #REQUIRED >

<!ELEMENT narrow EMPTY>
<!ATTLIST narrow xlink:href CDATA #IMPLIED >
```

Appendix: B ODRL XML Schema (Non-Normative)

NOTE: The XML Schema will become Normative when the XML Schema becomes a W3C Recommendation. Version 0.9 of ODRL will contain the XML Schema based on its current "Candidate Recommendation" status.

Authorization for Metacomputing Applications

G. Gheorghiu, T. Ryutov and B.C. Neuman
Information Sciences Institute
University of Southern California
4676 Admiralty Way suite 1001
Marina del Rey, CA 90292
grig, tryutov, bcn@isi.edu
(310)822-1511 (voice) (310)823-6714 (fax)

Abstract

One of the most difficult problems to be solved by metacomputing systems is to ensure strong authentication and authorization. The problem is complicated since the hosts involved in a metacomputing environment often span multiple administrative domains, each with its own security policy. This paper presents a distributed authorization model used by our resource allocation system, the Prospero Resource Manager [8]. The main components of our design are Extended Access Control Lists, EACLs, and a General Authorization and Access API, GAA API. EACLs extend conventional ACLs to allow conditional restrictions on access rights. In the case of the Prospero Resource Manager, specific restrictions include limits on the computational resources to be consumed and on the characteristics of the applications to be executed by the system, such as name, version or endorser. The GAA API provides a general framework for applications to access the EACLs. We have built a prototype of the system.

1. Introduction

Metacomputing is sometimes defined as the abstraction of geographically dispersed computing and communication resources (e.g. supercomputers and high-speed networks) into a single metacomputer [2]. Ideally, the user of the system is presented with a consistent and familiar interface that hides the geographic scale, the complexity and the heterogeneity.

A metacomputing system usually crosses administrative domains and involves a very large number of computing resources. Such systems have particularly sensitive requirements for security. This is one of the most difficult requirements to satisfy, due to the large scale and heterogeneity of the resources involved. The problem is complicated by the variety of representations and by the application of access control policies across multiple administrative domains.

This paper describes the authentication and authorization mechanisms and policies used by the Prospero Resource Manager (PRM [8]), a scalable resource allocation system that manages processing resources in metacomputing environments. PRM uses Kerberos [9] to achieve strong authentication and integrates a new distributed authorization model. Because different administrative domains might use different security services for authentication of principals (e.g. DCE, X.509), we designed the system to be extensible, allowing a variety of security services to be used instead of or in addition to Kerberos. The model is based on two ideas:

1. Extended Access Control Lists (EACL): conventional Access Control Lists (ACL) are extended with an optional field added to each ACL entry specifying restrictions on authorized rights. In the case of PRM, the attributes include strength of authentication, limits on the physical resources managed by the system (e.g. CPU load, memory usage) and characteristics of applications that the users are willing to run on their processors (e.g. name, version, endorser).

2. General Authorization and Access API (GAA API): we defined a common API to facilitate authorization decisions for applications. PRM invokes the GAA API functions to determine if a requested operation or set of operations is authorized or if additional checks are necessary.

Ease of use and configurability are important issues to be considered for any resource management system. For this reason, we developed a scalable mechanism based on the Prospero Directory Service to facilitate the management of the extended access control lists.

The paper is organized as follows. Section 2 describes the Prospero Resource Manager. Section 3 presents the motivation for the authorization model applied to metacomputing applications. Section 4 discusses the two components of the distributed authorization model: the EACL framework and the GAA API. Section 5 shows how the model is adapted and integrated within PRM. Section 6 describes the management of the EACL using the Prospero Directory Service. Section 7 discusses related work.

2. The Prospero Resource Manager

The design of the Prospero Resource Manager was guided by the concept of the Virtual System Model, in which resources of interest are readily accessible and those of less interest are hidden from view [8]. PRM applies this concept to the problem of allocating resources in large scale systems by dividing the functions of resource management between three types of managers: the system manager, the job manager and the node manager. Each manager makes scheduling decisions at a different level of abstraction and this separation of management enables PRM to scale as the number of managed resources increases.

Throughout the paper, we will use the term *node* to denote a processing element, be it a processor in a multiprocessor environment or a workstation whose resources are made available for running jobs. A *job* consists of a set of communicating tasks, running on the nodes allocated to the job. A *task* consists of one or more threads of control of an application, together with the address space in which they run.

2.1. The system manager

In PRM, the total collection of processing resources is divided into subsets which correspond usually to administrative domains. Each subset is managed by a *system manager* which is responsible for allocating its resources to jobs as needed. The system managers themselves can be organized in a hierarchical manner in order to avoid bottlenecks and ensure scalability.

The system manager maintains information about the characteristics of each resource it manages, together with the mapping from resources to jobs. The system manager receives status updates from node managers (e.g. availability, load information) and uses them to make allocation decisions. The system manager also responds to resource requests from job managers.

2.2. The job manager

The *job manager* offers a single point of contact for applications to request necessary resources. It hides from the application the complexity of managing the resources that have been allocated by the system manager to a particular job.

When a job is initiated, the job manager locates system managers (by using the Prospero Directory Service if available or from a configuration file) and sends resource requests. If the response from the system manager is affirmative, then the job manager allocates the resources to the tasks in the job and contacts the node manager for each resource in order execute the tasks on the appropriate processors. If a system manager refuses the request, for example when the job manager is not authorized to make the request or when there are no resources available, then the job manager contacts other system managers which can satisfy the request.

2.3. The node manager

Each resource in the system is managed by a *node manager* which is informed by the system manager about job managers that are authorized to use the resource. When the node manager receives a request from an authorized job manager, it responds by loading and executing the requested program. The node manager sends messages to the job manager upon termination or failure of tasks and to the system manager about the availability of the node for future assignments.

3. Motivation for a New Authorization Model

Metacomputing systems cover large networks connecting mutually suspicious domains, which are independently administered.

Consider the following scenario. A user logs onto a machine and wants to perform a computation on a remote machine residing in a different security domain. Let us identify the security issues to be considered:

- Establishing a trust relationship of the users between different security domains. The domain security manager must maintain an authorization database listing principals authorized to request resources belonging to this domain.
- Access control and authorization policies to protect server resources. In a wide area network, it is unlikely that sites would make their resources available to others if there are no means of protection. There should be a flexible mechanism to represent user-defined security policies, such as:
  - type and amount of resources that the node is willing to allocate, e.g. memory, processors, terminal access, access to the local files and directories, network connections

- applications that can be run on the node, e.g. name of application, version, platform, endorser
- requirement of payment or accounting for the resources consumed

- Enforcement of the security policies. There should be a mechanism for monitoring execution of the program on a particular node to ensure that the program keeps strictly to the limits imposed by the local administrators

Specification of security policies for principals from multiple administrative domains poses additional problems:

- There are multiple mechanisms for authentication of users in different domains. Therefore, there may be no single syntax for specification of principal names
- Similarly there may be no standard security policy representation. Administrators of each domain might use domain-specific policy syntax and heterogeneous implementations of the policies

Therefore our goal was to design a flexible and expressive mechanism for representing and evaluating authorization policies. It should be general enough to support a variety of mechanisms based on public or secret key cryptosystems and provide integration of local and distributed security policies.

4. Overview of the Model

Our model is designed for a system that spans multiple administrative domains where each domain can impose its own security policies. It is still necessary that a common authentication mechanism be supported between two communicating systems. The model we present enables the syntactic specification of multiple authentication policies, but it does not translate between heterogeneous authentication mechanisms.

4.1. The Extended Access Control List (EACL) framework

EACLs extend the conventional ACL concept by using conditional authorization as an extension to authorization policies, implemented as restrictions (or conditions, we use these words interchangeably) on authentication and authorization credentials. An EACL is associated with an object and lists principals allowed to access this object and the type of access granted.

The objects to be protected in PRM are hosts, but our model is suitable for applications in which the objects are files, physical devices like printers or faxes etc.

4.1.1. Notation

We will use the Backus-Naur Form to denote the elements of our EACL language. Square brackets, [ ], denote optional items and curly brackets, {}, surround items that can repeat zero or more times. A vertical line, |, separates alternatives. Items inside single quotes are the terminal symbols. The wild-card symbol `"*"` is used in an EACL just as in the UNIX environment.

4.1.2. EACL : Specification Format

An EACL consists of a set of EACL entries. Each EACL entry represents access control policies directly associated with a particular principal entity. An EACL entry specifies a principal or a list of principals, a set of granted and/or denied access rights, and optionally, any associated conditions.

```
eacl_entry::= principal {principal}
  access_rights {condition}
  {access_rights {condition}} ';'
```

4.1.3. Specification of Principals

The principal is specified according to the following format:

```
principal::=
 principal_type sec_mech  principal_ID |
 'ANYBODY'
principal_type::='HOST' | 'USER' |
 'GROUP' | 'APPLICATION'
```

where `sec_mech` and `principal_ID` are alphanumerical strings.

Different administrative domains might use different authentication mechanisms, each having a particular syntax for specification of principals. For example, an application may use Kerberos V5 [9] as an authentication service. Kerberos V5 provides secret-key based authentication and the format of the Kerberos V5 principal name is `user_name/instance@realm`. Other domains may use DCE to obtain the user's identity credentials, usually identified by a User ID and Group ID. Another domain might use client authentication in SSL, based on public-key cryptography, where principals are identified by a global name, syntactically tied to the X.500 directory. In our model, the syntax of `principal_ID` is defined according to the underlying `sec_mech`, but is tagged to identify the name space.

Principals can be aggregated into a single entry when the same set of access rights and conditions applies to all of them. `ANYBODY` is used to represent all principals regardless of authentication. Examples of principal entities are:

```
ANYBODY
USER KERBEROS.V5 kot@ISI.EDU
HOST IPaddress 164.67.21.82
GROUP DCE 8
APPLICATION CHECKSUM 0x75AA31
```

4.1.4 Specification of Access rights

Access rights are specified using the format:

```
access_rights::='<' tag ':' ['-'] value
 { tag ':' ['-']value } | '*' '>'
```

where `tag` and `value` are alphanumerical strings.

Access rights are names for types of access to the protected object. All operations defined on the object are grouped by type of access to the object they represent, and named using a tag. The right is granted when there is no "-" preceding the right specification, otherwise the right is denied. The meaning of access control rights is application specific.

4.1.5. Specification of Conditions

Conditions specify the type-specific policies under which an operation can be performed on an object. The format used for specifying access rights conditions is as follows:

```
condition::= type ':' value
```

where `type` and `value` are alphanumerical strings.

A condition is interpreted according to its type. Conditions can be categorized as generic or specific. A condition is generic if it is interpreted by the GAA API. For example: time of day, authentication mechanism, required endorsement. Specific conditions are interpreted by the application: CPU load , memory usage, applications that are to be loaded on the node.

4.1.6. EACL evaluation

The authorization language we presented supports authorization models based on the closed world model, when all rights are implicitly denied. Authorizations are granted by an explicit listing of positive access rights. The open world model, which is based on implicit granting of all rights and listing of only negative authorizations, can be represented in our model by including ANYBODY * as the final entry in an EACL. This will grant everybody all rights regardless of authentication. Denial of rights is then specified using negative rights in entries earlier in the ACL.

If one allows both negative and positive authorizations in individual or group entries, inconsistencies must be resolved according to different resolution rules. The design approach we adopted allows the ordered interpretation of EACLs. An ordered evaluation approach is easier to implement, it allows only partial evaluation of EACL and resolves the authorization conflicts. Evaluation of ordered EACL starts from the first to the last in the list of EACL entries. The resolution of inconsistent authorization is based on ordering: the authorizations or denials that have already been examined take precedence over later ones. Other interpretations are possible, but we found that for many such policies, resolution of inconsistencies was either NP-Complete or undecidable.

4.2. GAA API

The GAA API is used by applications to decide whether the subject is authorized for access. In this subsection we provide a brief description of the GAA API routines.

4.2.1. GAA API functions

1) `gaa_get_object_eacl`

This function is called before other GAA API routines which require a handle to the object EACL on which to operate. It returns a handle to the object EACL.

2) `gaa_check_authorization`

This function tells the application server whether the requested operation is authorized, or if additional application-specific checks are required. It returns the code YES (indicating authorization) if all requested operations are authorized, NO (indicating denial of authorization) if at least one operation is not authorized, MAYBE (indicating need for application-specific checks) if there are some unevaluated conditions and additional application-specific checks are required. A list of conditions is also returned, each condition being marked as evaluated or not evaluated, and if evaluated, marked as met or not met. The time period during which the authorization is granted is returned as a condition that may be used by the application.

If no operation was specified as an input, a list of authorized rights is returned as a condition that must be checked by the application. This allows the application to discover access control policies associated with the target object. The application must understand the conditions that are returned unevaluated, otherwise it rejects the request. If the application understands the conditions, it checks them against the information about the request, the target object, or other environmental conditions to determine whether the conditions have been met.

4.2.2 GAA API Security Context

The security context is an argument passed to the GAA API. Some of its constituents follow:

Identity Verified authentication information, such as principal ID for a particular security mechanism. To determine which entries apply, the GAA API checks if the specified principal ID appears in an EACL entry that is paired with a privilege for the type of access requested.

Authorization Attributes Verified authorization credentials, such as group membership, group non-membership and proxies.

Delegated Credentials Delegation is supported through inclusion of delegated credentials, such as those supported by *restricted proxies* [6].

Evaluation and Retrieval Functions for Upcalls These functions are called to evaluate application-specific conditions; request additional credentials and verify them.

5. Applying the Distributed Authorization Model to PRM

We will first discuss the integration of the EACL framework into PRM and then we will show how PRM makes use of the GAA API to enforce the policies expressed through the EACL.

5.1. EACL conditions specific to PRM

Our experience with deploying PRM on a wide scale has shown that administrators are more willing to grant access to their workstations if they can restrict access to users or organizations they trust. Administrators must also be able to specify restrictions on the specific applications that will run on their systems. These restrictions are important in the context of movement of executable or interpreted content between different systems and platforms, i.e. what is usually known as "mobile code". We have therefore introduced EACL conditions specific to this type of policy:

- name of application:

```
application_name : matlab
```

- name of interpreter, in case the application is written in an interpreted language:

```
interpreter_name : Tcl
```

- platform the application runs on:

```
application_platform : Solaris
```

- version number for the application:

```
application_version : 1.0
```

- endorser or certifying authority for the application:

```
application_endorser : Globus
```

Authorizing a user to run an application on the specific resources is often not detailed enough for system administrators. What is needed is a way to impose and enforce limits on the physical resources consumed by the applications. To specify these limits, PRM uses specific EACL conditions:

- CPU load, expressed as maximum percentage of the CPU time that an application is allowed to use:

```
cpu_load : 20%
```

- memory usage, expressed as maximum size in Kbytes that a process can occupy in main memory:

```
mem_usage : 1024
```

- machine idle time, expressed as minimum interval in minutes that the machine has to be idle before any application managed by PRM is allowed to run:

```
idle_time : 30
```

5.2. Using the GAA API in PRM

5.2.1. Creation of the GAA API security context

For communications, PRM uses calls to the Asynchronous Reliable Delivery Protocol (ARDP), which handles several security services including authentication, integrity and payment. ARDP calls the Kerberos library through a security API, requesting the principal's identity, which is placed into the security context and is passed to the GAA API. Figure 1 shows the flow of control: the system manager calls ARDP requesting the principal's identity (1); the request and the verification of the principal's identity credentials take place (2, 3, 4, 5); ARDP places the principal's authentication credentials in the security context (6a) and returns it to the system manager (6); the system manager calls the GAA API (7); the security context, containing the verified principal's identity is being passed into the GAA API (7a).

When additional security attributes are required for the requested operation, the list of required attributes is returned and obtained by the application. The application or transport may add an upcall function to the security context which is passed to the GAA API and used to request required additional credentials. Such additional credentials are requested, verified, and added to the security context by this upcall function.

5.2.2. Authorization Walk-through

Here we present two authorization scenarios. First, let's consider a request from user Joe to run `matlab` on the host `kot.isi.edu` on Monday at 7:30 PM. Assume that this host has the following ordered EACL stored in the Prospero directory service:

```
USER  kerberos.v5  joe@ISI.EDU
    <HOST : load > time_window : 6AM-8PM,
                   cpu_load : 20% ;
GROUP kerberos.v5  operator@ISI.EDU
```

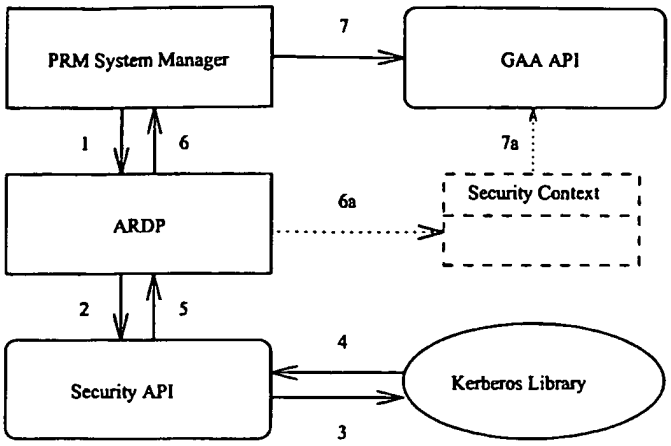


Figure 1. Creation of the GAA API security context

```
    USER  kerberos.v5  tom@ISI.EDU
    <HOST : * > <DEVICE : power_down> ;
ANYBODY <HOST : load> time_day : sat-sun,
                 time_window : 6AM-8PM,
                 cpu_load : 10% ;
```

When a job manager contacts a system manager with the request for resources, the system manager calls the `gaa_get_object_eacl` function to obtain a handle to the EACL of `kot.isi.edu`. The upcall function for retrieving the EACL for the specified object from the Prospero virtual file system is passed to the GAA API and is called by `gaa_get_object_eacl`, which returns the EACL handle. The system manager calls ARDP, which handles authentication as explained in Figure 1 and section 5.2.1. If Joe is authenticated successfully, then the verified identity credential is placed into the security context, specifying Joe as the Kerberos principal `joe@ISI.EDU`.

The `gaa_check_authorization` function is called by the system manager, which asks if Joe is authorized to load `matlab` to `kot.isi.edu`. In evaluating the EACL, the first entry applies. It grants the requested operation, but there two conditions that must be evaluated. The first condition `time_window : 6AM-8PM` is generic and is evaluated directly by the GAA API. Since the request was issued on Monday at 7:30 PM this condition is satisfied. The second condition `cpu_load : 20%` is PRM-specific. If the security context passed by PRM defined a condition evaluation function for upcall, then this function is invoked and if this condition is met then the final answer is YES (authorized).

During the execution of the task the node manager is monitoring if the task is abiding to the limits imposed on the local resources and authorization time. If the corresponding upcall function was not passed to the GAA API, the answer is MAYBE and the set of conditions is returned. Conditions are marked as either evaluated or not evaluated. In our example `time_window : 6AM-8PM` was evaluated and met; `cpu_load : 20%` was not evaluated and should be checked by PRM.

Next, we present an authorization scenario where additional credentials are needed. Let's consider a request from user Joe to run `matlab` on the host `kot.isi.edu` on Monday at 8:30 PM. In EACL evaluation, the first entry applies but does not grant this operation, since the first condition is not met. The temporary answer is NO (not authorized). The second entry grants this permission. If the security context defines a credential retrieval function for upcall, then this function is invoked and if either a group "operator" membership credential or delegated credential from user Tom for Joe is obtained, then the final answer is YES. If the credential retrieval upcall function was not passed to the GAA API, the answer is NO.

6. Managing the EACL using the Prospero Directory Service

We have mentioned in section 2 that PRM deals with scalability issues by splitting the task of managing the resources across the three types of managers. Our goal in designing a mechanism for the management of the EACL files was to enable easy sharing of a default authorization policy among node managers, while allowing customization of the policy at the level of individual hosts.

We use the Prospero Directory Service [7] to store the information associated with the EACL files. The EACL files themselves are objects stored in the Prospero directory service.

The following scenario shows how the management of

the files is accomplished:

1. The administrator of the domain whose resources are managed by a system manager running on host A creates an EACL file describing the default authorization policy which applies to the domain.

2. The administrator registers with the Prospero server. We supply a script which takes as input the location of the EACL file and creates a Prospero object representing a link to the file, together with two attributes for the link:

`SYSTEM_MANAGER A`

`EACL_DEFAULT True`

3. If the administrator of a particular host B in the domain managed by A wants to specify a local authorization policy different from the default one, a similar procedure is followed, except that the link to the local EACL file is created with the following attributes:
`NODE_MANAGER B`
`EXTEND_DEFAULT Prepend/Append/Replace`
(`Prepend` if the local policy is prepended to the default policy, `Append` if the local policy is appended to the default and `Replace` if the local policy completely replaces the default)

4. When a system manager is contacted by a job manager with a request for resources, it first authenticates the user, as was explained in the authorization scenario in section 5. Before requesting resources from a node manager running on a particular node B, the system manager retrieves the EACL file associated with that node by looking for a link with attribute `NODE_MANAGER = B`. If no such link is found, the default EACL file provided for the domain will be used and it will be obtained by retrieving a link with attributes `SYSTEM_MANAGER = A` and `EACL_DEFAULT = True`. If a link with `NODE_MANAGER = B` is found, then a second query is issued for the value of the attribute `EXTEND_DEFAULT`. If the value is `Prepend` or `Append`, the system manager will have to retrieve the default EACL file first, and then prepend or append to it the contents of the EACL file for node B (note that the distinction between the two cases `Prepend/Append` is necessary because the EACL evaluation takes into account the order of the EACL entries). If the value is `Replace`, then only the EACL file for node B will be retrieved and used.

5. After retrieval of the EACL file, evaluation of the conditions listed in the file follows, as detailed in the authorization scenario from section 5. If all the conditions are met, the job manager is allowed to use the resources on that particular host.

6. During the execution of tasks on a particular host, the node manager periodically checks whether the task is abiding to the limits imposed on the local resources. If it is not, then the task is interrupted and the job manager is notified.

7. Related Work

Nagaratnam and Byrne [5] present a model for Internet user agents to control access to client resources. This model protects client machines from hostile downloadable content and allows the client to selectively grant access to trusted agents. The authenticity of the code is based on digital signatures of principals certifying it. All access control requests are mediated by calling a security manager component and decisions are based on the user's access control specifications stored in the policy database.

The model is restricted to using the Javakey utility as an authentication mechanism based on public key digital signatures, while our model is general enough to use a variety of security mechanisms based on public or secret key cryptosystems.

Another disadvantage of that model is the duplication of common information. Each user has to maintain a database of any principals specified in the policy database and their public keys, as well as specification of groups. These databases should be properly integrity-protected. In contrast, PRM uses Kerberos to achieve strong authentication. The authentication database is maintained centrally by the KDC and stored on a physically secure machine. Our model also supports a group certification mechanism. A group server maintains and provides group membership information, and issues group membership and non-membership certificates. The certificates are placed into the GAA API security context and checked by the GAA API when making authorization decisions. There is no need for each user to maintain authentication and group specification databases locally.

The Generalized Access Control List framework described by Woo and Lam [10] presents a language-based approach for specifying authorization policies. The GACL model supports only system state-related conditions within which rights are granted, such as current system load and maximum number of copies of a program to be run concurrently. This may not be sufficient for distributed applications. Our model allows fine-grained control over the conditions.

Both restricted proxies [6] and the use-condition model [4] allow conditions and privilege attributes to be embedded in authorization credentials or certificates. These mechanisms can be readily integrated with the authorization model presented here: the restrictions or conditions caried in the proxy or certificate are evaluated by the GAA API in addition to the restrictions in the matching EACL entry.

The CRISIS architecture [1] provides ACLs that are related to the type of the protected object. For example, file ACLs list principals allowed read, write or execute access to the file, whereas node ACLs contain principals allowed to run jobs on the node. CRISIS ACLs do not support con-

straints on the resources that principals are allowed to consume. The emphasis of our work is on providing a general framework for representing security policies and facilitating authorization decisions for applications. Our model provides a uniform authorization mechanism that is capable of supporting different operations and different kinds of protected objects.

The Tivoli Management Environment (TME 10) is a commercially available system from IBM which takes a role-based approach to security [3]. TME roles are named capabilities, containing a list of objects and access permissions to those objects. Objects can have default access and can be associated with more than one role. Each role will have a different level of access to the object. Roles are defined to support a particular job function within an organization, e.g. customer support or management. Groups are assigned roles, thus giving members of those groups access capabilities to the objects assigned to those roles. The TME security model can be easily expressed by our EACL framework:

1) An EACL is associated with each object to be protected. Default access to the object is represented by including `ANYBODY default_rights` as the last entry in the EACL.

2) The object's EACL will contain entries listing groups and a set of access rights, granted by TME roles assigned to each group.

For example, in TME the group `Support_users` is assigned the `Customer_support` role which grants RWE permissions to file `/cust_supp_dir/*` In our system, an EACL associated with the object `/cust_supp_dir/*` will have the following entry:

```
GROUP sec_mech support_users FILE:R
FILE:W FILE:E ;
```

TME lacks flexibility in supporting user-defined security policies. It has a fixed predefined set of object types and generic access permissions that are available on each object type. In addition, the TME model requires the creation of a new role to include each new combination of objects and access rights. This becomes very cumbersome for systems where a large number of operations exist on various objects.

8. Conclusions

By extending the traditional Access Control Lists with restrictions on authorized rights, and by using General Authorization and General Authorization and Access API, it becomes possible to support a flexible distributed authorization mechanism allowing applications and users to define their own access control policy types, and integrating local and distributed security policies. The problem of translation of the policies is addressed by using general or PRM-specific evaluation functions. In this paper, we have omitted discussion of many practical details due to space limitation. A prototype of the presented model has been developed at the Information Sciences Institute of the University of Southern California.

Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency under the Scalable Computing Infrastructure (SCOPE) Project, TNT, Contract No. DABT63-95-C-0095, Security Infrastructure for Large Distributed Systems (SILDS) Project, Contract No. DABT63-94-C-0034, and by a grant from Xerox Corporation. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Intelligence Center and Fort Huachuca Directorate of Contracting, the Defense Advanced Research Projects Agency, the U.S. Government, or Xerox Corporation.

References

[1] E. Belani, A. Vahdat, T. Anderson, and M. Dahlin. The CRISIS wide area security architecture. *Proceedings of the 7th USENIX Security Symposium, San Antonio, Texas*, January 1998.

[2] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, Summer 1997.

[3] IBM. TME 10 security management. http://www.tivoli.com/redbooks/html/sg242021/2021fm.htm, October 1997.

[4] W. Johnston and C. Larsen. A use-condition centered approach to authenticated global capabilities: Security architectures for large-scale distributed collaboratory environments. LBNL Report 38850.

[5] N. Nagaratnam and S. Byrne. Resource access control for an Internet user agent. *Proceedings of the third USENIX Conference on Object-Oriented Technologies and Systems, Portland, Oregon*, June 1997.

[6] B. C. Neuman. Proxy-based authorization and accounting for distributed systems. *Proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh*, May 1993.

[7] B. C. Neuman, S. Augart, and S. Upasani. Using Prospero to support integrated location-independent computing. *Proc. Symp.on Mobile and Location-Independent Computing, Cambridge, MA*, pages 29–34, August 1993.

[8] B. C. Neuman and S. Rao. The Prospero Resource Manager: A scalable framework for processor allocation in distributed systems. *Concurrency: Practice and Experience*, June 1994.

[9] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, pages 33–38, September 1994.

[10] T. Woo and S. Lam. A framework for distributed authorization. *Proc. ACM Conference on Computer and Communications Security, Fairfax, Virginia*, November 1993.

A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms

TAHER ELGAMAL, MEMBER, IEEE

Abstract—A new signature scheme is proposed, together with an implementation of the Diffie–Hellman key distribution scheme that achieves a public key cryptosystem. The security of both systems relies on the difficulty of computing discrete logarithms over finite fields.

I. Introduction

In 1976, Diffie and Hellman [3] introduced the concept of public key cryptography. Since then, several attempts have been made to find practical public key systems (see, for example, [6], [7], [9]) depending on the difficulty of solving some problems. For example, the Rives–Shamir–Adleman (RSA) system [9] depends on the difficulty of factoring large integers. This paper presents systems that rely on the difficulty of computing logarithms over finite fields.

Section II shows a way to implement the public key distribution scheme introduced by Diffie and Hellman [3] to encrypt and decrypt messages. The security of this system is equivalent to that of the distribution scheme. Section III introduces a new digital signature scheme that depends on the difficulty of computing discrete logarithms over finite fields. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. Section IV develops some attacks on the signature scheme, none of which seems to break it. Section V gives some properties of the system. Section VI contains a conclusion and some remarks.

II. The Public Key System

First, the Diffie–Hellman key distribution scheme is reviewed. Suppose that A and B want to share a secret K_{AB}, where A has a secret x_A and B has a secret x_B. Let p be a large prime and α be a primitive element mod p, both known. A computes $y_A \equiv \alpha^{x_A} \bmod p$, and sends y_A. Similarly, B computes $y_B \equiv \alpha^{x_B} \bmod p$ and sends y_B. Then the secret K_{AB} is computed as

$$\begin{aligned} K_{AB} &\equiv \alpha^{x_A x_B} \bmod p \\ &\equiv y_A^{x_B} \bmod p \\ &\equiv y_B^{x_A} \bmod p. \end{aligned}$$

Hence both A and B are able to compute K_{AB}. But, for an intruder, computing K_{AB} appears to be difficult. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. For more details refer to [3].

In any of the cryptographic systems based on discrete logarithms, p must be chosen such that $p-1$ has at least one large prime factor. If $p-1$ has only small prime factors, then computing discrete logarithms is easy (see [8]).

Now suppose that A wants to send B a message m, where $0 \le m \le p-1$. First A chooses a number k uniformly between 0 and $p-1$. Note that k will serve as the secret x_A in the key distribution scheme. Then A computes the "key"

$$K \equiv y_B^k \bmod p, \tag{1}$$

where $y_B \equiv \alpha^{x_B} \bmod p$ is either in a public file or is sent by B. The encrypted message (or ciphertext) is then the pair (c_1, c_2), where

$$c_1 \equiv \alpha^k \bmod p \qquad c_2 \equiv Km \bmod p \tag{2}$$

and K is computed in (1).

Note that the size of the ciphertext is double the size of the message. Also note that the multiplication operation in (2) can be replaced by any other invertible operation such as addition mod p.

The decryption operation splits into two parts. The first step is recovering K, which is easy for B since $K \equiv (\alpha^k)^{x_B} \equiv c_1^{x_B} \bmod p$, and x_B is known to B only. The second step is to divide c_2 by K and recover the message m.

The public file consists of one entry for each user, namely y_i for user i (since α and p are known for all users). It is possible that each user chooses his own α and p, which is preferable from the security point of view although that will triple the size of the public file.

It is not advisable to use the same value k for enciphering more than one block of the message, since if k is used more than once, knowledge of one block m_1 of the message enables an intruder to compute other blocks as follows. Let

$$c_{1,1} \equiv \alpha^k \bmod p \qquad c_{2,1} \equiv m_1 K \bmod p,$$

$$c_{1,2} \equiv \alpha^k \bmod p \qquad c_{2,2} \equiv m_2 K \bmod p.$$

Then $m_1/m_2 \equiv c_{2,1}/c_{2,2} \bmod p$, and m_2 is easily computed if m_1 is known.

Breaking the system is equivalent to breaking the Diffie–Hellman distribution scheme. First, if m can be computed from c_1, c_2, and y, then K can also be computed from y, c_1, and c_2 (which appears like a random

Manuscript received February 6, 1984; revised November 12, 1984. This work was supported by the National Science Foundation under Grant ECS83 07741. The material in this paper was presented at the IEEE Crypto '84 Conference, August 1984, Santa Barbara, CA.

The author was with the Information Systems Laboratory, Stanford University, Stanford, CA. He is now with Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304, USA.

0018-9448/85/0700-0469$01.00

number since k and m are unknown). That is equivalent to breaking the distribution scheme. Second, (even if m is known) computing k or x from c_1, c_2, and y is equivalent to computing discrete logarithms. The reason is that both x and k appear in the exponent in y and c_1.

III. A Digital Signature Scheme

A new signature scheme is described in this section. The public file contains the same public keys for encrypting messages as well as verifying signatures.

Let m be a document to be signed, where $0 \le m \le p-1$. The public file still consists of the public key $y \equiv \alpha^x \bmod p$ for each user. To sign a document, a user A should be able to use the secret key x_A to find a signature for m in such a way that all users can verify the authenticity of the signature by using the public key y_A (together with α and p), and no one can forge a signature without knowing the secret x_A.

The signature for m is the pair (r, s), $0 \le r, s < p-1$, chosen such that the equation

$$\alpha^m \equiv y^r r^s \bmod p \tag{3}$$

is satisfied.

A. The Signing Procedure

The signing procedure consists of the following three steps.

1) Choose a random number k, uniformly between 0 and $p-1$, such that $\gcd(k, p-1) = 1$.

2) Compute

$$r \equiv \alpha^k \bmod p. \tag{4}$$

3) Now (3) can be written as

$$\alpha^m \equiv \alpha^{xr} \alpha^{ks} \bmod p, \tag{5}$$

which can be solved for s by using

$$m \equiv xr + ks \bmod (p-1). \tag{6}$$

Equation (6) has a solution for s if k is chosen such that $\gcd(k, p-1) = 1$.

B. The Verification Procedure

Given m, r, and s, it is easy to verify the authenticity of the signature by computing both sides of (3) and checking that they are equal.

Note 1: As will be shown in Section IV, the value of k chosen in step 1) should never be used more than once. This can be guaranteed, for example, by using as a "k generator" a DES chip used in the counter mode as a stream cipher.

IV. Some Attacks on the Signature Scheme

This section introduces some of the possible attacks on the signature scheme. Some of these attacks are easily shown to be equivalent to computing discrete logarithms over $GF(p)$. It has not yet been proved that breaking the signature scheme is equivalent to computing discrete logarithms, or equivalent to breaking the distribution scheme. However, none of the attacks shown in this section appear to break the system. The reader is encouraged to develop new attacks, or find fast algorithms to perform one of the attacks described in this section. The attacks will be divided into two groups. The first group includes some attacks for recovering the secret key x, and in the second group we show some attacks for forging signatures without recovering x.

A. Attacks Aiming to Recover x

Attack 1: Given $\{m_i: i = 1, 2, \cdots, l\}$ documents, together with the corresponding signatures $\{(r_i, s_i): i = 1, 2, \cdots, l\}$, an intruder may try to solve l equations of the form (6). Since there are $l+1$ unknowns (since each signature uses a different k), the system of equations is underdetermined and the number of solutions is large. The reason is that each value for x yields a solution for the k_i since a system of linear equations with a diagonal matrix of coefficients will result. Since $p-1$ is chosen to have at least one large prime factor q, recovering $x \bmod q$ requires an exponential number of message-signature pairs.

Note 2: If any k is used twice in the signing, then the system of equations is uniquely determined and x can be recovered. So for the system to be secure, any value of k should never be used twice.

Attack 2: Trying to solve equations of the form (3) is always equivalent to computing discrete logarithms over $GF(p)$, since both unknowns x and k appear in the exponent.

Attack 3: An intruder might try to develop some linear dependencies among the unknowns $\{k_i, i = 1, 2, \cdots, l\}$. This is also equivalent to computing discrete logarithms since if $k_i \equiv ck_j \bmod (p-1)$, then $r_i \equiv r_j^c \bmod p$, and if c can be computed then computing discrete logarithms is easy.

B. Attacks for Forging Signatures

Attack 4: Given a document m, a forger may try to find r, s such that (3) is satisfied. If $r \equiv \alpha^j \bmod p$ is fixed for some j chosen at random, then computing s is equivalent to solving a discrete logarithm problem over $GF(p)$.

If the forger fixes s first, then r could be computed from the equation

$$r^s y^r \equiv A \bmod p. \tag{7}$$

Solving equation (7) for r is not yet proved to be at least as hard as computing discrete logarithms, but we believe that it is not feasible to solve (7) in polynomial time. The reader is encouraged to find a polynomial time algorithm for solving (7).

Attack 5: It seems possible that (3) can be solved for both r and s simultaneously, but we have not been able to find an efficient algorithm to do that.

Attack 6: The signature scheme allows the following attack, whereby the intruder, knowing one legitimate signature for one message, can generate other legitimate sig-

natures and messages. This attack does not allow the intruder to sign an arbitrary message and therefore does not break the system. This property exists in all the existing digital signature schemes and can be avoided by either requiring that m has to have a certain structure or by applying a one-way function to the message m before signing it.

Given a signature (r, s) for the message (m), then

$$\alpha^m = y^r r^s \bmod p.$$

Select integers A, B, and C arbitrarily such that $(Ar - Cs)$ is relatively prime to $p - 1$. Set

$$r' \equiv r^A \alpha^B y^C \bmod p,$$
$$s' \equiv sr'/(Ar - Cs) \bmod (p - 1),$$
$$m' \equiv r'(Am + Bs)/(Ar - Cs) \bmod (p - 1).$$

Then it is claimed that (r', s') signs the message (m'). Calculate

$$\begin{aligned} y^{r'} r'^{s'} &\equiv y^{r'} \left(r^A \alpha^B y^C\right)^{sr'/(Ar-Cs)} \\ &\equiv \left(y^{r'Ar - r'Cs + r'Cs} r^{Asr'} \alpha^{Bsr'}\right)^{1/(Ar-Cs)} \\ &\equiv \left(\left(y^r r^s\right)^{Ar'} \alpha^{Bsr'}\right)^{1/(Ar-Cs)} \\ &\equiv \alpha^{(mAr' + Bsr')/(Ar-Cs)} \\ &\equiv \alpha^{m'} \end{aligned}$$

(all calculations mod p).

As a special case, setting $A = 0$, legitimate signatures can be generated with corresponding messages without ever seeing any signatures:

$$r' \equiv \alpha^B y^C \bmod p,$$
$$s' \equiv -r'/C \bmod (p - 1),$$
$$m' \equiv -r'B/C \bmod (p - 1).$$

It can be shown that (r', s') signs (m').

V. Properties of Our System and Comparison to Other Signature Schemes and Public Key Systems

Let m be the number of bits in either p for the discrete logarithm problem or n for the integer factoring problem. Then the best known algorithm for both computing discrete logarithms and factoring integers (which is the function used in some of the existing systems such as the RSA system [9]) is given by (see [1], [5], [10])

$$O\left(\exp \sqrt{cm \ln m}\right), \tag{8}$$

where the best estimate for c is $c = 0.69$ for factoring integers (due to Schnorr and Lenstra [10]), as well as for discrete logarithms over GF(p) (see [5]). These estimates imply that we have to use numbers that are about the size of the numbers used in the RSA system in order to obtain the same level of security (assuming the current value for c for both the discrete logarithms problem and the integer factorization problem). So the size of the public file is larger than that for the RSA system. (For the RSA system, each user has one entry n as his public key, together with the encryption key in the public file.)

A. Properties of the Public Key System

As shown above, our system differs from the other known systems. First, due to the randomization in the enciphering operation, the cipher text for a given message m is not repeated, i.e., if we encipher the same message twice, we will not get the same cipher text $\{c_1, c_2\}$. This prevents attacks like a probable text attack where if the intruder suspects that the plain text is, for example, m, then he tries to encipher m and finds out if it was really m. This attack, and similar ones, will not succeed since the original sender chose a random number k for enciphering, and different values of k will yield different values of $\{c_1, c_2\}$. Also, due to the structure of our system, there is no obvious relation between the enciphering of m_1, m_2, and $m_1 m_2$, or any other simple function of m_1 and m_2. This is not the case for the known systems, such as the RSA system.

Suppose that p is of about the same size as that required for n in the case of the RSA system. Then the size of the cipher text is double the size of the corresponding RSA cipher text.

For the enciphering operation, two exponentiations are required. That is equivalent to about $2 \log p$ multiplications in GF(p). For the deciphering operation only one exponentiation (plus one division) is needed.

B. Properties of the Signature Scheme

For the signature scheme using the above arguments for the sizes of the numbers in our system and the RSA system, the signature is double the size of the document. Then the size of the signature is the same size as that needed for the RSA scheme, and half the size of the signature for the new signature scheme that depends on quadratic forms published by Ong and Schnorr [6], and also Ong, Schnorr, and Shamir [7] (since both systems are based on the integer factoring problem). The Ong–Schnorr–Shamir system has been broken by Pollard and new variations are being suggested. Thus, it is not clear at the present time whether a secure system based on modular equations can be found, and hence no further remarks will be made regarding these schemes.

Note that, since the number of signatures is p^2, while the number of documents is only p, each document m has a lot of signatures but any signature signs only one document.

For the signing procedure, one exponentiation (plus a few multiplications) is needed. To verify a signature, it seems that three exponentiations are needed, but it was pointed to the author by Shamir that only 1.875 exponentiations are needed. This is done by representing the three exponents m, r, s in their binary expansion. At each step square the number $\alpha^{-1} y r$ and divide by the necessary

factor to account for the different expansions of m, r, and s. The different multiples of α^{-1}, y, and r can be stored in a table consisting of eight entries. We expect that 0.875 of the time a multiplication is needed. That accounts for the 1.875 exponentiations needed.

VI. Conclusions and Remarks

The paper described a public key cryptosystem and a signature scheme based on the difficulty of computing discrete logarithms over finite fields. The systems are only described in $GF(p)$. The public key system can be easily extended to any $GF(p^m)$, but recent progress in computing discrete logarithms over $GF(p^m)$ where m is large (see [2, 5]) makes the key size required very large for the system to be secure. The subexponential time algorithm has been extended to $GF(p^2)$ [4] and it appears that it can be extended to all finite fields, but the estimates for the running time for the fields $GF(p^m)$ with a small m seem better at the present time. Hence, it seems that it is better to use $GF(p^m)$ with $m = 3$ or 4 for implementing a cryptographic system. The estimates for the running time of computing discrete logarithms and for factoring integers are the best known so far, and if the estimates remain the same, then, for the same security level, the size of the public key file and the size of the cipher text will be double the size of those for the RSA system.

Acknowledgment

The author would like to thank a referee for including Attack 6 described in Section IV.

References

[1] L. Adleman, "A subexponential algorithm for the discrete logarithm problem with applications to cryptography," in *Proc. 20th IEEE Symp. Foundations of Computer Science* 1979, pp. 55–60.
[2] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Trans. Inform. Theory*, vol. IT-30, pp. 587–594, 1984.
[3] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 472–492, 1976.
[4] T. ElGamal, "A subexponential-time algorithm for computing discrete logarithms over $GF(p^2)$," *IEEE Trans. Inform. Theory*, this issue.
[5] A. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," *Proc. Eurocrypt 84*, to appear.
[6] H. Ong and C. Schnorr, "Signatures through approximate representations by quadratic forms," to appear.
[7] H. Ong, C. Schnorr, and A. Shamir, "An efficient signature scheme based on quadratic forms," in *Proc. 16th ACM Symp. Theoretical Computer Science*, 1984, pp. 208–216.
[8] S. Pohlig and M. Hellman, "An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance," *IEEE Trans. Inform. Theory*, vol. IT-24, pp. 106–110, 1978.
[9] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
[10] C. Schnorr and H. W. Lenstra Jr., "A Monte Carlo factoring algorithm with finite storage," *Math. Comput.*, vol. 43, pp. 289–311, 1984.

This textbook is the first for many years which covers all the principal topics of telecommunications. It provides a comprehensive coverage for those students seeking a sound foundation in the subject. In addition, its rigorous treatment of both theory and applications makes the book suitable for students intending to follow more specialist courses.

The authors have provided a sound analytical base, and developed a number of theoretical models. However, the limitations and assumptions of these models are always stressed, and emphasis is placed on relating the theory to practical problems.

The book is generously illustrated with more than 300 diagrams; each chapter contains problems, with solutions, and references to more specialized texts.

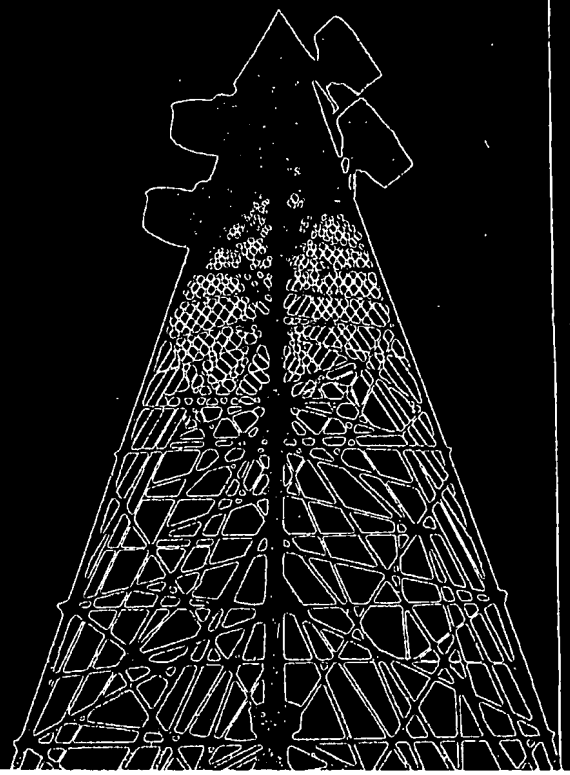
John Dunlop is a senior lecturer at the University of Strathclyde. His research interests include speech signal processing, underwater communications, digital processing of radar signals, local area communications networks and distributed processor systems.

Geoffrey Smith is a reader at the University of Strathclyde and is responsible for teletraffic engineering research. He is also involved in research in packet-switched computer communication networks and performance evaluation of several local area network mechanisms when carrying both voice and data traffic.

Van Nostrand Reinhold (UK)

TELECOMMUNICATIONS ENGINEERING

J. Dunlop & D.G. Smith



First published in 1984 by
Van Nostrand Reinhold (UK) Co. Ltd
Molly Millars Lane, Wokingham, Berkshire, England

Reprinted 1984, 1986, 1987

Typeset in 9/11 pt Times by Thomson Press (India) Ltd, New Delhi

Printed and bound in Hong Kong

Library of Congress Cataloging in Publication Data

Dunlop, J.
Telecommunications engineering.

Includes bibliographies and index.
1. Telecommunications. I. Smith, D.G. II. Title.
TK 5101.D845 1984 621.38 84–3531
ISBN 0–442–30586–9 (pbk.)

Table 10.5 Zone numbers

1	North America	6	Australasia
2	Africa	7	USSR
3	Europe	8	Eastern Asia
4	Europe	9	Far East and Middle East
5	South America	0	Spare

(USSR). Throughout each of these zones there is a linked numbering scheme that means, for example, that no subscriber in Canada has the same national number as a subscriber in the USA. Consequently, to connect to anyone in zone 1 the digit 1 is followed by the national number. A similar situation exists in the USSR. Europe is at the other extreme; there are many countries with large national networks that have nine digits in their national numbers. For these, a two-digit country code is required, and that can only be achieved by having two zone numbers allocated to Europe.

The division of the world into the zones shown in Table 10.5 is intended to be satisfactory until early in the next century, but clearly, as some large countries develop their telephone networks, some adjustment will be necessary at some future time.

The national and international numbering schemes we have discussed above are the simplest. However, in several parts of the world there are small exceptions, particularly in regard to local calls. In the scheme where the national number is used for all calls within a country, it can lead to irritation on the part of the subscriber and long set-up times for the exchange equipment. Consequently, in many countries local calls use a shorter code. For calls within the same area, the area code is omitted, and for small single-exchange areas no exchange code is used for own-exchange calls. Coupled with this last arrangement will be a very short code for calls to adjacent exchanges; these arrangements are particularly well suited to rural areas. The disadvantage of short codes is that they change with the location of the calling subscriber, and therefore a short code directory must be available in each exchange area.

10.33 Routeing Calls

The early type of switching equipment, called step-by-step or Strowger, operated by using the dialled pulses to move the selectors to the position corresponding to the digit dialled. In many ways this was an excellent system, but one major disadvantage was that it allowed no flexibility in the way calls were routed – the route was pre-determined by the dialled digits. Although some systems were modified to overcome this problem it was not until common-control equipment became widely used that the path a call took between calling and called subscribers could be chosen to allow the most efficient use of the available capacity in the system. The function of the common control in the routing process was to store the dialled digits in a register and then translate them into routing digits which would indicate to the switching

system the path to take through the network. This register – translator combination is essential to automatic trunk and international dialling schemes; it allows the telephone administration to manage the system efficiently by changing routes as circumstances alter without having to change subscribers' numbers. This therefore separates the subscriber from the system. The subscriber dials the national number from any location and the register–translator automatically selects an appropriate route.

10.34 Digital Systems

Several factors have acted to push telephony from analogue to digital working. To make such a major change, telephone administrations and equipment manufacturers have been persuaded that it is in digital operation that the future lies, and the decade from the mid-seventies has been characterized in all equipment producing countries by huge investments of manpower and plant in a race to manufacture an efficient digital telecommunications system. There are more manufacturers involved than at any other time and great financial commitments have been made in the hope that a market will be available for the many products being developed.

Traditionally the two basic elements of a telephone system were transmission and

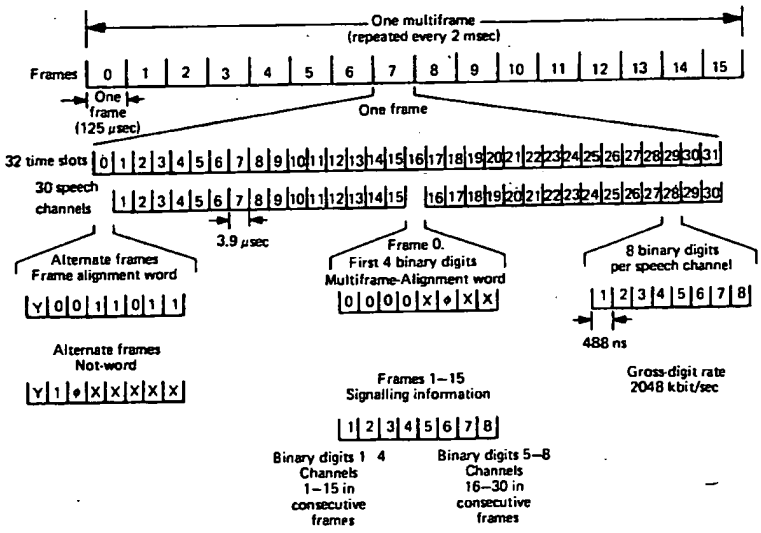


Fig. 10.37 32-frame PCM multiplex frame arrangement: *x*, digits not allocated to any particular function and set to state *one*; *y*, reserved for international use (normally set to state *one*); 0, digits normally *zero* but changed to *one* when loss of frame alignment occurs and/or system-fail alarm occurs (TS0 only) or when loss of multiframe alignment occurs (TS16 only).

switching. However, with digital operation the whole system is considered as an entity.

Following the development of PCM several countries installed PCM links between analogue exchanges. These worked in a basic 24- or 32-channel format with an encoder and decoder. We can see how the 32-channel system is formed with reference to Fig. 10.37. Strictly speaking, it should be referred to as a 30-channel, 32-time-slot system, because two of the time slots contain signalling and synchronizing information, not speech samples. The sampling rate of each speech channel determines the length of each time slot. For telephony the sampling rate used is 8 kHz, which means that the time between adjacent samples of the same channel is 125 μsec. One frame, consisting of 32 time-slots lasts for this time. A time slot is therefore approximately 3.9 μsec long. For reasons that we shall see in a moment, 16 frames are put together to form a multi-frame which has a time span of 2 msec. This is the basic unit of the PCM system applied to telephony.

Within a time slot there is the encoded information about the speech sample for that channel. In most systems it is coded into 256 ($= 2^8$) levels and there are therefore 8 binary digits within each time slot; each digit must be less than 0.48 μsec long. The technique of PCM is given in detail in Section 3.5.

The channel digits bearing speech information must be sent to the right destination and monitored for release and clear-down. Signals must therefore be associated with each channel and in PCM telephony that is done by allocating a signalling word to each channel once per multi-frame. Sufficient information for signalling can be contained in a 4-digit word so that an 8 digit word can contain two signals. Hence a signalling time slot can contain enough information for two signals.

Figure 10.37 indicates that time slot sixteen (TS16) is used to carry the signalling. In the second frame it carries the signals related to speech channels 0 and 15, in the third frame those for channels 1 and 16 and so on until frame 16 when TS16 has signals for speech channels 14 and 29. The next frame is the first of the following multi-frame and the sequence is repeated.

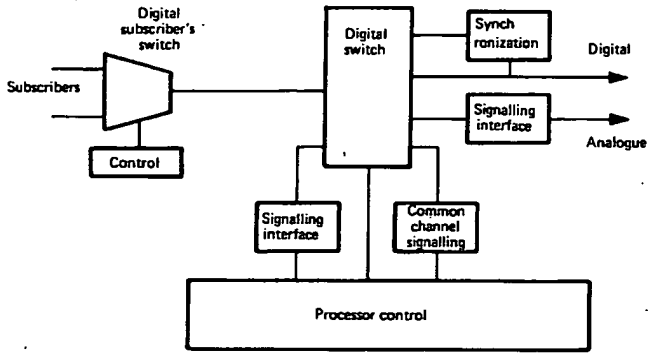


Fig. 10.38 Digital telephone exchange.

switching. However, with digital operation the whole system is considered as an entity.

Following the development of PCM several countries installed PCM links between analogue exchanges. These worked in a basic 24- or 32-channel format with an encoder and decoder. We can see how the 32-channel system is formed with reference to Fig. 10.37. Strictly speaking, it should be referred to as a 30-channel, 32-time-slot system, because two of the time slots contain signalling and synchronizing information, not speech samples. The sampling rate of each speech channel determines the length of each time slot. For telephony the sampling rate used is 8 kHz, which means that the time between adjacent samples of the same channel is 125 μsec. One frame, consisting of 32 time-slots lasts for this time. A time slot is therefore approximately 3.9 μsec long. For reasons that we shall see in a moment, 16 frames are put together to form a multi-frame which has a time span of 2 msec. This is the basic unit of the PCM system applied to telephony.

Within a time slot there is the encoded information about the speech sample for that channel. In most systems it is coded into 256 ($=2^8$) levels and there are therefore 8 binary digits within each time slot; each digit must be less than 0.48 μsec long. The technique of PCM is given in detail in Section 3.5.

The channel digits bearing speech information must be sent to the right destination and monitored for release and clear-down. Signals must therefore be associated with each channel and in PCM telephony that is done by allocating a signalling word to each channel once per multi-frame. Sufficient information for signalling can be contained in a 4-digit word so that an 8 digit word can contain two signals. Hence a signalling time slot can contain enough information for two signals.

Figure 10.37 indicates that time slot sixteen (TS16) is used to carry the signalling. In the second frame it carries the signals related to speech channels 0 and 15, in the third frame those for channels 1 and 16 and so on until frame 16 when TS16 has signals for speech channels 14 and 29. The next frame is the first of the following multi-frame and the sequence is repeated.

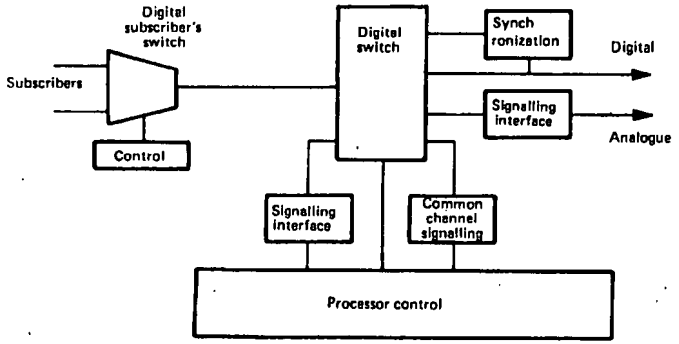


Fig. 10.38 Digital telephone exchange.

TS0 in each frame and TS16 in the first frame, carry synchronization and alignment words to ensure that the transmission and reception of the system is maintained in step.

Modern digital electronics, with its fast logic and complex integrated circuitry, has provided the techniques for producing digital switching and control systems at a cost comparable with analogue, coupled with an ability to provide better facilities, cheaper maintenance and more flexibility. The increasingly widespread use of data in various forms, and the need to send such information over large distances, has also encouraged the development of digital telecommunications and it has led to the intention to integrate together speech and data links.

Digital exchanges consist of the basic components showing in Fig. 10.38. Most of the control of the system is handled by microprocessor devices, singly or in clusters, which are driven by software. Here we are not able to discuss the huge new field of telecommunications software engineering, but it provides the most important challenge in modern system design. The software must be efficient, reliable, secure, understandable and well documented. In theory it affords a degree of flexibility in the operation of the system which is much higher than is possible in hard-wire control. However, such is the complexity of large telephone networks that software development presents the most difficult problems to designers, for it must last for tens of years and although made up of very long programs, it must be able to cope with dramatic changes in hardware technology.

In analogue exchanges the usual figure of merit used is the grade of service, or probability of blocking, but in digital switches the blocking is virtually zero. In these systems the major problems concern delay in the processing of calls caused by the processor units becoming overloaded. The analysis of such problems is very difficult and if simple queueing theory models do not apply resort must be made to computer simulation. One of the difficult tasks of the software engineer is to produce a satisfactory compromise between short efficient programs and those that are longer, more complex and more reliable and secure.

Referring again to Fig. 10.38, look first at the digital switch. It can have many structural forms, depending on the size of the system and the technology used, but as an example we will consider the common time–space–time (TST) configuration. TST is a shorthand for the time-switch, space-switch, time-switch arrangement shown in Fig. 10.39(a).

The time switch is split into several units, each having M PCM links of L channels, as Fig. 10.39(b). Consequently, if the time switch is non-blocking it will have an outlet highway of $N = ML$ time slots. The space switch is square with R inlet highways and R outlet highways. The purpose of the TST unit is to allow a particular call, which occupies a specific channel into one of the time switches, to be connected to a particular outlet channel. Basically the switching is between highways on either side of the space switch. Each highway has N time slots and in order for a particular call to be connected say from $H1$ to $H3$ it must find a time slot which is free in both highways. This slot may not be the same as the required incoming and outgoing slots for the call, and so some time delay, provided by the time switches, is necessary. The method used to produce the delay again depends on

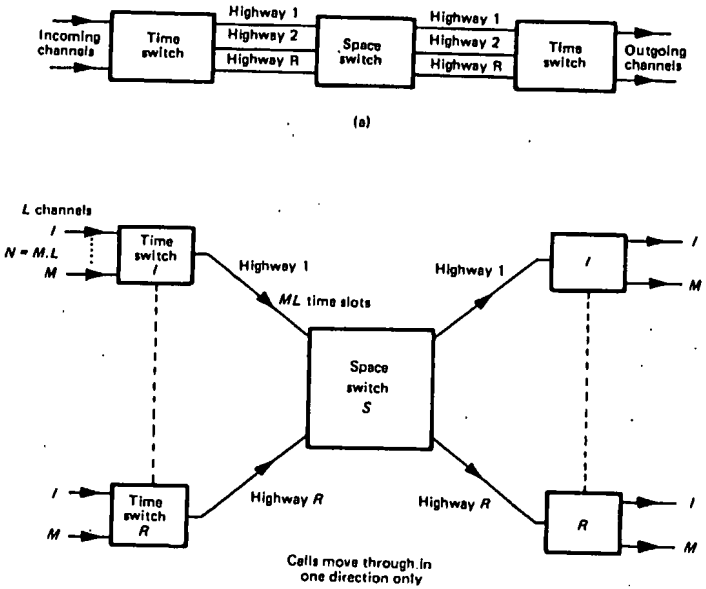


Fig. 10.39 Digital switch: (a) TST block diagram; (b) network representation.

the technology employed, but the delay may be from 1 to 31 time slots depending on the relative positions of the time slots in and out of the unit, and the chosen free time slot in the space switch.

To understand the behaviour of the TST switch in terms of the link systems considered earlier it is important to appreciate that for each time slot the interconnections in the space switch will be different; at each time slot there will be a different set of calls in progress and the connections between the highways will only last for one time slot period then new connections will be established. This can be represented by having N space switches (Fig. 10.40), one for each time slot.

Whether or not blocking occurs in the TST unit depends entirely on the dimensions of the space switch, and since in modern systems switches are comparatively inexpensive they are usually large enough to make blocking negligible. For total non-blocking there must be at least as many outlets as inlets on the time switches, and the space switch highways must have $2N - 1$ time slots, where N is the number of time slots in a link to a time switch.

Digital switches are uni-directional, and that implies that two paths are required to connect two channels X and Y, one for conversation from X to Y and the other

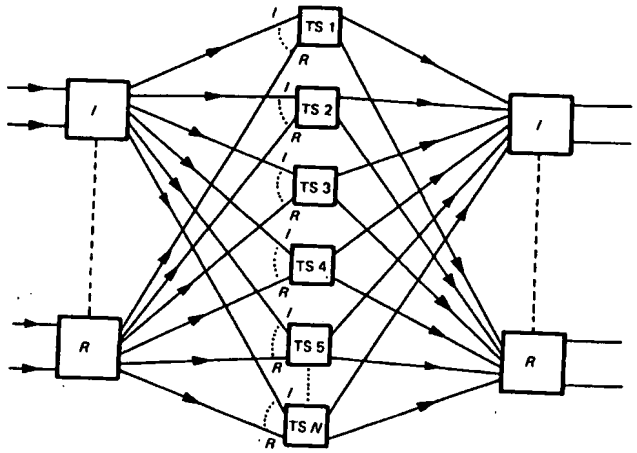


Fig. 10.40 Analogue equivalent of TST switch.

for conversation from Y to X. To reduce the control process, the X to Y slot is chosen according to whatever rules are used by the designer, and the Y to X interconnection is allocated a fixed number of time-slots from it, e.g. one, or half a frame. By this method, if the X to Y connection is available, the Y to X must also be free.

The digital switch just described is situated in a main exchange, forming part of the trunk network. Interconnections between exchanges are made, for the information paths, via PCM links. However, signalling is carried over a common channel, using signalling system CCITT No. 7 as described in Section 10.3. The inter-exchange signals will not only be concerned with setting up calls between exchanges, but with accounting, administration fault diagnosis and maintenance. As described earlier, the CCITT No. 7 system transmits signals as messages that can have variable length. Each message is preceded by labels that identify its origin and destination exchange, the type of message (call handling, fault, etc.) and includes error-detection and acknowledgement bits. If an error is detected in a message, that and all subsequent messages are retransmitted to ensure that the sequence received at the far end is in the correct order.

The error rate has an important bearing on the capacity of the signalling channel; re-transmission of incorrectly received messages obviously takes up time that could be used for new signals and consequently slows down the overall process. The capacity is specified in terms of number of messages per busy hour given that the delay from end to end is not greater than some predetermined value.

On the subscriber side of the digital switch there will be a local unit of some description. For large areas a local digital exchange would be used with mf signalling from subscriber to exchange where conversion to a PCM format would

take place before concentration through a digital switch. Alternatively for very small units no exchange facility would be available, but a simple digital concentrator would be used to take in the analogue channels, convert them to PCM and multiplex them onto a single highway to the nearest local exchange. Calls between subscribers on the same concentrator would then have to pass through the local exchange. Signalling in these PCM links would be on TS16 and the control, software and firmware at the main exchange would convert it to common channel if a trunk call was required.

The introduction of digital systems is rarely a starting point for the telephone system. Usually a system exists and the digital equipment has to be grafted on to it. Whatever method is used, interworking between the old and new systems is required, and one area of difficulty is the interfacing of various analogue signalling systems with the new equipment designed to operate on TS16 or common channel. This interfacing can be a severe problem if there are many existing signalling schemes in a particular network, and the development of satisfactory units can add considerably to system costs.

10.35 Conclusion

In some respects, the whole concept of telecommunications is changing, and with it the services and functions provided by telecommunications operating companies. The very rapid growth of information technology, with its demand for high-speed, large-capacity, data links, with video output, and the introduction of new facilities on telephone exchanges, are both causing manufacturers to rethink the whole philosophy of how communication systems should best be provided. In the next few years the main point at issue will be to decide whether future systems should be fully integrated, or not. If so, should they be based on a digital telephone system, with its centralized switching units, or on one of the many data networks, with distributed control? Conversely, economy and efficiency may dictate an extension of the present hybrid scheme in which both methods co-exist, with perhaps an element of inter-working between them. It is too early in the development of distributed systems to predict what changes will take place in the next decade, but that major redesign will occur is beyond doubt.

References

1. Brockmeyer, B., Halstron, H.L. and Jensen, A., *The Life and Works of A.K. Erlang*, Copenhagen Telephone Company, 1943.
2. Jacobaeus, C., "A study on congestion in link systems", *Ericsson Technics*, No. 48, 1950.
3. Cooper, R.B., *Introduction to Queueing Theory*, Edward Arnold, London, 1981, Chapter 5.
4. Fry, T.C., *Probability and its Engineering Uses*, Van Nostrand Reinhold, Wokingham, 1965.

print version

A3

Wired News

Search:

Wired News Search

[an error occurred while processing this directive][an error occurred while processing this directive][an error occurred while processing this directive]

Issue 2.09 - Sep 1994

Superdistribution

By Brad Cox

It has become a maxim: intangible electronic goods - software applications, online magazine stories, clip art - are quite distinct from tangible goods like baskets, potatoes, and oil refinery machinery. Tangible goods are made of atoms; electronic goods are made of bits. While those who produce electronic goods must expend the same capital, labor, and knowledge as those producing tangible goods, their products can be copied in nanoseconds and transported at the speed of light.

The hard-to-copy nature of tangible goods made the traditional pay-per-copy mechanism a natural choice. But an info product's ease of duplication so thoroughly undercuts the traditional notion of pay-per-copy that the possibility of an abundant supply of pre-fabricated information-age goods is nearly nixed.

But imagine a significantly altered market mechanism for electronic goods. Instead of treating ease-of-replication as a liability to be prevented - via labor-intensive copy protection and legal or moral restrictions - this new model treats ease-of-replication as the asset upon which a new foundation for software engineering could be based. In Japan this new way is called superdistribution. Superdistribution lets information flow freely, without resistance. Eschewing the low-tech property-rights mechanisms already widespread (shrinkwrap software, license servers, dongles, demoware, shareware), superdistribution allows miners, refiners, fabricators, assemblers, distributors, and marketers to cooperate and compete as producers and consumers of electronic goods within a global information-age society.

Existing copyright law distinguishes between copyright (the right to copy or distribute) and useright (the right to "perform," or to use a copy once obtained). In the eyes of the law, when Joe Sixpack buys a record or CD at a store, he's actually purchasing a *bundle* of rights that includes ownership of a physical medium along with a limited useright that allows use of the music on that medium only for personal enjoyment. Large television and radio companies buy an entirely different bundle of rights. They often have the same media (whose only difference is a "not for resale" sticker on the cover) thrust upon them for free by publishing companies in expectation of substantial fees for the useright to play the music on the air. These fees are administered by ASCAP (American Society of Composers, Authors and Publishers) and BMI (Broadcast Musicians Institute), who monitor how often each record is broadcast to how large a listening audience.

Similarly, superdistribution treats each personal computer as a broadcasting station whose "audience"

consists of a single "listener." First pioneered in 1987 by Ryoichi Mori, head of the Japan Electronics Industry Development Association, superdistribution is based on the observation that electronic objects are fundamentally unable to monitor their own copying but trivially able to monitor their use. For example, making software - whether it's Microsoft's Word or Mike's string-compare subroutine - count how many times it has been invoked is easy, but making it count how many times it has been copied is much more difficult. So why not build an information-age market economy around this difference? If revenue collection were based on monitoring the use of software inside a computer, vendors could dispense with copy protection altogether. They could distribute electronic objects for free in expectation of a usage-based revenue stream. (This, of course, raises the same hairy privacy issues that we trade off when we choose to use credit cards instead of cash or talk by telephone rather than face to face. The real risk to privacy here does not arise when usage information is used only for billing, but from any possibility that it might be used for other purposes.)

Treating ease-of-replication as an *asset* rather than a *liability*, superdistribution actively encourages free distribution of information-age goods via any distribution mechanism imaginable. It invites users to download superdistribution software from networks, to give it away to their friends, or to send it as junk mail to people they've never met.

Why this generosity? Because the software is actually "meterware." It has strings attached, whose effect is to decouple revenue collection from the way the software was distributed. Superdistribution software contains embedded instructions that make it useless except on machines that are equipped for this new kind of revenue collection.

Superdistribution-equipped computers are otherwise quite ordinary. They run ordinary pay-by-copy software just fine, but they have additional capabilities that only superdistribution software uses. In Mori's prototype, these extra services are provided by a silicon chip that plugs into a Macintosh coprocessor slot. The hardware is surprisingly uncomplicated (its main complexities being tamper-proofing, not base functionality), and far less complicated than hardware that the computer industry has been routinely building for decades. Electronic objects intended for superdistribution invoke this hardware, which provides instructions. These instructions check that revenue-collection hardware is present, prior usage reports have been uploaded, and prior usage fees have been paid. They also keep track of how many times they have been invoked, storing the resulting usage information temporarily in a tamper-proof persistent RAM. Periodically (say monthly), this usage information is uploaded to an administrative organization for billing, using encryption technology to discourage tampering and to protect the secrecy of the metered information. (Think of your utility bill.)

Software users receive monthly bills for use of each top-level component - say, Microsoft Excel, *Myst*, or a Net-based rock video. When these bills are paid, payments are divvied up between the makers of the component and makers of subcomponents - in proportion to usage. For example, for the rock video, payment might go to MTV as well as to the artist. In other words, the end-user's payments are recursively distributed through the producer-consumer hierarchy. The distribution is governed by usage metering information collected from each end-user's machine, plus usage pricing data provided to the administrative organization by each component vendor. (The various rounds of payment resemble those made by Visa or MasterCard.)

Since communication is infrequent and involves only a small amount of metering information, the communication channel could be as simple as a modem that autodials a hardwired 800 number each month. Many other solutions are viable, such as flash cards or even floppy disks to be mailed back and forth each month.

Consider an author who wishes to distribute or sell a multimedia document that cannot be handled as a simple text file. Without superdistribution, the author's market is confined to those who have already purchased a program capable of displaying this document - a run-time version of Macromedia Director, for example. The same occurs at each lower level of the producer/consumer hierarchy. The market of a programmer who wishes to sell a reusable software component is restricted to those who have already purchased the components and tools upon which the software component relies.

With superdistribution, the market is no longer restricted to those who own Director, because it will be acquired by the customers' operating system as if it were a part of the document. The creator of the document accrues revenue from those who read it, as does the creator of Director.

The user's operating system acquires subcomponents of the document, such as Director and any subcomponents it relies on (QuickDraw, etc.), from the hard drive's cache, automatically loading it as needed from the network. The operating system can do this automatically and transparently because loading software involves no financial commitments when revenue is based on usage instead of acquisition of copies.

Superdistribution addresses the perennial, implicit questions of those who might potentially provide the smaller-granularity reusable software components upon which an advanced software engineering culture could be founded: Where do software components come from? Why should I bother to provide them? Why should I engage in such gritty activities as testing and documenting reusable software components sufficiently so that others might use them? What is in it for me?

Where software's ease-of-replication is a liability today (by disincentivizing those who would provide it), superdistribution turns this liability into an asset by allowing goods to be distributed for free. Where software vendors must now spend heavily to overcome software's invisibility, superdistribution would thrust software out into the world to serve as its own advertisement. Where the personal computer revolution isolates individuals inside a stand-alone personal computer, superdistribution establishes a cooperative/competitive community around an information-age market economy.

By separating revenue collection from acquisition of copies, hard drives and computers can disappear and become just part of the plumbing that conveys information-age goods between producers and consumers. Computers and telecommunications links become invisible, a transparent window through which individuals can communicate, cooperate, coordinate, and compete as members of an advanced socioeconomic community.

Brad Cox (bcox@gmu.edu) is founder of the Coalition for Electronic Markets and a faculty member in George Mason University's Program on Social and Organizational Learning.

Atomic Proxy Cryptography

Matt Blaze Martin Strauss
AT&T Labs – Research
Florham Park, NJ 07932
{mab,mstrauss}@research.att.com

DRAFT – 2 November 1997 – DO NOT RE-DISTRIBUTE*

Abstract

This paper introduces *atomic proxy cryptography*, in which an *atomic proxy function*, in conjunction with a public *proxy key*, converts ciphertext (messages in a public key encryption scheme or signatures in a digital signature scheme) for one key (k_1) into ciphertext for another (k_2). Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. Various kinds of proxy functions might exist; *symmetric* atomic proxy functions assume that the holder of k_2 unconditionally trusts the holder of k_1, while *asymmetric* proxy functions do not. It is not clear whether proxy functions exist for previous public-key cryptosystems. Several new public-key cryptosystems with symmetric proxy functions are described: an encryption scheme, which is at least as secure as Diffie-Hellman, an identification scheme, which is at least as secure as the discrete log, and a signature scheme derived from the identification scheme via a hash function.

1 Introduction

1.1 Overview

A basic goal of public-key encryption is to allow only the key or keys selected at the time of encryption to decrypt the ciphertext. To change the ciphertext to a different key requires re-encryption of the message with the new key, which implies access to the original cleartext and to a reliable copy of the new encryption key. Intuitively, this seems a fundamental, and quite desirable, property of good cryptography; it should not be possible for an untrusted party to change the key with which a message can be decrypted.

This paper, on the other hand, investigates the possibility of *atomic proxy functions* that convert ciphertext for one key into ciphertext for another without revealing secret decryption keys or cleartext messages. An atomic proxy function allows an untrusted party to convert ciphertext between keys without access to either the original message or to the secret component of the old key or the new key. In proxy cryptography, the holders of public-key pairs A and B create and publish a *proxy key* $\pi_{A\to B}$ such that $D(\Pi(E(m,e_A),\pi_{A\to B}),d_B) = m$, where $E(m,e)$ is the public encryption function of message m under encryption key e, $D(c,d)$ is the decryption function of ciphertext c under decryption key d, $\Pi(c,\pi)$ is the atomic proxy function that converts ciphertext

*Current draft available at ftp://ftp.research.att.com/dist/mab/proxy.ps

c according to proxy key π, and e_A, e_B, d_A, d_B are the public encryption and secret decryption component keys for key pairs A and B, respectively. The proxy key gives the owner of B the ability to decrypt "on behalf of" A; B can act as A's "proxy." In other words, the Π function effectively allows the "atomic" computation of $E(D(c, d_A), e_B)$ without revealing the intermediate result $D(c, d_A)$.

We consider atomic proxy schemes for encryption, identification and signatures. An encryption proxy key $\pi_{A \to B}$ allows B to decrypt messages encrypted for A and an identification or signature proxy key $\pi_{A \to B}$ allows A to identify herself as B or to sign for B (i.e., transforms A's signature into B's signature). Generating encryption proxy key $\pi_{A \to B}$ obviously requires knowledge of at least the secret component of A (otherwise the underlying system is not secure) and similarly generating identification or signature proxy key $\pi_{A \to B}$ requires B's secret, but the proxy key itself, once generated, can be published safely.

1.2 Categories of proxy schemes

Encryption proxy functions (and similarly but contravariantly, identification or signature proxy functions) can be categorized according to the degree of trust they imply between the two key holders. Clearly, A must (unconditionally) trust B, since the encryption proxy function by definition allows B to decrypt on behalf of A. *Symmetric* proxy functions also imply that B trusts A, e.g., because d_B can be feasibly calculated given the proxy key plus d_A. *Asymmetric* proxy functions do not imply this bilateral trust. (Note that this model implies that proxy cryptography probably makes sense only in the context of public-key cryptosystems. Any secret-key cryptosystem with an asymmetric proxy function could be converted into a public-key system by publishing one key along with a proxy key that converts ciphertext for that key into ciphertext for a second key (which is kept secret.))

We can also categorize the asymmetric proxy schemes that might exist according to the convenience in creating the proxy key. In an *active asymmetric* scheme, B has to cooperate to produce the proxy key $\pi_{A \to B}$ feasibly, although the proxy key (even together with A's secret key) might not compromise B's secret key. In a *passive asymmetric* scheme, on the other hand, A's secret key and B's public key suffice to construct the proxy key. Clearly, any passive asymmetric scheme can be used as an active asymmetric scheme, and any asymmetric scheme can be used as a symmetric scheme.

Finally, we can distinguish proxy schemes according to the "metadata" they reveal about the identity of the keys being transformed. *Transparent* proxy keys reveal the original two keys to a third party. *Translucent* proxy keys allow a third party to verify a guess as to which two keys are involved (given their public keys). *Opaque* proxy keys reveal nothing, even to an adversary who correctly guesses the original keys (but who does not know the private keys involved).

1.3 Proxy schemes in theory and practice

The proxy relationship is necessarily transitive. If there are public proxy keys $\pi_{A \to B}$ and $\pi_{B \to C}$, then anyone can compute a proxy function for $A \to C$. Symmetric proxy schemes further establish equivalence classes of keys where the secret component of any key can be used to decrypt messages for any other key in the same class. Note that creating a single symmetric proxy key between a key in one class and a key in another effectively joins the two classes into one.

The notion of proxy cryptography is a rather natural generalization of public-key cryptography and has some pleasing theoretical properties. The proxy schemes we consider below have the additional property that anyone can use the proxy key $\pi_{A\to B}$ to transform the public key of A to the public key of B. For such proxy schemes, as we will see in the various examples below, certain aspects of the security of publishing a proxy key actually follow from the fact that anyone, trusted or not, can use a proxy key to transform ciphertext and keys.

For example, suppose random messages m and m' are encrypted with random secret keys a and b as $E(m,a)$, $E(m',b)$. Suppose that knowing the proxy key $\pi_{A\to B}$ enables Eve, who knows neither a nor b, to recover m or m'. Then, ignoring B altogether and starting with just two (presumably secure) ciphertexts $E(m,a)$ and $E(m',a)$, Eve can pick a random proxy key $r = \pi_{A\to Q}$ for some Q, transform $E(m',a)$ to $E(m',q)$ (where q is the unknown secret key of Q), transform A's public key into Q's public key, and proceed with the hypothesized cryptanalysis. We conclude that if it is safe for A to publish k messages then it is safe for A and B to publish a total of k messages *and* to publish a proxy key, provided only that Eve can successfully *apply* the proxy key to transform ciphertext and public keys.

Because proxy keys are tied to specific key pairs, it is not necessary in many applications to certify or otherwise take special care in distributing them (except to prevent denial-of-service). In particular, it is generally sufficient to rely on the certification and trust established in A (for encryption) or B (for signatures) when using proxy key $\pi_{A\to B}$, since a valid proxy key can by definition only be generated with the cooperation of the owner. Furthermore, the proxy function can be safely applied at any convenient time or place, by the message's sender or receiver, or at any intermediate (and possibly untrusted) point in the network.

Proxy functions potentially also have practical utility for key management in real systems. For example, some pieces of secure hardware (*e.g.*, smartcards) limit the number of secret keys that can be stored in secure memory, while some applications might require the ability to decrypt messages for more keys than the hardware can accommodate. With proxy cryptography, once a new key is created and a corresponding proxy key generated, the secret component of the old (or new) key can be destroyed, with the (public and externally-applied) proxy key maintaining the ability to decrypt for both. In effect, proxy functions allow us to increase the number of public keys without also increasing the number of secret bits or the amount of secret computation. Because proxy functions can be computed anywhere, messaging systems, such as electronic mail, can proxy "forward" messages encrypted with one key to a recipient who holds a different key. Proxy functions make it possible to associate a single key with a network or physical address but still decrypt messages forwarded (and proxied) from other addresses. Finally, proxy functions effectively allow changing or adding a key without obtaining new certificates or altering the distribution channel for the previous public key; this could be useful when it is difficult to distribute or certify new keys (e.g., old keys were published in widely-distributed advertisements or embedded in published software, or the certification authority charges high fees for new certificates).

1.4 Security of proxy schemes and ad hoc substitutes

If Alice wants Bob to be able to read her mail, instead of issuing a proxy key she might just give Bob her secret key (perhaps, obviating the need to involve Bob, by encrypting it in Bob's public key and publishing it). This would be inferior to using a proxy scheme for several reasons. First, as discussed above, Bob's computing environment may be limited and therefore incapable

of automatically processing encrypted secret keys; any new software to decrypt and manage such keys would have to run within the environment trusted by Bob. Proxy processing, on the other hand, can take place entirely outside of Alice's and Bob's trusted environments and without their active involvement. Furthermore, encrypting one's secret key with another's public key is not in general secure. The cryptosystem we present below, a variant[1] of ElGamal, is thought to be secure in part because the cryptanalysis problem is random-self-reducibile—which allows one to assert mathematically that recovering m from the public information $\langle e_a, E(m, e_a), e_b \rangle$ is hard on average if it is hard at worst. The task of recovering m from $\langle e_a, E(m, e_a), E(d_a, e_b), e_b \rangle$, however, may be considerably easier since $E(d_a, e_b)$, in the context of e_a and e_b, may leak information about d_a—specifically, the new cryptanalysis problem is probably not random-self-reducible and due to the problem's obscurity it is not clear what, if any, mathematical guarantees of security can be given. By contrast, the proxy scheme we give below is just as strong as the underlying ElGamal-like cryptosystem.[2]

1.5 Related work

A natural question to ask is whether there exist atomic proxy functions (and feasible schemes to generate proxy keys) for any public key cryptosystems.

Previous work on delegating the power to decrypt has focused on developing efficient transformations that allow the original recipient to forward *specific ciphertexts* to another recipient. Mambo and Okamoto[MO97] develop this formulation and give efficient transforms (more efficient than decryption and re-encryption) for ElGamal and RSA. Mambo, Usuda and Okamoto[MUO96] apply a similar notion to signature schemes.

While such schemes have value from the standpoint of efficiency, they are not, however, "atomic proxy cryptosystems" by our definition because the transforming function must be kept secret and applied online by the original keyholder on a message-by-message basis (the schemes are not atomic). The security semantics of these systems are essentially the same as a decryption operation followed by a re-encryption operation for the new recipient. Our formulation of proxy cryptography is distinguished from the previous literature by the ability of the keyholder to publish the proxy function and have it applied by untrusted parties without further involvement by the original keyholder.

2 Proxy encryption

Although the problem of proxy cryptography seems like a natural extension of public-key cryptography, existing cryptosystems do not lend themselves to obvious proxy functions. RSA[RSA78] with a common modulus is an obvious candidate, but that scheme is known to be insecure[Sim83][DeL84]. Similarly, there do not appear to be obvious proxy functions for many of the previous discrete-log-

[1] David Wagner notes that our proxy scheme can be extended to work with standard ElGamal[ElG85] encryption.

[2] Note that Bob of this example may be a government mandating that Alice provide him with access to her key. It has been argued that such a scheme makes the system as a whole less trustworthy due to the extra engineering effort involved; we argue here that in the case of random-self-reducible cryptosystems such as ElGamal variants, requiring Alice to encrypt her secret key using the government's public key may also weaken the underlying cryptosystem in the precise mathematical sense of spoiling the random-self-reducibility.

based cryptosystems. This is not to say, of course, that proxy functions for existing systems do not exist, only that we have not discoverd them.

We now describe a new secure discrete-log-based public-key cryptosystem that does have a simple proxy function. The scheme is similar in structure to ElGamal encryption[ElG85], but with the parameters used differently and the inverse of the secret used to recover the message. (This approach has merit beyond proxy encryption; [Hug94] proposed a Diffie-Hellman-like key agreement protocol based on the inverse of the secret, which allows a message's sender to determine the key prior to identifying its recipient).

2.1 Cryptosystem $\mathcal{X}$ (encryption)

Let p be a prime of the form $2q+1$ for a prime q and let g be a generator in Z_p^*; p and g are global parameters shared by all users. A's secret key a, $0 < a < p-1$, is selected at random and must be in Z_{2q}^*, *i.e.*, relatively prime to $p-1$. (A also calculates the inverse $a^{-1} \bmod 2q$). A publishes the public key $g^a \bmod p$. Message encryption requires a unique randomly-selected secret parameter $k \in Z_{2q}^*$. To encrypt m with A's key, the sender computes and sends two ciphertext values (c_1, c_2):

$$\begin{aligned} c_1 &= mg^k \bmod p \\ c_2 &= (g^a)^k \bmod p \end{aligned}$$

Decryption reverses the process; since

$$c_2^{(a^{-1})} = g^k \pmod p$$

it is easy for A (who knows a^{-1}) to calculate g^k and recover m:

$$m = c_1((c_2^{(a^{-1})})^{-1}) \bmod p$$

The speed of the scheme is comparable to standard ElGamal encryption, although initial key generation requires the additional calculation and storage of a^{-1}.

2.2 Symmetric proxy function for $\mathcal{X}$

Observe that the c_1 ciphertext component produced by Cryptosystem $\mathcal{X}$ is independent of the recipient's public key. Recipient A's key is embedded only in the c_2 exponent; it is sufficient for a proxy function to convert ciphertext for A into ciphertext for B to remove A's key a from c_2 and replace it with B's key b. Part of what a proxy function must do, then, is similar to the first step of the decryption function, raising c_2 to a^{-1} to remove a. The proxy function must also contribute a factor of b to the exponent. Clearly, simply rasing c_2 to a^{-1} and then to b would accomplish this, but obviously such a scheme would not qualify as a secure proxy function; anyone who examines the proxy key learns the secret keys for both A and B.

This problem is avoided, of course, by combining the two steps into one. Hence, the proxy key $\pi_{A\to B}$ is $a^{-1}b$ and the proxy function is simply $c_2^{\pi_{A\to B}}$. Note that this is a symmetric proxy function; A and B must trust one another bilaterally. B can learn A's secret (by multiplying the proxy key by b^{-1}), and A can similarly discover B's key. This proxy function is also translucent; the proxy key does not directly reveal A or B, but anyone can verify a guess by encrypting a message with A's public key, applying the proxy function, and comparing the result with the encryption of the same message (with the same k) with B's public key. Observe that applying the proxy function is more efficient than decryption and re-encryption, in that only one exponentiation is required.

2.3 Security of $\mathcal{X}$

First, we show that $\mathcal{X}$ is secure—that cleartext and secret keys cannot be recovered from ciphertext and public keys. Beyond that, we also show that publishing the proxy key compromises neither messages nor secret keys. Since recovering a secret key enables an adversary to recover a message and since cryptanalysis is easier with more information (i.e., a proxy key), it is sufficient to show that no cleartext is recoverable from ciphertext, public keys, and proxy keys. Specifically, we will show that the problem of recovering m from

$$(g^a, g^b, g^c, \ldots, mg^k, g^{ak}, a^{-1}b, a^{-1}c, \ldots).$$

is at least as hard as Diffie-Hellman.

Theorem 1 *Suppose there exists a randomized algorithm f that with probability $\epsilon > 1/|p|^{O(1)}$ succeeds in recovering m from the public information*

$$(g^a, g^b, \ldots, mg^k, g^{ak}, b/a, \ldots)$$

where the probability is taken over f's random choices as well as over m and the parameters a, b, and k. Then, for each $\eta = 2^{-|p|^{O(1)}}$, there exists a randomized polynomial-time algorithm for Diffie-Hellman that succeeds with probability $1 - \eta$.

Proof. For simplicity we assume there are only two public keys and one proxy key; the general case is similar. Suppose we had an algorithm F that always succeeds in recovering m from $(g^a, g^b, mg^k, g^{ak}, b/a)$. Then note that on input g^x, g^y, we have $F_1(g^x, g^y) = F(g^y, g^y, 1, g^x, 1)^{-1} = g^{x/y}$, and since $F_1(g, g^y) = g^{1/y}$ we'd have

$$F_2(g^x, g^y) = F_1(g^x, F_1(g, g^y)) = F_1(g^x, g^{1/y}) = g^{xy}.$$

In fact, f is only guaranteed to recover m with probability ϵ so we need to use the random-self-reducibility [Fei93] of the discrete log to achieve our objective.

On input g^x, g^y for $x, y \in Z_{2q}^*$ let f_1 pick $r, s, t, u \in Z_{2q}^*$ at random and query

$$f(g^{ry}, g^{sy}, g^t, g^{ux}, s/r).$$

By hypothesis, with probability ϵ we get $g^{t-(ux)/(ry)}$ from which f_1 can recover $g^{x/y}$. Since $f_1(g, g^y) = g^{1/y}$ we can define f_2 by

$$f_2(g^x, g^y) = f_1(g^x, f_1(g, g^y));$$

this is equal to

$$f_1(g^x, g^{1/y}) = g^{xy}$$

with probability at least ϵ^2.

Our next step is to construct an algorithm that runs correctly with high probability on most inputs in Z_{2q}^*. For this, we define the algorithm $f_3(g^x, g^y)$, $x, y \in Z_{2q}^*$, as follows. Pick r, s, t, u at random with $r, s, u \in Z_{2q}^*$ and $0 \le t < 2q$ even. Compute $f_2(g^{rx}, g^{sy}) = g^{rsxy+c}$ and $f_2(g^{(t+x)/u}, g^{uy}) = g^{ty+xy+c'}$ for some c, c' that depend on the respective input to f_2. Check whether $(g^{rsxy+c})^{1/rs} =$

$(g^{ty+xy+c'})/g^{ty}$ and is of the form g^z for $z \in Z_{2q}^*$ and if so output the common value, otherwise abort.

We need to show that the probability that f_3 outputs the correct answer is substantial and the probability that it outputs an incorrect answer is negligible. Note that the inputs to f_2 are random, so, by hypothesis, at least ϵ^4 of the time $c = c' = 0$ and therefore f_3 will answer correctly. For f_3 to answer incorrectly, we must have $c, c' \neq 0$ and $c/rs = c'$. Note also that in this case c and c' must be even so that $xy + c' = xy + c/(rs)$ are in Z_{2q}^*. Even conditioned on the four inputs to two calls to f_2, the six random variables r, s, t, u, x, y have two degrees of freedom left, and it is easy to see that r and s and therefore rs remain uniformly distributed. Thus, c/rs is uniformly distributed among even numbers modulo $2q$ and only equals c' with probability $1/q$. Thus if we repeat f_3 a total of $O(-\log \eta)/\epsilon^4$ times we will have a probability $1 - \eta$ of a correct answer and only a tiny chance of getting any incorrect answer.

Next, we show how to construct an algorithm $f_4(g^x, g^y)$ that succeeds, with high probability, on all inputs g^x, g^y such that $x, y \in Z_{2q}^*$. Pick r, s at random from Z_{2q}^* and compute $f_3(g^{rx}, g^{sy})^{1/rs}$. The input to f_3 is uniformly distributed, so by hypothesis $f_3(g^{rx}, g^{sy}) = g^{rsxy}$ with high probability and we recover g^{xy}.

Before considering general g^x, g^y we recall some facts about arithmetic modulo $2q$. The integers modulo $2q$ consist of 0, q, $(q-1)$ multiples of 2 (other than 0), and $(q-1)$ invertible elements (the odd numbers other than q). Given an input g^x where g is a primitive element modulo $p = 2q + 1$, one regards x modulo $2q$. We can learn whether x is invertible from g^x: If $x = 0$ then $g^x = 1$, if $x = q$ then $g^x = -1$, if x is odd then $(g^x)^q = g^q = -1$ and if x is even then $(g^x)^q = g^0 = 1$. (Raising g^x to the power q is polynomial-time, but expensive. However, we do not need to do this when using the cryptosystem.)

Finally, consider general input g^x, g^y. The cases $x = 0, x = q$ or $y = 0, y = q$ are easy to detect and handle, so we assume that we are not in one of these cases. We can determine s and t in $\{0, 1\}$ so that $x + sq, y + tq \in Z_{2q}^*$. We have $g^{xy} = g^{(x+sq)(y+tq)}/g^{xtq+ysq+stq^2} = \pm g^{(x+sq)(y+tq)} = \pm f_4(g^{x+sq}, g^{y+tq})$ (and we can determine the sign). □

Similarly one can show that recovering a from $(g^a, g^b, mg^k, g^{ak}, b/a)$ is as hard as the discrete log, so publishing the proxy key does not compromise a—not even to the level of Diffie-Hellman.

3 Proxy identification

In this section we describe a discrete-log-based identification scheme. With p, g, a as before, Alice wishes to convince Charlotte that she controls a; Charlotte will verify using public key g^a. As before, the proxy key $\pi_{A \to B}$ will be a/b—it will be safe to publish a/b and Alice and Charlotte can easily use a/b to transform the protocol so Charlotte is convinced that Alice controls b.

Note that in the case of a secure identification proxy key that transforms identification by A into identification by B, it is B whose secret is required to construct the proxy key because identification as B should not be possible without B's cooperation.

3.1 Cryptosystem $\mathcal{Y}$ (identification)

Let p and g be a prime and a generator in Z_p^*, respectively. Alice picks random $a \in Z_{2q}^*$ to be her secret key and publishes g^a as her public key. Each round of the identification protocol is as

follows:

- Alice picks a random $k \in Z_{2q}^*$ and sends Charlotte $s_1 = g^k$.
- Charlotte picks a random bit and sends it to Alice.
- Depending on the bit received, Alice sends Charlotte either $s_2 = k$ or $s_2' = k/a$.
- Depending on the bit, Charlotte checks that $(g^a)^{s_2'} = s_1$ or that $g^{s_2} = g^k$.

This round is repeated as desired. As with existing protocols, there may be ways to perform several rounds in parallel for efficiency [FFS88].

3.2 Symmetric proxy function for $\mathcal{Y}$

A symmetric proxy key is simply a/b. Proxy identification is useful as follows: Suppose Charlotte wants to run the protocol with g^b instead of g^a. Either Alice or Charlotte or any intermediary can use the proxy key to convert Alice's responses k/a to k/b. Also, either party can transform its share of the key pair (a, g^a) to b or g^b before any protocol takes place. Thus Alice and Bob can authenticate for each other but otherwise the protocol is secure. This proxy scheme is translucent.

3.3 Security of $\mathcal{Y}$

Theorem 2 *Protocol $\mathcal{Y}$, with or without proxy keys published, is a zero-knowledge protocol that convinces the verifier that the prover knows the secret key.*

Proof. Without proxy keys published, this protocol is similar to others in the literature (see, e.g., [FFS88]). Note that if a prover could produce both k/a and k then the prover could produce a from g^a (perhaps only with significant probability).

Now suppose that a proxy key a/b is published for random public keys g^a and g^b, and suppose that D can then impersonate A. Since D could already generate a random proxy key r and matching public key g^{ar}, it follows that D could impersonate A even without knowing a/b and g^b. Thus publishing proxy keys does not weaken the system. □

4 Proxy signature

The concept of proxy cryptography also extends to digital signature schemes. A signature proxy function transforms a message signature so that it will verify with a public key other than that of the original signer. In other words, a signature proxy function $\Pi(s, \pi_{A\to B})$ with proxy key $\pi_{A\to B}$ transforms signature s signed by the secret component of key A such that $V(m, \Pi(S(m, A), \pi_{A\to B}), B) =$ VALID, where $S(m, k)$ is the signature function for message m by key k and $V(m, s, k)$ is the verify function for message m with signature s by key k.

Again, existing digital signature schemes such as RSA[RSA78], DSA[NIS91], ElGamal[ElG85], etc. do not have obvious proxy functions (which, again, is not to say that such functions do not exist).

As in the case of proxy identification, in order to construct a proxy key that transforms A's signature into B's signature, B's secret must be required to construct the proxy key because signing for B should not be possible without B's cooperation.

Now we will see how to use the proxy identification scheme to construct a proxy signature scheme. We suppose there exists a hash function h whose exact security requirements will be discussed below. The parameters p, g, a, b are as before.

4.1 Cryptosystem $\mathcal{Z}$ (signature)

To sign a message m, Alice picks $k_1, k_2, \ldots k_\ell$ at random and computes $g^{k_1}, \ldots g^{k_\ell}$. Next Alice computes $h(g^{k_1}, \ldots g^{k_\ell})$ and extracts ℓ pseudorandom bits $\beta_1, \ldots, \beta_\ell$. For each i, depending on the i'th pseudorandom bit, Alice (who knows a) computes $s_{2,i} = (k_i - m\beta_i)/a$; that is, $s_{2,i} = (k_i - m)/a$ or $s_{2,i} = k_i/a$. The signature consists of two components:

$$\begin{aligned} s_1 &= (g^{k_1}, \ldots, g^{k_\ell}) \\ s_2 &= ((k_1 - m\beta_1)/a, \ldots, (k_\ell - m\beta_\ell)/a) \end{aligned}$$

To verify the signature, first the β_i's are recovered using the hash function. The signature is then verified one "round" at a time, where the i'th round is $(g^{k_i}, (k_i - m\beta_i)/a)$. To verify $(g^k, (k - m\beta)/a)$ using public key g^a, the recipient Charlotte raises (g^a) to the power $(k - m\beta)/a$ and checks that it matches $g^k/g^{m\beta}$.

4.2 Symmetric proxy function for $\mathcal{Z}$

A symmetric proxy key $\pi_{A\to B}$ for this signature scheme is a/b. The proxy function Π leaves s_1 alone and maps each component $s_{2,i}$ to $s_{2,i}\pi_{A\to B}$. The proxy scheme is translucent.

4.3 Security of $\mathcal{Z}$

This scheme relies on the existence of a "hash" function h. Specifically,

Assumption 1 *We assume there exists a function h such that:*

- *On random input (g^a, m), it is difficult to generate $\{r_i\}$ and $\{\beta_i\}$ such that*

$$h(g^{ar_1 + m\beta_1}, \ldots, g^{ar_\ell + m\beta_\ell}) = (\beta_1 \ldots, \beta_\ell).$$

- *More generally, it is difficult to generate such $\{r_i\}$ and $\{\beta_i\}$ on input g^a, m, and samples of signatures on random messages signed with a.*

It is not our intention to conjecture about the existence of such functions h. In particular, we do not know the relationship between Assumption 1 and assumptions about collision freedom or hardness to invert.[3] We note that this generic transformation of a protocol to a signature scheme has appeared in the literature [FS86].

[3] Assumption 1 *does* imply that, on random input g^a, it is hard to find (r_i) making all the β_i's zero, i.e., such that $h(g^{ar_1}, \ldots, g^{ar_\ell}) = 0$.

We now analyze Assumption 1. Note that in order to produce a legitimate signature on m that verifies with g^a, a signer needs to produce $\langle g^{k_i} \rangle$ and $\langle (k_i - m\beta_i)/a \rangle$. Thus, putting $\langle \beta_i \rangle = h(\langle g^{k_i} \rangle)$ and then $\langle r_i \rangle = \langle (k_i - m\beta_i)/a \rangle$, it is straightforward to see that the signer could actually produce r_i's and β_i's of the stated type in the course of producing the signature.

While we do not address the security of h, we can state that issuing proxy keys does not weaken the system.

Theorem 3 *Suppose h satisfies Assumption 1. Then, for most b, it is also hard to produce $\{r_i\}$ and $\{\beta_i\}$ given additional input $a/b, g^b$, and samples of messages signed with b.*

Proof. As above, a signer not having access to b's messages and proxy keys could simulate this by choosing a random proxy key r, generating $g^b = g^{a^r}$, and convert some messages signed with a into messages signed with b. □

5 Conclusions

Intuitively, atomic proxy cryptography is a fairly natural extension of the basic notion of public-key cryptography. It surely seems plausible, given that there exist cryptosystems that can grant the ability to encrypt without granting the ability to decrypt, that there might also exist cryptosystems that can grant the ability to *re-encrypt* without granting the ability to decrypt. However, it is not at all obvious whether there exist atomic proxy schemes in general.

Indeed, while this paper has demonstrated that there do exist efficient and secure public-key encryption and signature schemes with symmetric atomic proxy functions, this observation probably raises more new questions than it answers. In particular, do proxy functions exist for public-key cryptosystems based on problems other than discrete-log? (One possibility is that, for some cryptosystems, proxy functions do exist but it is infeasible to find a proxy key.) More importantly, we have yet to discover a secure *asymmetric* proxy function of any kind; asymmetric proxy functions are probably much more useful in practice, since there are likely many situations where trust is only unidirectional. Are there cryptosystems for which asymmetric proxy functions exist?

6 Acknowledgements

The authors thank Steve Bellovin, Jack Lacy, Dave Maher, Andrew Odlyzko and David Wagner for helpful discussions and comments on earlier drafts.

References

[DeL84] J. M. DeLaurentis. A further weakness in the common modulus protocol for the RSA cryptoalgorithm. *Cryptologia*, 8:235–239, 1984.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IT-31(4):469–472, July 1985.

[Fei93] Joan Feigenbaum. Locally random reductions in interactive complexity theory. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:73–98, 1993.

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Crypto 86*, number 263 in LNCS, pages 186–194, Santa Barbara, CA, USA, August 1986.

[Hug94] Eric Hughes. An encrypted key transmission protocol. *CRYPTO '94* Rump Session presentation, August 1994.

[MO97] M. Mambo and E. Okamoto. Proxy cryptosystems: delegation of the power to decrypt ciphertexts. *IEICE Trans. Fundamentals*, E80-A(1), 1997.

[MUO96] M. Mambo, K. Usuda, and E. Okamoto. Proxy signatures: delegation of the power to sign messages. *IEICE Trans. Fundamentals*, E79-A(9), 1996.

[NIS91] NIST. A proposed federal information processing standard for digital signature standard (DSS). Draft Tech. Rep. FIPS PUB XXX, August 1991.

[RSA78] Ron L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[Sim83] G. J. Simmons. A "weak" privacy protocol using the RSA crypto algorithm. *Cryptologia*, 7:180–182, 1983.

Atomic Proxy Cryptography

Matt Blaze

AT&T Shannon Laboratory

This talk introduces *atomic proxy cryptography*, in which an atomic proxy function, in conjunction with a public proxy key, converts ciphertext (messages in a public key encryption scheme or signatures in a digital signature scheme) for one key (k_1) into ciphertext for another (k_2). Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. Various kinds of proxy functions might exist; {\em symmetric} atomic proxy functions assume that the holder of k_2 unconditionally trusts the holder of k_1, while {\em asymmetric} proxy functions do not. It is not clear whether proxy functions exist for previous public-key cryptosystems. Several new public-key cryptosystems with symmetric proxy functions are described: an encryption scheme, which is at least as secure as Diffie-Hellman, an identification scheme, which is at least as secure as the discrete log, and a signature scheme derived from the identification scheme via a hash function.

Full paper available.

This is joint work with Martin Strauss.

Matt Blaze, Atomic Proxy Cryptography

Gates 498, 10/20/98, 4:15 PM

Matt Blaze's Technical Papers

Last updated 6 August 2006

Many of my technical papers are available here. Newer papers are usually in Adobe PDF format; like it or not, PDF is the de facto standard format for scientific papers these days. Most of the older papers are in PostScript format; you'll need a PostScript printer or viewer (such as GhostView) to read them. Most of these files have also been converted to Adobe PDF format (using ps2pdf) and can be viewed or printed with a PDF viewer such as Acrobat, acroread4, or xpdf. If you have a choice, you'll probably find the PostScript version looks and works better than the PDF version does (ps2pdf doesn't do particularly well with some of the fonts). A few papers are available as plain ASCII text or LaTeX source.

Wiretapping, Surveillance and Countermeasures

The Trustworthy Network Eavesdropping and Countermeasures (TNEC) project studies the reliability of communications interception systems and technologies. A better understanding of the limitations of eavesdropping techniques could lead to more trustworthy law enforcement wiretap evidence (or at least more appropriate treatment of electronic evidence), networks with properties that inherently frustrate (or facilitate) interception, and new techniques for achieving communications security.

One of our first efforts is a comprehensive analysis of the wiretapping technologies used by law enforcement (for both voice and data). We have found serious exploitable weaknesses in fielded interception systems. For details, including audio demos of novel eavesdropping countermeasures, see the wiretapping web page here.

- M. Sherr. E. Cronin, S. Clark and M. Blaze. http://www.crypto.com/papers/wiretapping/ web page.

- M. Sherr, E. Cronin, S. Clark and M. Blaze. "Signaling Vulnerabilities in Wiretapping Systems." *IEEE Security and Privacy.* November/December 2005. [PDF].

Similar vulnerabilities exist in digital Internet eavedropping systems as well:

- E. Cronin, M. Sherr, and M. Blaze. "The Eavedsdropper's Dilemma." Technical Report MS-CIS-05-24. University of Pennsylvania. 2005. [PDF].

Another focus of the TNEC project examines local host-based surveillance. The *JitterBug* demonstrates a novel eavesdropping threat against typed keyboard input. Commercially-available hardware keyboard "sniffers" can easily capture and store an unsuspecting user's keystrokes. Because a subverted keyboard has no direct network connection, sniffer attacks are generally assumed to require either support software on the host or periodic in-person access by the attacker to retrieve the data. We show that this need not be the case. A new technique based on "JitterBugs" can exflitrate captured data entirely through subtle perturbations in the precise times at which typed keystrokes are passed to the host. Whenever a user runs an interactive network application (such as SSH), an attacker can derive previously captured keystrokes entirely by observing the timing of network packets, even from across the

Internet or via encrypted wireless traffic. The JitterBug demonstrates that input devices must be scrutinized as part of any trusted computing base and, more generally, that simple "supply chain attacks" can represent a practical and serious threat to data confidentiality. (Gaurav Shah and Andres Molina won the Best Student Paper award at USENIX Security 2006 for this work.)

- G. Shah, A. Molina, and M. Blaze. "Keyboards and Covert Channels." *Proc. 15th USENIX Security Symposium.* Vancouver, BC. August 2006. [PDF].

- Gaurav Shah's JitterBug page: `http://www.cis.upenn.edu/~gauravsh/jitterbug.html`. JitterBug prototype code and PCB template.

Physical and "Human-Scale" Security

Cryptologic techniques can be applied outside of computers and networks, Perhaps surprisingly, the abstractions used in analyzing secure computing and communications systems turn out also to be useful for understanding mechnical locks and their keyspaces. Indeed, modeling master keyed locks as online authentication oracles leads directly to efficient solutions for what might naively seem like exponential problems for the attacker. In fact, it seems like almost a textbook example, as if master keying practices for locks were designed specifically to illustrate this class of weakness. We sometimes assume that hardware-based security is inherently superior to that based in software, but even the humble mechanical lock can be just as insecure as complex computing systems, and can fail in similar ways.

A widely circulated paper of mine describes attacks against master keyed mechanical locks. For an overview of the attack, which was described in the January 23rd 2003 *New York Times*, click here. For a brief commentary on the reaction to this paper, see my essay, "Keep it secret, stupid!" (click here), which was originally posted to comp.risks.

(Warning: there are embedded photos in this paper; they make the PS and PDF files very large. The GZIPed PostScript version is 5.7MB long (uncompresses to 14MB), and the PDF version is 4MB long.)

- M. Blaze. "Cryptology and Physical Security: Rights Amplification in Master-Keyed Mechanical Locks." March 2003. *IEEE Security and Privacy.* March/April 2003. [GZIPed PostScript], [PDF].

- My *Notes on Picking Pin Tumbler Locks*, intended primarily for use by students in my security seminar, can be found here [HTML].

While the security metrics and mechanical safeguards used in safes and vaults may not relay on the latest technology, they are often quite ingenious. They may have much to teach computer security. Some of what I understand about the subject is in the survey paper below (warning -- heavily illustrated 2.5MB .pdf file). And for a brief commentary on the reaction to *this* paper, see my essay, "the second sincerest form of flattery" (click here), which was originally posted to `interesting-people`.

- M. Blaze. "Safecracking for the Computer Scientist." *U. Penn CIS Department Technical Report.* 7 December 2004 (revised 20 December 2004). [PDF].

This position paper, presented at the Cambridge Security Protocols Workshop 2004, introduces and advocates the "Human Scale Security Project," which supports the above work.

- M. Blaze. "Toward a broader view of security protocols." *12th Cambridge International Workshop on Security Protocols*. Cambridge, UK. April 2004. [PDF].

Trust Management

These papers introduce the "trust management" approach to specifying and enforcing security policy.

- The Trust Management Web Page, updated regularly.

- M. Blaze, J. Ioannidis, A. Keromytis. "Offline Micropayments without Trusted Hardware." *Financial Cryptography 2001*. Grand Cayman, February 2001. [PostScript], [PDF].

- M. Blaze, J. Ioannidis, A. Keromytis. "Trust Management for IPSEC." *NDSS 2001*. San Diego, February 2001. [PDF].

- M. Blaze, J. Feigenbaum, J. Ioannidis, A. Keromytis. The KeyNote Trust Management System, Version 2. *RFC-2704*. IETF, September 1999. [ASCII Text].

- M. Blaze, J. Ioannidis, A. Keromytis. "Compliance Checking and IPSEC Policy Management." *Internet Draft*. `draft-blaze-ipsp-trustmgt-00.txt`. IETF, March 2000. [ASCII Text].

- M. Blaze, J. Ioannidis, A. Keromytis. "DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System." *RFC-2792*. IETF, March 2000. [ASCII Text]

- M. Blaze, J. Ioannidis, and A. Keromytis. " Trust Management and Network-Layer Security Protocols." *1999 Cambridge Protocols Workshop*. Cambridge, April 1999. [PostScript], [PDF], [LaTeX Source].

- M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. "The Role of Trust Management in Distributed Systems Security." Chapter in *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, (Vitek and Jensen, eds.) Springer-Verlag, 1999. [PostScript], [PDF].

- M. Blaze, J. Feigenbaum, M. Strauss. "Compliance-Checking in the PolicyMaker Trust-Management System." *Proc. 2nd Conference on Financial Cryptography*. Anguilla 1998. LNCS 1465, pp 251-265, Springer-Verlag, 1998. [PostScript], [PDF].

- M. Blaze, J. Feigenbaum and J. Lacy. "Decentralized Trust Management." *IEEE Symposium on Security and Privacy*, Oakland, CA. May 1996. [PostScript], [PDF].

Angelos Keromytis's KeyNote Trust Management toolkit and open-source reference

implementation is available here as a GZIPed TAR archive. The toolkit runs under most Unix-like (BSD, linux, etc.) platforms, with limited support for Win32 platforms.

Also see Angelos Keromytis' KeyNote web page for the latest details on the KeyNote implementation.

Remotely-Keyed Encryption

These papers introduce and formalize the notion of "remotely-keyed" encryption, in which a low-bandwidth, but trusted device (such as a smart card) assists a high-bandwidth, but untrusted host with bulk encryption.

- M. Blaze, J. Feigenbaum, and M. Naor. "A Formal Treatment of Remotely Keyed Encryption (Extended Abstract)". Eurocrypt '98, Helsinki. LNCS 1403 pp. 251-265. [PostScript], [PDF].

- M. Blaze. "High-Bandwidth Encryption with Low-Bandwidth Smartcards." January 18, 1996. *Cambridge Workshop on Fast Software Encryption*, February 1996. [PostScript], [PDF].

Key Escrow

These papers describe and evaluate various key escrow proposals, from a technical (as opposed to political) perspective.

- *The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption* (second edition). June 1998. [HTML], [PDF].

- *The Risks of Key Recovery, Key Escrow, and Trusted Third-Party Encryption* (first edition). May 1997. (OBSOLETE: superseded by second edition, above). [ASCII Text], [PDF], [PostScript].

- M. Blaze. "Oblivious Key Escrow." *First Cambridge Workshop on Information Hiding* May 1996. Springer 1997. [PostScript], [PDF], [LaTeX source].

- Memo from NSA regarding key length report, with comments from M. Blaze and W. Diffie. July 18, 1996. [ASCII Text].

- M. Blaze, W. Diffie, R. Rivest, B. Schneier, T. Shimomura, E. Thompson and M. Wiener. "Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security". Report of ad hoc panel of cryptographers and computer scientists. January 1996. [ASCII Text], [PDF], [PostScript].

- M. Blaze, J. Feigenbaum and F.T. Leighton. "Master-Key Cryptosytems." Abstract presented at *Crypto '95 (rump session)*, Santa Barbara, CA, August 1995. [PostScript], [PDF]

- M. Blaze. "Protocol Failure in the Escrowed Encryption Standard." *Proceedings of Second ACM Conference on Computer and Communications Security*, Fairfax, VA, November 1994. [PostScript], [PDF].

Network-Layer Security

These papers describe the design and implementation network-layer and related security protocols, including JFK, a secure key exchange protocol, and swIPe, a predecessor to the IPSEC standard. (At this point, swIPe is of primarily historical interest, although the USENIX paper should be of some value to IPSEC implementors. JFK is a useful key exchange protocol that should be especially valuable for IPSEC and network security key management).

- W. Aiello, S. M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A. D. Keromytis, and O. Reingold. "Efficient, DoS-Resistant, Secure Key Exchange for Internet Protocols." In Proc. ACM Computer and Communications Security (CCS) Conference. November 2002, Washington, DC. (pp 48-58). [PDF].

- J. Ioannidis and M. Blaze. "The swIPe IP Security Protocol." *Internet Draft*. December 1993. [ASCII Text].

- J. Ioannidis and M. Blaze. "Architecture and Implementation of Network Layer Security Under UNIX." *Proceedings of the Fourth USENIX Security Workshop*, October 1993. [PostScript], [PDF].

Cryptographic Applications

- R. Levein, L. McCarthy, M. Blaze. "Transparent Internet E-mail Security (DRAFT)". August 9, 1996. [PostScript], [PDF].

- M. Blaze and S.M. Bellovin. "Session-Layer Encryption." *Proceedings of the USENIX Security Workshop*, June 1995. [PostScript].

- M. Blaze. "Key Management in an Encrypting File System." *USENIX Summer 1994 Technical Conference,* Boston, MA, June 1994. [PostScript], [PDF].

- M. Blaze. "A Cryptographic File System for Unix." *Proceedings of the First ACM Conference on Computer and Communications Security*, Fairfax, VA, November 1993. [PostScript], [PDF].

  The latest CFS code can be found here.

Ciphers and Algorithms

- S. M. Bellovin, M. Blaze. "Cryptographic Modes of Operation for the Internet." NIST Workshop on AES Modes. Santa Barbara, CA. August 2001. [PDF].

- M. Blaze, M. Strauss. "Atomic Proxy Cryptography." Full version of our *EuroCrypt '98* paper. May 1997. [PostScript], [PDF].

- M. Blaze. "Efficient Symmetric-Key Ciphers Based on an NP-Complete Subproblem (DRAFT)". October 2, 1996. [PostScript], [PDF]

- M. Blaze and B. Schneier. "The MacGuffin Block Cipher Algorithm." *Leuven Workshop on Cryptographic Algorithms*, Leuven, Belgium, December 1994.

[PostScript], [PDF].

Cryptography Policy, Export Regulations, and Politics

- M. Blaze. Declaration in Felten, et al v. RIAA. 13 August 2001. [ASCII Text].
- S. Bellovin, M. Blaze, D. Farber, P. Neumann, E. Spafford. "Comments on the Carnivore System Technical Review." Formal comments to the US Department of Justice. 3 December 2000. [HTML].
- M. Blaze & S. M. Bellovin. "Tapping, Tapping on my Network Door." INSIDE RISKS 124. *CACM*, October 2000. [HTML].
- M. Blaze. "Cryptography Policy and the Information Economy." Draft. 17 December 1996. [PostScript], [PDF], [ASCII Text].
- My prepared testimony before the Senate Commerce Committee subcommittee on Science, Technology, and Space. June 26, 1996 [ASCII Text].
- M. Blaze. "My Life as an International Arms Courier." January, 1995. Adapated from posting to *comp.risks* [ASCII Text]

Peer-to-Peer Networking

My dissertation work, over ten years ago, anticipated and analyzed what we would now call "Peer-to-Peer" file distribution.

- M. Blaze. Caching in Large-Scale Distributed File Systems. PhD thesis. Princeton University Department of Computer Science. November 1992. [PostScript].

Other People's Papers

From time to time, I make available papers from other researchers that I didn't write myself but that are of wide interest and don't otherwise have a home. Here's what's available now:

- S. Fluhrer, I. Mantin and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. Preliminary Draft, July 25, 2001. [PostScript].
- A. Biryukov and A. Shamir. Real Time Cryptanalysis of the Alleged A5/1 on a PC. Preliminary Draft, December 9, 1999. [PostScript].

Click here to return to the crypto.com home page.

Bruce Schneier

Crypto Bibliography

Citations by First Author - B

A. Back, U. Möller, and A. Stiglic, Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems, Proceedings of the 4th Information Hiding Workshop (IHW2001), Springer-Verlag, LNCS v. 2137, pp. 243-254. [.pdf]

S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, A Message Authentication Code based on Latin Squares, Australian Conference on Information Security and Privacy (ACISP '97), Springer-Verlag, LNCS 1270, pp. 194-203, 1997. [.ps.Z]

S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, On Password-Based Authenticated Key Exchange using Collisionful Hash Functions. In Australian Conference on Information Security and Privacy (ACISP '96), Springer-Verlag, LNCS 1172, pp. 299-310, 1996. [.ps.Z]

S. Bakhtiari, R. Safavi-Naini, and J. Piprzyk, On Selectable Collisionful Hash Functions, Australian Conference on Information Security and Privacy (ACISP '96), Springer-Verlag, LNCS 1172, pages 287-298, 1996. [.ps.Z]

T. Baldin, G. Bleumer, and R. Kanne, CryptoManager - Eine intuitive Programmierschnittstelle für kryptographische Systeme; Sicherheitsschnittstellen - Konzepte, Anwendungen und Einsatzbeispiele, Proc. Workshop Security Application Programming Interfaces 94, Deutscher Universitäts Verlag, München 1994, 79-94. [.ps.gz]

T. Baldin and G. Bleumer, CryptoManager++ -- An object oriented software library for cryptographic mechanisms; 12th IFIP International Conference on Information Security (IFIP/Sec '96), Chapman & Hall, London 1996, 489-491. [.ps.gz]

D. Balfanz and L. Gong, Experience with Secure Multi-Processing in Java, Proceedings of the 18th IEEE International Conference on Distributed Computing Systems (ICDCS), Amsterdam, Netherlands, May 1998. [.ps.gz]

J. Bar-Ilan and D. Beaver, Non-Cryptographic Fault-Tolerant Computing in a Constant Expected Number of Rounds of Interaction (extended abstract); Proceedings of **PODC**, ACM, 1989, 201-209. [.pdf]

R. Bar-Yehuda, B. Chor, E. Kushilevitz, and A. Orlitsky, Privacy, Additional Information, and Communication, IEEE IT 39(6), 1993, pp. 1930-1943. [.ps.Z]

N. Baric and B. Pfitzmann, Collision-Free Accumulators and Fail-Stop Signature Schemes Without Trees; Eurocrypt '97, LNCS 1233, Springer-Verlag, Berlin 1997, 480-494. [.ps.gz]

E. Basturk, M. Bellare, C. S. Chow, and R. Guerin, Secure transport protocols for high-speed networks, IBM Research Report 19981, March, 1994.

O. Baudron, H. Gilbert, L. Granboulan, H. Handschuh, A. Joux, P. Nguyen, F. Noilhan, D. Pointcheval, T. Pornin, G. Poupard, J. Stern, and S. Vaudenay, Report on the AES Candidates, Proceedings of the Second AES Candidate Conference, Rome, Italy, 1999. [.pdf]

B. Baum-Waidner, B. Pfitzmann, and M. Waidner, Unconditional Byzantine Agreement with Good Majority; STACS'91, LNCS 480, Springer-Verlag, Heidelberg 1991, 285-295. [.ps.gz]

D. Bayer, S. Haber, and W. Stornetta, Improving the Efficiency and Reliability of Digital Time-Stamping, Sequences II: Methods in Communication, Security, and Computer Science, eds. R. Capocelli, A. DeSantis, and U. Vaccaro, Springer-Verlag, 1993, pp. 329-334. [.pdf]

P. Beauchemin, G. Brassard, C. Crépeau, C. Goutier, and C. Pomerance, Two observations on probabilistic primality testing; In Advances in Cryptology: Proceedings of Crypto '86, volume 263 of Lecture Notes in Computer Science, pages 443-450. Springer-Verlag, 1987. [.ps.gz]

P. Beauchemin, G. Brassard, C. Crépeau, C. Goutier, and C. Pomerance, The generation of random

numbers that are probably prime, Journal of Cryptology, 1(1):53-64, 1988. [.ps]

D. Beaver, S. Micali, and P. Rogaway, The Round Complexity of Secure Protocols (extended abstract); Proceedings of the 22nd **STOC**, ACM, 1990, 503-513. [.ps] [.ps.gz]

D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, Security with Low Communication Overhead (extended abstract), Advances in Cryptology - Crypto '90 Proceedings, Springer-Verlag, 1991, 62-76. [.pdf]

D. Beaver, J. Feigenbaum, J. Kilian, and P. Rogaway, Locally Random Reductions: Improvements and Applications, Journal of Cryptology, 10 (1997), pp. 17-36. [.pdf] [.ps]

D. Beaver, Commodity-Based Cryptography (extended abstract); Proceedings of the 29th **STOC**, ACM, 1997, 446-455. [.pdf]

D. Beaver and S. Haber, Cryptographic Protocols Provably Secure Against Dynamic Adversaries (extended abstract); Advances in Cryptology - Eurocrypt '92, Springer-Verlag, 1993, 307-323. [.pdf]

D. Beaver, J. Feigenbaum, and V. Shoup, Hiding Instances in Zero-Knowledge Proof Systems (extended abstract), in Advances in Cryptology - Crypto '90, Lecture Notes in Computer Science, vol. 537, Springer, Berlin, 1991, pp. 326-338. [.pdf]

D. Beaver, S. Micali, and P. Rogaway, The round complexity of secure protocols; Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing, (STOC 90), 1990, 503-513. [.ps] [.ps.gz]

D. Beaver, Foundations of Secure Interactive Computing (extended abstract); Advances in Cryptology - Crypto '91 Proceedings, Springer-Verlag, 1992, 377-391. [.pdf]

D. Beaver and S. Goldwasser, Multiparty Computation with Faulty Majority, Advances in Cryptology: Crypto '89, ed. Gilles Brassard. [.pdf]

D. Beaver and N. So, Global, Unpredictable Bit Generation Without Broadcast (extended abstract); Advances in Cryptology - Eurocrypt '93, Springer-Verlag, 1994, 424-434. [.pdf]

D. Beaver, Efficient Multiparty Protocols Using Circuit Randomization (extended abstract); Advances in Cryptology - Crypto '91 Proceedings, Springer-Verlag, 1992, 420-432. [.pdf]

D. Beaver, How to Break a "Secure" Oblivious Transfer Protocol (extended abstract); Advances in Cryptology - Eurocrypt '92, Springer-Verlag, 1993, 285-296. [.pdf]

D. Beaver, J. Feigenbaum, R. Ostrovsky, and V. Shoup, Instance-Hiding Proof Systems; submitted for journal publication. Available as DIMACS Technical Report 93-65, Rutgers University, Piscataway, 1993. [.ps.Z]

R. Beigel and J. Feigenbaum, On Being Incoherent Without Being Very Hard, Computational Complexity, 2 (1992), pp. 1-17.

A. Beimel, Y. Isahi, T. Malkin, and E. Kushilevitz, One-way functions are essential for single-server private information retrieval, Proc. of the 31st Annu. ACM Symp. on the Theory of Computing (STOC), pp. 89-98, 1999. [.ps]

A. Beimel and B. Chor, Secret Sharing with Public Reconstruction, IEEE Trans. on Info. Theory, 44 (5):1887-1896, 1998. Extended abstract in Crypto '95. [.ps]

A. Beimel and M. Franklin, Reliable communication over partially authenticated networks, Theoretical Computer Science, (220)1:185--210, 1999. Preliminary version in WDAG '97, volume 1320 of LNCS, pages 245-259, Springer, 1997. [.ps]

A. Beimel, Secure Schemes for Secret Sharing and Key Distribution, Ph.D. Thesis, Dept. of Computer Science, Technion, 1996. [.ps]

A. Beimel, T. Malkin, and S. Micali, The All-or-Nothing Nature of Two-Party Secure Computation, CRYPTO '99., vol. 1666 of LNCS, pages 80 - 97, 1999. [.ps]

A. Beimel and B. Chor, Universally ideal secret sharing schemes. IEEE Trans. on Info. Theory, 40 (3):786-794, 1994. Extended abstract in Crypto '92. [.ps]

Capability-Based Computer Systems

1

Capability- and Object-Based System Concepts

Although the complexity of computer applications increases yearly, the underlying hardware architecture for applications has remained unchanged for decades. It is, therefore, not surprising that the demands of modern applications have exposed limitations in conventional architectures. For example, many conventional systems lack support in:

1. *Information sharing and communications.* An essential system function is the dynamic sharing and exchange of information, whether on a timesharing system or across a network. Fundamental to the sharing of storage is the addressing or naming of objects. Sharing is difficult on conventional systems because addressing is local to a single process. Sharing would be simplified if addresses could be transmitted between processes and used to access the shared data.
2. *Protection and security.* As information sharing becomes easier, users require access controls on their private data. It must also be possible to share information with, or run programs written by, other users without compromising confidential data. On conventional systems, all of a user's objects are accessible to any program which the user runs. Protection would be enhanced if a user could restrict access to only those objects a program requires for its execution.
3. *Reliable construction and maintenance of complex systems.* Conventional architectures support a single privileged mode of operation. This structure leads to monolithic design; any module needing protection must be part of the single operating system kernel. If, instead, any module could execute within a protected domain, systems could be built as a collection of independent modules extensible by any user.

Over the last several decades, computer industry and university scientists have been searching for alternative architectures that better support these essential functions. One alternative architectural structure is *capability-based* addressing. Capability-based systems support the *object-based* approach to computing.

This book explains the capability/object-based approach and its implications, and examines the features, advantages, and disadvantages of many existing designs. Each chapter presents details of one or more capability-based systems. Table 1-1 lists the systems described, where they were developed, and when they were designed or introduced.

System	Developer	Year	Attributes
Rice University Computer	Rice University	1959	segmented memory with "codeword" addressing
Burroughs B5000	Burroughs Corp.	1961	stack machine with descriptor addressing
Basic Language Machine	International Computers Ltd., U.K.	1964	high-level machine with codeword addressing
Dennis and Van Horn Supervisor	MIT	1966	conceptual design for capability supervisor
PDP-1 Time-sharing System	MIT	1967	capability supervisor
Multicomputer/ Magic Number Machine	University of Chicago Institute for Computer Research	1967	first capability hardware system design
CAL-TSS	U.C. Berkeley Computer Center	1968	capability operating system for CDC 6400
System 250	Plessey Corp., U.K.	1969	first industrial capability hardware and software system
CAP Computer	University of Cambridge, U.K.	1970	capability hardware with microcode support
Hydra	Carnegie-Mellon University	1971	object-based multi-processor O.S.
StarOS	Carnegie-Mellon University	1975	object-based multi-processor O.S.
System/38	IBM, Rochester, MN.	1978	first major commercial capability system, tagged capabilities
iAPX 432	Intel, Aloha, OR.	1981	highly-integrated object-based micro-processor system

Table 1-1: Major Descriptor and Capability Systems

Before surveying these systems at a detailed architectural level, it is useful to introduce the concepts of capabilities and object-based systems. This chapter defines the concept of capability, describes the use of capabilities in memory addressing and protection, introduces the object-based programming approach, and relates object-based systems to capability-based addressing.

Simplified examples of capability-based and conventional computer systems are presented throughout this chapter. These examples are meant to introduce the capability model by contrasting it with more traditional addressing mechanisms. In fact, many design choices are possible in both domains, and many conventional systems exhibit some of the properties of capability systems. No one of the following models is representative of all capability or conventional systems.

1.1 Capability-Based Systems

Capability-based systems differ significantly from conventional computer systems. Capabilities provide (1) a single mechanism to address both primary and secondary memory, and (2) a single mechanism to address both hardware and software resources. While solving many difficult problems in complex system design, capability systems introduce new challenges of their own.

Conceptually, a capability is a token, ticket, or key that gives the possessor permission to access an entity or object in a computer system. A capability is implemented as a data structure that contains two items of information: a *unique object identifier* and *access rights*, as shown in Figure 1-1.

The identifier *addresses* or *names* a single object in the computer system. An object, in this context, can be any logical or physical entity, such as a segment of memory, an array, a file, a

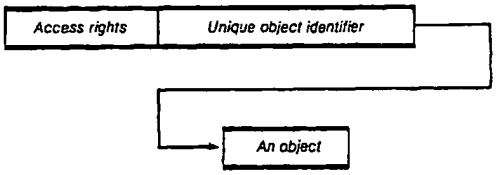


Figure 1-1: A Capability

line printer, or a message port. The access rights define the *operations* that can be performed on that object. For example, the access rights can permit read-only access to a memory segment or send-and-receive access to a message port.

Each user, program, or procedure in a capability system has access to a *list of capabilities*. These capabilities identify all of the objects which that user, program, or procedure is permitted to access. To specify an object, the user provides the *index* of a capability in the list. For example, to output a record to a file, the user might call the file system as follows:

```
PUT( file_capability , "this is a record" );
```

The capability specified in the call serves two purposes. First, it identifies the file to be written. Second, it indicates whether the operation to be performed (PUT in this case) is permitted.

A capability thus provides addressing and access rights to an object. Capabilities are the basis for object protection; a program cannot access an object unless its capability list contains a suitably privileged capability for the object. Therefore, the system must prohibit a program from directly modifying the bits in a capability. If a program could modify the bits in a capability, it could forge access to any object in the system by changing the identifier and access rights fields.

Capability system integrity is usually maintained by prohibiting direct program modification of the capability list. The capability list is modified only by the operating system or the hardware. However, programs can obtain new capabilities by executing operating system or hardware operations. For example, when a program calls an operating system routine to create a new file, the operating system stores a capability for that file in the program's capability list. A capability system also provides other capability operations. Examples include operations to:

1. Move capabilities to different locations in a capability list.
2. Delete a capability.
3. Restrict the rights in a capability, producing a less-privileged version.
4. Pass a capability as a parameter to a procedure.
5. Transmit a capability to another user in the system.

Thus, a program can execute direct control over the movement of capabilities and can share capabilities, and therefore, objects, with other programs and users.

It is possible for a user to have several capability lists. One list will generally be the master capability list containing capabilities for secondary lists, and so on. This structure is similar to a multi-level directory system, but, while directories address only files, capabilities address objects of many types.

1.1.1 Memory Addressing in Computer Systems

This section presents simplified models for both conventional and capability-based memory addressing systems. Although capabilities can control access to many object types, early capability-based systems concentrated on using capabilities for primary memory addressing. The first use of capabilities for memory protection was in the Chicago Magic Number Machine [Fabry 67, Yyngve 68], and an early description of capability-based memory protection appeared in Wilkes' book on timesharing systems [Wilkes 68]. Later, [Fabry 74] described the advantages of capabilities for generalized addressing and sharing.

For purposes of a simplified model, consider a conventional computer supporting a multiprogramming system in which each program executes within a single process. A program is divided into a collection of segments, where a segment is a contiguous section of memory that represents some logical entity, such as a procedure or array. A process defines a program's address space: that is, the memory segments it can access. The process also contains data structures that describe the user, and a directory that contains the names of a set of files. These files represent the user's long-term storage.

When a program is run, the operating system creates a process-local segment table that defines the memory segments available to the program. The segment table is a list of *descriptors* that contain physical information about each segment. Figure 1-2 shows example formats for a process virtual address and segment table descriptor. The operating system loads various segments needed by the program into primary memory, and loads the segment table descriptors with the physical address and length of each segment. A process can then access segments by reading from or writing to virtual addresses.

Each virtual address contains two fields: the segment number and the offset of a memory element within that specified segment. On each virtual address reference the hardware uses the segment number field as an index to locate an entry in the

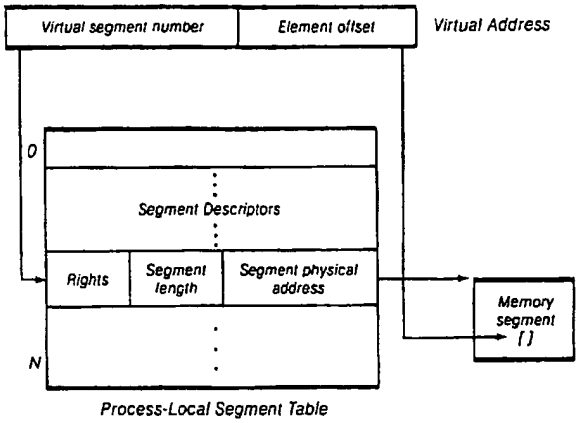


Figure 1-2: Conventional Segment Address Translation

process segment table. This descriptor contains the physical location of the segment. The length field in the descriptor is used to check that the offset in the virtual address is within the segment bounds. The rights field in the segment table entry indicates the type of access permitted to that segment (for example, read or write).

The model shown in Figure 1-2 has the following properties:

1. The system supports a segmented process virtual address space. A virtual address is local to the process and is translated through the process-local segment table.
2. A program can construct any virtual address and can attempt to read or write that address. On each reference, the hardware ensures that (a) the segment exists, (b) the offset is valid, and (c) the attempted operation is permitted. Otherwise, an error is signaled.
3. Loading of segment table entries is a privileged operation and can be accomplished only by the operating system. In general, a segment table is created at the time a program is loaded. The program then executes in a static addressing environment.
4. Sharing of segments between processes requires that the operating system arrange for both process-local segment tables to address the shared segments. If two processes wish to use the same virtual address to access a shared segment,

the segment descriptors must be in the same locations in both segment tables.

5. Any dynamic sharing of segments requires operating system intervention to load segment descriptors.

A *capability-based system* also supports the concept of a process that defines a program's execution environment. In the capability system, each process has a capability list that defines the segments it can access. Instead of the segment table descriptors available to the conventional system hardware, the capability addressing system consists of a set of *capability registers*. The *program* can execute hardware instructions to transfer capabilities between the capability list and the capability registers. The number of capability registers is generally small compared to the size of the capability list. Thus, at any time, the capability registers define a subset of the potentially accessible segments that can be physically addressed by the hardware. A simplified hardware model for this system is shown in Figure 1-3.

The model shown in Figure 1-3 has the following properties:

1. The system has a segmented virtual address space. A segment of memory can only be addressed by an instruction if a capability for that segment has been loaded into a capability register.

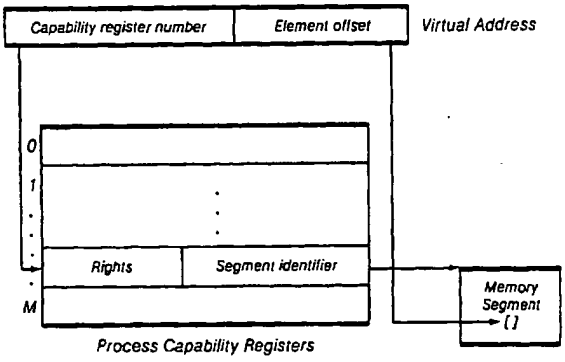


Figure 1-3: Capability Register Addressing

2. While loading of a segment descriptor in the conventional system is privileged, loading of a capability register is not. Instead of controlling the loading of the register, the capability system controls *the pattern of bits* that can be loaded. Only a valid capability can be loaded into a capability register.
3. The capability system provides a dynamically changing address space. The address space changes whenever the program changes one of the capability registers.
4. A virtual address identifies a process-local capability register. In this sense, a virtual address has similar properties to a virtual address in the conventional system. Sharing a virtual address does not in itself give access to the same segment.
5. A capability, however, is *not* process-local. Capabilities are *context independent*; that is, the segment addressed by a capability is independent of the process using that capability. A process can share a segment by copying or sending a capability from its capability list to the capability list of a cooperating process. Each of the processes can then access the segment.

One important difference between the conventional and capability approaches involves the ability of a program to affect system-wide or process-local objects. In the conventional system, a program executes within a virtual address space defined by a process. Every procedure called by that program has access to the process address space, including segments and files. Every procedure executes within an identical protection environment.

In the capability system, a procedure can only affect objects for which capability registers have been loaded. It is possible, therefore, for different procedures called by the same program to have access to different segments. Although all procedures may have the potential to load capability registers from the capability list, some procedures may choose to execute within a very small addressing sphere.

The ability to restrict the execution or addressing environment of a procedure has several benefits. First, if a procedure is allowed access only to those segments absolutely needed, the hardware can detect any erroneous references. For example, a reference past the end of an array might be caught before it destroys another variable. Second, if a procedure is found to be in error, it is easy to determine what segments might have been affected. If the segments that could have been modified were local to the procedure, recovery might be substantially easier.

Most capability systems go a step further by allowing each procedure to have a *private* capability list. A procedure can

thus protect its objects from accidental or malicious access by its callers, and a program can protect its objects from access by called procedures. Every procedure can have, in effect, its own address space. To permit a procedure access to a local object, a program can pass a capability for the object as a parameter when the procedure is called. Therefore, in a capability system, every procedure can be protected from every other procedure because each has a private capability list. When one procedure calls another, it knows that the called procedure can access only local objects for which capabilities are passed.

1.1.2 The Context of an Address

Each object in a capability system has a unique identifier. Conceptually, each object's identifier is unique for all time. That is, an identifier is assigned when an object is created and that identifier is never reused, even after the object is deleted. During the object's lifetime, its unique identifier is used within capabilities to specify the object. An attempt to use a capability with an identifier for a deleted object causes an error.

In practice, the object identifier field of a capability must be used by hardware to locate the object. From the hardware viewpoint, the identifier is an address—either the address of a segment or perhaps the address of a central descriptor that contains physical information about the segment. The need to handle addresses efficiently in hardware typically causes addresses to be small—16 or 32 bits, for example. For this reason, identifiers tend to have too few bits to be unique for all time. However, the choice of the number of bits in an identifier is an important system design decision that dictates the way in which capabilities can be used.

In conventional systems, an address is meaningful only within a single process. In a capability system, addresses (capabilities and their identifiers) are context-independent. That is, the interpretation of a capability is independent of the process using it. The unique identifier within a capability must have a system-wide interpretation. Unique identifiers must be large enough to address all of the segments likely to be in use by all executing processes at any time. This allows capabilities to be freely passed between processes and used to access shared data.

Addressing on most conventional systems is restricted in terms of time as well as context. An address is meaningful only within the lifetime of a single process. Therefore, addresses cannot be used to name objects whose lifetimes are greater than

the process creating the objects. If a process wishes to create a long-term storage object, such as a file, it must interface to the file system. Files typically require different naming, protection, and storage mechanisms than memory segments.

A significant advance made possible by capabilities is the naming and protection of both long-term and short-term objects with a single mechanism. If the identifier field is very large, it may be possible to implement identifiers unique for all time. Each object is addressed by capabilities containing its unique identifier, independent of whether it is stored in primary or secondary memory. The operating system or hardware can maintain data structures that indicate the location of each object. If a program attempts to access an object in secondary memory, the hardware or operating system can bring the object into primary memory so that the operation can be completed. From the program's point of view, however, there is a single-level address space. Capabilities, as well as data, can be saved for long periods of time and stored in secondary memory.

There are, therefore, several contexts in which an address can have meaning. For example, for:

1. Primary memory segments of a single process.
2. Primary memory segments of all existing processes.
3. All existing segments in both primary and secondary memory.

Most conventional systems support only type 1, while capabilities allow for any of the listed addressing types. More importantly, while conventional systems are concerned only with the protection of *data*, capability systems are concerned also with the protection of *addresses*. A process on a capability system cannot fabricate new addresses. As systems become more general in their addressing structure as in types 2 and 3, the protection of addresses becomes crucial to the integrity of the system.

1.1.3 Protection in Computer Systems

Lampson contrasts the capability approach with the traditional approach by showing the structure of protection information needed in a traditional operating system [Lampson 71]. Figure 1-4 depicts an access matrix showing the privileges that each system user is permitted with respect to each system object. For example, user Fred has read and write privileges to File1 and no privileges to File2, while user Sandy is allowed to read both files.

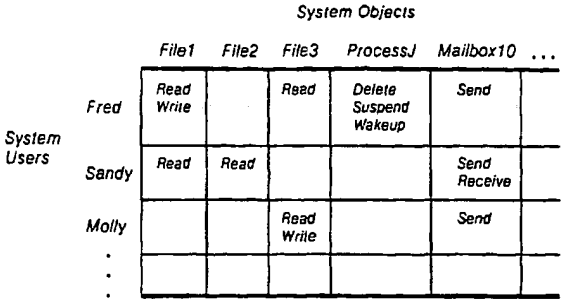


Figure 1-4: System Object Access Matrix

One conventional approach to the maintenance of protection information is *access control lists*, in which the operating system keeps an *access list* for each object in the system. Each object's list contains the names of users permitted access to the object and the privileges they may exercise. When a user attempts to access an object, the operating system checks the access list associated with that object to see if the operation is authorized. Each of the columns of Figure 1-4 represents an access control list.

The capability system offers an alternative structure in which the operating system arranges protection information by user instead of by object. A *capability list* is associated with each user in the system. Each capability contains the name of an object in the system and the user's permitted privileges for accessing the object. To access an object, the user specifies a capability in the local capability list. Each of the rows of Figure 1-4 represents a capability list. Figure 1-5 shows an access list

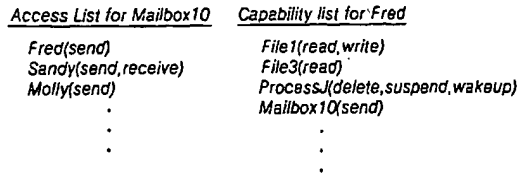


Figure 1-5: Access Control and Capability Lists

and a capability list derived from the protection matrix in Figure 1-4.

One important difference between the capability list and access list is the user's ability to *name* objects. In the access list approach, a user can attempt to name any object in the system as the target of an operation. The system then checks that object's access list. In the capability system, however, a user can only name those objects for which a capability is held: that is, to which some access is permitted.

In either case, the integrity of the system is only as good as the integrity of the data structures used to maintain the protection information. Both access control list and capability list mechanisms must be carefully controlled so that users cannot gain unauthorized access to an object.

Similar protection options exist outside the computer world. A useful analogy is the control of a safe deposit box. Suppose, for example, that Carla wishes to keep all of her valuables in a safe deposit box in the bank. On occasion, she would like one or more trustworthy friends to make deposits or withdrawals. There are basically two ways that the bank can control access to the box. First, the bank can maintain a list of people authorized to access the box. To make a transaction, Carla or any of her friends must prove their identity to the bank's satisfaction. The bank checks the (access control) list for Carla's safe deposit box and allows the transaction if the person is authorized. Or, instead of maintaining a list, the bank can issue Carla one or more keys to her safe deposit box. If Carla needs to have a friend access the box, she simply gives a key to the friend.

A number of observations can be made about these two alternative protection systems. The properties of the access list scheme are:

1. The bank must maintain a list for each safe deposit box.
2. The bank must ensure the validity of the list at all times (e.g., it cannot allow the night watchman to add a name).
3. The bank must be able to verify the identity of those asking to use a box.
4. To allow a new person to use the box, the owner must visit the bank, verify that he or she is the owner of the box, and have the new name added to the list.
5. A friend cannot extend his or her privilege to someone else.
6. If a friend becomes untrustworthy, the owner can visit the bank and have that person's name removed from the list.

The alternative scheme involving keys has the following properties:

1. The bank need not be involved in any transactions once the keys are given, except to allow a valid keyholder into the vault.
2. The physical lock and key system must be relatively secure; that is, it must be extremely difficult to forge a key or to pick the lock on a safe deposit box.
3. The owner of a box can simply pass a key to anyone who needs to access the box.
4. Once a key has been passed to a friend, it is difficult to keep them from giving the key to someone else.
5. Once a friend has made a transaction, the owner can ask for the key back, although it may not be possible to know whether or not the friend has made a copy.

The advantage of the key-based system is ease of use for both the bank and customer. However, if today's friends are likely to become tomorrow's enemies, the access list has the advantage of simple guaranteed access removal. Of course, the access control list and the key (or capability) systems are not mutually exclusive, and can be combined in either the computer or banking world to provide the advantages of both systems for increased protection.

1.2 The Object-Based Approach

Over the last few decades, several areas of computer science have converged on a single approach to system design. This approach, known as *object-based computing*, seeks to raise the level of abstraction in system design. The events that have encouraged object-based design include:

1. Advances in computer architecture, including capability systems and hardware support for operating systems concepts.
2. Advances in programming languages, as demonstrated in Simula [Dahl 66], Pascal [Jensen 75], Smalltalk [Ingalls 78], CLU [Liskov 77], and Ada [DOD 80].
3. Advances in programming methodology, including modularization and information hiding [Parnas 72] and monitors [Hoare 74].

This section introduces the object approach and discusses its relationship to capability-based computer systems.

What is object-based computing? Simply stated, the object approach is a method of structuring systems that supports *ab-*

straction. It is a philosophy of system design that decomposes a problem into (1) a set of *abstract object types*, or resources in the system, and (2) a set of *operations* that manipulate *instances* of each object type.

To make this idea more concrete, consider the following simplified example. Imagine that we are programming a traffic simulation for a city. First, define a set of objects that represent, abstractly, the fundamental entities that make up the traffic system. Some of the object *types* for the traffic simulation might be:

- passenger
- bus
- bus stop
- taxi
- car

Then, for each object type, define the operations that can be performed. Bus objects, for example, might support the operations:

- PUT_BUS_INTO_SERVICE(bus_number)
- MOVE_BUS(bus_number, bus_stop)
- LOAD_PASSENGERS(bus_number, passenger_list)
- UNLOAD_PASSENGERS(bus_number, passenger_list)
- GET_PASSENGER_COUNT(bus_number)
- GET_POSITION(bus_number)
- REMOVE_BUS_FROM_SERVICE(bus_number)

Each bus operation accepts a bus number as a parameter. At any time there may be many bus objects in the system, and we identify each bus by a unique number. Each of these bus objects is an *instance* of the *type* bus. The *type* of an object identifies it as a member of a class of objects that share some behavioral properties, such as the set of operations that can be performed on them.

What has been gained by defining the system in this way? First, there now exist a fundamental set of objects and operations for the simulation. We can now implement the procedures to perform the operations on each type of object. Since only a limited number of procedures operate on each object type, access to the internal data structures used to maintain the state of each type can be restricted. This isolation of the knowledge of those data structures should simplify any future

changes to one of the object abstractions because only a limited set of procedures is affected.

Second, and more importantly, we have raised the *level of abstraction* in the simulation program. That is, we can now program the simulation using buses, passengers, and bus stops as the fundamental objects, instead of bits, bytes, and words, which are normally provided by the underlying hardware. The buses and passengers are our data types just as bits and bytes are the data types supported in hardware. The simulation program will consist mainly of control structures plus procedure calls to perform operations on instances of our fundamental objects.

Of course, in this example, the procedures implementing the operations are programmed using lower-level objects, such as bytes, words, and so on. Or, they may be further decomposed into simpler abstract objects that are then implemented at a low level. Object-based systems provide a fundamental set of objects that can be used for computing. From this basis, the programmer constructs new higher-level object types using combinations of the fundamental objects. In this way the system is extended to provide new features by creating more sophisticated abstractions.

This methodology aims to increase productivity, improve reliability, and ease system modification. Through the use of well-defined and well-controlled object interfaces, systems designers hope to simplify the construction of complex computer systems.

1.2.1 Capabilities and Object-Based Systems

In the simulation example, each object is identified by a unique number. To move a bus from one stop to another, we call the MOVE_BUS operation with the unique number of the bus to move. For purposes of the simple simulation, a small set of integers suffices to identify the buses or other objects. No protection is needed because these objects are implemented and used by a single program.

The use of the object approach to build operating system facilities presents different requirements. For example, suppose we wish to build a calendar system to keep track of scheduled meetings, deadlines, reminders, and so on. The fundamental object of the calendar system, from the user's point of view, is a calendar object. Our calendar management system provides routines that create a new calendar, and modify,

query, or display an existing calendar. Many users in the system will, of course, want to use this facility.

Several familiar issues now arise: (1) how does a user name a calendar object, (2) how is that calendar protected from access by other users, and (3) how can calendars be shared under controlled circumstances? Only the owner of a calendar should be able to make changes, and the annotations in each calendar must be protected from other users, since they might contain confidential information. However, a user might permit selected other users to check if he or she is busy during a certain time, in order to automate the scheduling of meetings.

Capabilities provide a solution to these problems. When a user creates a new calendar, the calendar creation routine allocates a segment of memory for which it receives a capability. This segment is used to store data structures that will hold the calendar's state. The create routine uses this capability to initialize the data structures, and then returns it to the caller as proof of ownership of the calendar. In order to later modify or query the calendar, the user specifies the returned capability; the capability identifies the calendar and allows the modify or query procedure to gain access to the data structures. Only a user with a valid capability can access a calendar.

A weakness with this scenario is that the calendar system cannot prevent the calendar owner from using its capability to access the data structures directly. The calendar system would like to protect its data structures both to ensure consistency and to guarantee that future changes in data format are invisible outside of the subsystem. In addition, if a user passes a calendar capability to another user, the second user can then modify the data structures or read confidential information.

These problems exist because the calendar system returns a fully-privileged calendar capability to the user. Instead, what is needed is a capability that identifies a specific calendar and is proof of ownership, but does not allow direct access to the underlying data structures. In other words, the calendar system would like to return only *restricted* capabilities to its clients. However, the calendar system must retain the ability to later *amplify* the privileges in one of its restricted capabilities so that it can access the data structures for a calendar.

There are several ways of providing type managers with this special ability. (These mechanisms are examined in detail throughout the book.) However, given this power over capabilities for its objects, a type manager can ensure that its clients operate only through the well-defined object operation interface. A client can pass a capability parameter to the type man-

ager when requesting a service, but cannot otherwise use the capability to read or write the object it addresses. This facility is fundamental to any system that allows creation and protection of new system types.

1.3 Summary

The capability concept can be applied in hardware and software to many problems in computer system design. Capabilities provide a different way of thinking about addressing, protection, and sharing of objects. Some of the properties of capabilities illustrated in this chapter include their use in:

1. Addressing primary memory in a computer system.
2. Sharing objects.
3. Providing a uniform means of addressing short- and long-term storage.
4. Support for a dynamic addressing environment.
5. Support for data abstraction and information hiding.

These, of course, are advantages of capability-based systems. The most important advantage is support for object-based programming. Object-based programming methodology seeks to simplify the design, implementation, debugging, and maintenance of sophisticated applications. While capabilities solve a number of system problems, their use raises a whole new set of concerns. And, as is often the case in computer system design, the concept is much simpler than the implementation.

The remainder of this book is devoted to examining many different capability-based and object-based designs. The characteristics of each system are described with emphasis on addressing, protection, and object management. Each system represents a different set of tradeoffs and presents different advantages and disadvantages. When comparing the systems, consider the differences in goals, technologies, and resources available to the system developers.

The final chapter of this book considers issues in capability system design common to all of the systems described. A few of the questions to be considered follow. It may be useful to remember these questions when examining each system design.

1. What is the structure of an address?
2. How is a capability represented? How is a capability used to locate an object?

3. How are capabilities protected?
4. What is the lifetime of a capability?
5. What types of objects are supported by the hardware and software?
6. What is the lifetime of an object?
7. How can users extend the primitive set of objects provided by the base hardware and software?

1.4 For Further Reading

The concept of capability is formally defined in the 1966 paper by Dennis and Van Horn [Dennis 66]. Chapter 3 examines this paper in some detail. The paper by Fabry [Fabry 74] compares capability addressing and conventional segmented addressing of primary memory, while Redell [Redell 74a] describes issues in capability systems and the use of sealing mechanisms that support the addition of new object types to a system. These papers are a fundamental part of capability literature.

Capability systems have been discussed in various contexts. Two papers by Lampson [Lampson 69 and Lampson 71] describe the requirements for protection in operating systems and the capability protection model. The surveys by Linden [Linden 76] and Denning [Denning 76], which appeared in a special issue of *ACM Computing Surveys*, describe capability systems and their relationship to security and fault tolerance in operating systems.

The architecture books by Myers [Myers 82] and Iliffe [Iliffe 82] also discuss some of the systems described in this book. Myers' book contains details of Sward [Myers 80], a capability-based research system built at IBM that is omitted here. A capability system model, as well as discussion of some existing capability systems, appears in the book by Gehringer [Gehringer 82]. Jones [Jones 78a] provides a good introduction to the concepts of object-based programming.

The Burroughs B5000 computer. (Courtesy Burroughs Corporation.)

Divertible Protocols and Atomic Proxy Cryptography

Matt Blaze Gerrit Bleumer Martin Strauss

AT&T Labs – Research
Florham Park, NJ 07932 USA
{mab,bleumer,mstrauss}@research.att.com

Abstract. First, we introduce the notion of divertibility as a protocol property as opposed to the existing notion as a language property (see Okamoto, Ohta [OO90]). We give a definition of protocol divertibility that applies to arbitrary 2-party protocols and is compatible with Okamoto and Ohta's definition in the case of interactive zero-knowledge proofs. Other important examples falling under the new definition are blind signature protocols. We propose a sufficiency criterion for divertibility that is satisfied by many existing protocols and which, surprisingly, generalizes to cover several protocols not normally associated with divertibility (e.g., Diffie-Hellman key exchange). Next, we introduce *atomic proxy cryptography,* in which an *atomic proxy function,* in conjunction with a public *proxy key,* converts ciphertexts (messages or signatures) for one key into ciphertexts for another. Proxy keys, once generated, may be made public and proxy functions applied in untrusted environments. We present atomic proxy functions for discrete-log-based encryption, identification, and signature schemes. It is not clear whether atomic proxy functions exist in general for all public-key cryptosystems. Finally, we discuss the relationship between divertibility and proxy cryptography.

1 Introduction

This paper investigates two general ways in which an intermediary sitting between the participants of a 2-party protocol might transform the communication messages without "destroying" the protocol. First, we consider *protocol divertibility,* in which the (honest) intermediary, called a *warden,* randomizes all messages so that the intended underlying protocol succeeds, but information contained in subtle deviations from the protocol (for example, information coded into the values of supposedly random challenges) will be obliterated by the warden's transformation. Next, we introduce *atomic proxy cryptography,* in which two parties publish a *proxy key* that allows an untrusted intermediary to convert ciphertexts encrypted for the first party directly into ciphertexts that can be decrypted by the second. The intermediary learns neither cleartext nor secret keys.

Our paper is organized as follows. In Section 2 we discuss divertible protocols. In Section 2.1 we define protocol divertibility. We propose a slightly stricter

definition than the original one by Okamoto and Ohta [OO90]. In Section 2.2, we present a sufficiency criterion for divertibility. Its usefulness is demonstrated by many examples of known diverted protocols from the literature. Also many known blind signature protocols can be interpreted as diverted proofs of knowledge and in this form they satisfy our criterion (see [Bleu97]). In Section 3, we introduce atomic proxy cryptography and propose a taxonomy for proxy schemes. In Sections 3.1 to 3.3 we give proxy schemes for encryption, identification, and signature. In Section 4, we discuss the deeper relationship between protocol divertibility and proxy cryptography.

2 Divertible Protocols

The idea of divertibility entered the cryptographic literature during the mid 80's with applications to identification protocols. The basic observation was that some 2-party identification protocols could be extended by placing an intermediary—called a warden for historical reasons [Sim84]—between the prover and verifier so that, even if both parties conspire, they cannot distinguish talking to each other through the warden from talking directly to a hypothetical honest verifier and honest prover, respectively. Since identification protocols were developed in close relation to interactive zero-knowledge proofs (ZKP), Okamoto and Ohta [OO90] (and later Desmedt and Burmester [BD91] and Ihto et al [ISS91]) established the notion of divertibility as a *language property*, i.e., a language is considered divertible if it can be recognized by a diverted interactive zero-knowledge proof system. In this paper, we establish divertibility as a *2-party protocol property*, which is orthogonal to zero knowledge or any other particular protocol property.

2.1 Definitions

In order to deal with protocols of more than two parties, we generalize the notion of *interactive Turing machine* (ITM) by Goldwasser et al [GMR89]. Then we define connections of ITMs and finally give the definition of protocol divertibility.

Definition 1 ((m,n)-Interactive Turing Machine).
An *(m,n)-Interactive Turing Machine* (*(m,n)-ITM*) is a Turing machine with $m \in \mathbb{N}$ read-only *input tapes*, m write-only *output tapes*, m read-only *random tapes*, a *work tape*, a read-only *auxiliary tape*, and $n \in \mathbb{N}_0$ pairs of *communication tapes*. Each pair consists of one read-only and one write-only tape that serves for reading in-messages from or writing out-messages to another ITM. (The purpose of allowing $n = 0$ will become clear below.) The random tapes each contain an infinite stream of bits chosen uniformly at random. Read-only tapes are readable only from left to right. If the string to the right of a read-only head is empty, then we say the tape is *empty*.

Associated to an ITM is a *security parameter* $k \in \mathbb{N}$, a family $D = \{D_\pi\}_\pi$ of tuples of domains, a probabilistic *picking algorithm* $pick(k)$ and an encoding

scheme S. Each member

$$D_\pi = (In_\pi^{(1)}, \ldots, In_\pi^{(m)}, Out_\pi^{(1)}, \ldots, Out_\pi^{(m)}, \Omega_\pi^{(1)}, \ldots, \Omega_\pi^{(m)}, \\ (IM_\pi^{(1)}, OM_\pi^{(1)}), \ldots, (IM_\pi^{(n)}, OM_\pi^{(n)}))$$

of D contains one input (output, choice, in-message, out-message) domain for each of the m input (output, random) tapes and n (read-only, write-only) communication tapes. The algorithm $pick(k)$ on input some security parameter k outputs a family index π. Finally, there is a polynomial $P(k)$ so that for each π chosen by $pick(k)$, S encodes all elements of all domains in D_π as bitstrings of length $P(k)$.

ITMs proceed in rounds. During each round, an ITM first reads the messages from all its read-only communication tapes, then performs some computations and finally writes a message to each of its write-only communication tapes. It may write an empty string—denoted ε. If, at the beginning of a round, an ITM finds all its input tapes and all its read-only communication tapes empty, then it performs a last computation, writes empty strings to all its write-only communication tapes, writes results to all its output tapes, and then stops. The overall number of reading, writing and computation steps during an execution of an ITM is bound by a polynomial in the security parameter k.

An (m, n)-ITM is an *m-party protocol* if $n = 0$, and linear if $n \leq 2$. The *native functions* of an ITM A are defined as the family

$$nativ_\pi : \prod_{i=1}^m \Omega_{\pi,i} \times \prod_{i=1}^m In_{\pi,i} \times \prod_{j=1}^n IM_{\pi,j} \to \prod_{j=1}^n OM_{\pi,j}$$

of functions that, on input (rnd, in, im), return the respective out-messages that A would write to its write-only communication tapes would it read this data from its random, input and read-only communication tapes.

Let A be an (m_A, n)-ITM and B be an (m_B, n)-ITM, which together make up a protocol $P = \langle A, B \rangle$. Let $m^* \leq \min(m_A, m_B)$ be the number of pairs of communication tapes shared by A and B. Then the *view* of A on B on respective inputs, denoted as,

$$view_B^{(A)} P([in_{A,1}, \ldots, in_{A,m_A}]^A, [in_{B,1}, \ldots, in_{B,m_B}]^B) \ ,$$

is defined as everything that A sees from B, i.e., the probability distribution of all m^*-tuples of pairs of in-messages sent by A to B and out-messages returned from B to A, where the probabilities are taken over the choices of the viewer A.[1]

For m-party protocols P, we adopt the following interface notation:

$$(out_1, \ldots, out_m) \leftarrow P(in_1, \ldots, in_m) \ ,$$

where the left arrow indicates a probabilistic assignment. If the inputs or outputs consist of several components, we delimit them by square brackets.

[1] This is a generalization of the definition given by Goldwasser, Micali and Rackoff [GMR89].

Definition 2 (Connections of ITMs).
Let A be an (m_A, n_A)-ITM and B be an (m_B, n_B)-ITM with equal picking algorithm *pick*. Then a connection $C = \langle A, B \rangle$ is any ITM consisting of A and B sharing $c \leq \min\{n_A, n_B\}$ pairs of their communication tapes. The picking algorithm of C is *pick*, and the domains of C are defined as the cartesian products of the respective domains of A and B. $\diamond$

Obviously, the linear connection operator $\langle \bullet, \bullet \rangle$ is associative and we can therefore omit brackets in the usual way:

$$\langle A, B, C \rangle \stackrel{\text{def}}{=} \langle \langle A, B \rangle, C \rangle = \langle A, \langle B, C \rangle \rangle \ .$$

All connections we consider in the following are linear and have a small constant number of rounds.

Definition 3 (Divertibility of Protocols).
Let $P = \langle A, B \rangle$ be a two-party protocol with interface $P([y, x_A]^A, [y, x_B]^B)$ and input domains $In_\pi = (Y_\pi \times X_{A,\pi}) \times (Y_\pi \times X_{B,\pi})$. Common inputs y are taken from Y_π, whereas private inputs x_A, x_B are taken from $X_{A,\pi}$ and $X_{B,\pi}$, respectively. The product domain of private inputs is denoted $X_\pi = X_{A,\pi} \times X_{B,\pi}$. Furthermore, let $R = \{R_\pi\}_\pi$ be a family of relations $R_\pi \subseteq Y_\pi \times X_\pi$.

The protocol P is called *perfectly (computationally) divertible* over R iff a (1,2)-ITM W exists such that the following properties hold:

EXTENSIBILITY: For all indices π, all common and private inputs $(y, x_A, x_B) \in R_\pi$, the ensembles of views of B on W and on A, i.e.,

$$view_W^{(B)}\langle A, W, B \rangle([y, x_A]^A, [y]^W, [y, x_B]^B), \text{ and}$$
$$view_A^{(B)}\langle A, B \rangle([y, x_A]^A, [y, x_B]^B)$$

as well as the views of A on W and on B, i.e.,

$$view_W^{(A)}\langle A, W, B \rangle([y, x_A]^A, [y]^W, [y, x_B]^B), \text{ and}$$
$$view_B^{(A)}\langle A, B \rangle([y, x_A]^A, [y, x_B]^B)$$

are equal (polynomially indistinguishable).

PERFECT (COMPUTATIONAL) INDISTINGUISHABILITY: For all polynomial-time actively adversary ITMs $\tilde{A}$, $\tilde{B}$, for all indices π, all common and private inputs $(y, x_A, x_B) \in R_\pi$ and all polynomial size strings q representing shared a priori knowledge of $\tilde{A}$ and $\tilde{B}$, the ensembles of simultaneous views of $\tilde{A}$ and $\tilde{B}$ upon W and of their views upon honest B and A, i.e.,

$$view_W^{(\tilde{A},\tilde{B})}\langle \tilde{A}, W, \tilde{B} \rangle([y, x_A, q]^{\tilde{A}}, [y]^W, [y, x_B, q]^{\tilde{B}}) \quad \text{and}$$
$$(view_B^{(\tilde{A})}\langle \tilde{A}, B \rangle([y, x_A, q]^{\tilde{A}}, [y, x_B]^B), view_A^{(\tilde{B})}\langle A, \tilde{B} \rangle([y, x_A]^A, [y, x_B, q]^{\tilde{B}}))$$

are equal (polynomially indistinguishable).[2] [3]

An ITM W that satisfies extensibility and perfect (computational) indistinguishability is said to *perfectly (computationally) divert* protocol P over R. $\diamond$

Divertibility as defined by Okamoto, Ohta [OO90] and almost equivalently by Itoh et al [ISS91] has been introduced as a *language property.* A language L is considered divertible, if there exists a diverted zero knowledge proof system for proving membership in L. In contrast, we define divertibility as a *2-party protocol property.* The main difference between the two definitions is that we ask for a concrete protocol P to be divertible, whereas they ask for existence of a divertible protocol meeting a certain specification S (namely to be a zero-knowledge proof). Consequently, Definition 3 (extensibility) relates the two interfaces of the diverted protocol P' to the interface of the given protocol P, where their definition relates them to S. Another difference is, that we suggest a stronger definition than Okamoto and Ohta's. We require Indistinguishability even for two attackers $\tilde{A}$ and $\tilde{B}$ who *know of each other* (a-priori common knowledge q) and who therefore know which of their views result from the same diverted protocol instance. We discuss this further in Section 2.4.

An immediate consequence of the definition is that if a protocol P is divertible, then we can insert second and third wardens and we, again, obtain a diverted protocol.

2.2 Main Divertibility Result

Theorem 4 (Criterion for Perfect Divertibility).
Let $P = \langle A, B \rangle$ be a two-party protocol with interface $P([y, x_A]^A, [y, x_B]^B)$. Let the input domains be $(Y_\pi \times X_{A,\pi}) \times (Y_\pi \times X_{B,\pi})$, the random domains be $\Omega_{A,\pi} \times \Omega_{B,\pi}$, the out-message domains be $OM_{A,\pi} \times OM_{B,\pi}$, and let the native functions of A and B be

$$nativ_{A,\pi} : \Omega_{A,\pi} \times Y_\pi \times X_{A,\pi} \times OM_{B,\pi} \rightarrow OM_{A,\pi} \ ,$$
$$nativ_{B,\pi} : \Omega_{B,\pi} \times Y_\pi \times X_{B,\pi} \times OM_{A,\pi} \rightarrow OM_{B,\pi} \ .$$

Furthermore, let $R = \{R_\pi\}_\pi$ be a family of relations $R_\pi \subseteq Y_\pi \times (X_{A,\pi} \times X_{B,\pi})$, which capture the correspondence between the private and the public inputs.

Then P is perfectly divertible over R if only there exist:

(i) a family $(\Omega_\pi, \odot, 1)$ of (not necessarily commutative) groups, and

[2] By $view_B^{(A)} P$, we denote the *view* of A on B in a protocol P. This notion as well as that of *polynomial indistinguishability* of families of random variables is defined, e.g., by Goldwasser, Micali and Rackoff [GMR89].

[3] Equality (polynomial indistinguishability) is required only for the views on *complete* runs of the diverted protocol, i.e., runs that the warden has not aborted, for example, because he has detected either $\tilde{A}$ or $\tilde{B}$ cheating.

(ii) three families of functions

$$base_\pi : Y_\pi \times X_{A,\pi} \times X_{B,\pi} \to OM_{A,\pi} \times OM_{B,\pi} \ ,$$
$$join_\pi : \Omega_{A,\pi} \times \Omega_{B,\pi} \times Y_\pi \times X_{A,\pi} \times X_{B,\pi} \to \Omega_\pi \ ,$$
$$divrt_\pi : \Omega_\pi \times Y_\pi \times OM_{A,\pi} \times OM_{B,\pi} \to OM_{A,\pi} \times OM_{B,\pi} \ ,$$

with the following properties:
Function $divrt(\omega, y, o_A, o_B)$ *is defined only for* (o_A, o_B) *that live in the respective image* $OM_{\pi,y}$ *of* $native_A$ *and* $native_B$*, i.e.,*

$$OM_{\pi,y} = native_A(\Omega_A, y, x_A, o_B) \times native_B(\Omega_B, y, x_B, o_A) \ ,$$

where $(y, x_A, x_B) \in R_\pi$.[4]
Second, for each fixed $\alpha, \beta, y, x_A, x_B \in R_\pi$*, the functions,*

$$join_\pi(\alpha', \beta, y, x_A, x_B) \quad \text{and} \quad join_\pi(\alpha, \beta', y, x_A, x_B) \ ,$$

are each bijective on Ω_A *and* Ω_B*, respectively.*
(iii) a warden W *that on input two in-messages* o_A, o_B *computes two out-messages* o'_A, o'_B *such that*

$$(o'_A, o_B) = divrt(\omega, y, (o_A, o'_B)) \ .$$

Now, for every π*, for all random choices* $\alpha \in \Omega_{A,\pi}, \beta \in \Omega_{B,\pi}$*, all common and corresponding private inputs* $(y, x_A, x_B) \in R_\pi$*, and all out-messages* $o_A \in OM_{A,\pi}$*,* $o_B \in OM_{B,\pi}$ *the following three conditions must hold:*

DECOMPOSITION:

$$\begin{aligned}&(nativ_A(\alpha, y, x_A, o_B), nativ_B(\beta, y, x_B, o_A)) \\ &= divrt(join(\alpha, \beta, y, x_A, x_B), y, base(y, x_A, x_B)) \ ,\end{aligned}$$

GROUND:

$$divrt(1, y, (o_A, o_B)) = (o_A, o_B) \ ,$$

MIXED ASSOCIATIVITY:

$$divrt(\omega', y, divrt(\omega, y, (o_A, o_B))) = divrt(\omega \odot \omega', y, (o_A, o_B)) \ .$$

◇

Proof. First observe that if *divrt* satisfies the premises GROUND and MIXED ASSOCIATIVITY, then it is injective as a function of ω: For all $(o_A, o_B) \in OM_{\pi,y}$, we have:

$$\begin{aligned}(o_A, o_B) = \ & divrt(1, y, (o_A, o_B)) \\ & divrt(\omega \odot \omega^{-1}, y, (o_A, o_B)) \ \text{(for any } \omega) \\ & divrt(\omega^{-1}, y, divrt(\omega, y, (o_A, o_B))) \ .\end{aligned}$$

[4] Note that the input variables o_A and o_B in the definition of $OM_{\pi,y}$ refer to the output of $nativ_B$ and $nativ_A$, respectively. This recursion is guaranteed to terminate by the following requirement (iii) below.

So, function *divrt* turns out to be bijective on Ω_π for the entire parameter domain $OM_{\pi,y}$. We may thus write: $divrt^{-1}(\omega, \bullet) = divrt(\omega^{-1}, \bullet)$.

In order to infer extensibility and indistinguishability of P, we look separately at the out-messages between $\langle A, W\rangle$ and B and those out-messages between A and $\langle W, B\rangle$. We deal with the former case in detail and argue that the latter case can be handled analogously due to symmetry reasons. Using DECOMPOSITION and MIXED ASSOCIATIVITY, we rewrite the above mentioned out-messages as follows:

$$\begin{aligned}
&(nativ_{\langle A,W\rangle}(\omega, \alpha, y, x_A, o_B), nativ_B(\beta, y, x_B, o_A)) \\
&= divrt(\omega, y, (nativ_A(\alpha, y, x_A, o'_B), nativ_B(\beta, y, x_B, o'_A))) \\
&= divrt(\omega, y, divrt(\omega', y, base(y, x_A, x_B))), \\
&\qquad \text{where } \omega' = join(\alpha, \beta, y, x_A, x_B) \\
&= divrt(\omega' \odot \omega, y, base(y, x_A, x_B)) \\
&= (nativ_A(\alpha', y, x_A, o_B), nativ_B(\omega, \beta', y, x_B, o_A)) \ , \\
&\qquad \text{where } (\alpha', \beta') = join^{-1}(\omega' \odot \omega, y, x_A, x_B) \ .
\end{aligned}$$

It then follows from the bijectiveness of *join* and the fact that $\odot$ is a group operation that the probability of each pair of out-messages is the same over Bob's choices β and over β'. Together with the analogous result for out-messages between A and $\langle W, B\rangle$ (this is where invertibility of *divrt* is needed), this settles extensibility.

For perfect indistinguishability, we need to deal with arbitrary attackers $\tilde{A}, \tilde{B}$, instead. Assume, these attackers produce their out-messages with a certain distribution D that respects the domain of function *divrt*. Otherwise, *divrt* is undefined and the distribution could be ignored according to indistinguishability. Then by decomposition, we see that this given distribution D can also be achieved by honest Alice and Bob if Bob would chose his β according to some appropriate distribution d. Following the above rewriting, and again taking into account that join is bijective and $\odot$ is a group operation, we conclude, that the distribution of $\omega' \odot \omega$ is d because, by presumption, the warden is honest and therefore ω is uniformly distributed. Hence, the out-messages of $\langle A, W\rangle$ and B are also distributed according to D, if the probabilities are taken over β'. Together with the analogous result for out-messages between A and $\langle W, B\rangle$, this in addition settles perfect indistinguishability and therefore perfect divertibility. □

2.3 New Example of Diverted Protocol

The most prominent examples of diverted protocols in the literature are diverted interactive proofs and blind signatures. Since divertibility has been introduced only in the former context, blind signatures are a good example to illustrate the more general concept of divertibility of protocols as proposed in Definition 3. The practical value of Theorem 4 is demonstrated in [Bleu97] by proving many protocols unconditionally divertible; in particular (i) the diverted ZKP that Okamoto and Ohta used to prove their main theorem [OO90] and (ii) a

blind modified ElGamal Signature, which was presented by Horster, Michels and Petersen [HMP95] who built on ideas of Camenisch, Piveteau and Stadler [CPS95].

Here, we consider a new sort of protocol for divertibility, namely key exchange. In Figure 1, we present a diverted Diffie-Hellman key exchange protocol [DH76]. Let p be a k-bit prime ($k \in \mathbb{N}$), q be a large prime divisor of $p-1$ and G_q be the unique (multiplicative) subgroup of order q in $\mathbb{Z}_p^*$. Furthermore, $g \neq 1$ denotes a randomly chosen element of G_q. (The restriction to $g \neq 1$ asserts that g generates G_q). p, q and g are global system parameters and neither Alice nor Bob have private inputs.

$$([a'']^A, [b'']^B) \leftarrow DDH([]^A, []^W, []^B)$$

	Alice		**Warden**		**Bob**
(1)	$\alpha \in_R \mathbb{Z}_q$		$\omega \in_R \mathbb{Z}_q$		$\beta \in_R \mathbb{Z}_q$
(2)	$a \leftarrow g^\alpha$				$b \leftarrow g^\beta$
(3)		$\xrightarrow{a}$	$(a', b') \leftarrow (a^\omega, b^\omega)$	$\xleftarrow{b}$	
(4)	$a'' \leftarrow b'^\alpha$	$\xleftarrow{b'}$		$\xrightarrow{a'}$	$b'' \leftarrow a'^\beta$

Fig. 1. Diverted Diffie-Hellman Key Exchange

Proposition 5. *The warden of protocol DDH computationally diverts the Diffie-Hellman protocol between Alice and Bob over $R = \emptyset$.* ◇

Proof (Sketch). If for given (a, b), an attacker could distinguish valid from invalid diverted out-messages (a', b') with non-negligible probability, i.e., probability $\geq \frac{1}{P(k)}$ for some polynomial P, then he had broken the simultaneous discrete log assumption [CEG88]. □

2.4 Why the Previous Definition is a Little too Weak

The previous definition of divertibility by Okamoto and Ohta [OO90], and by Itoh et al [ISS91] as well, requires that two attackers $\tilde{A}, \tilde{B}$ who on the one hand form a linear 3-party protocol P' with an intermediate warden and on the other hand form 2-party protocols $\langle \tilde{A}, B \rangle$ with an honest B and $\langle A, \tilde{B} \rangle$ with an honest A cannot distinguish their views in $\langle \tilde{A}, B \rangle$ and $\langle A, \tilde{B} \rangle$ from those in *separate* instances of $\langle \tilde{A}, W, \tilde{B} \rangle$. More formally, they require indistinguishability of the two ensembles (protocol inputs exactly as in Definition 3 before):

$$(view_W^{(\tilde{A})}\langle \tilde{A}, W, \tilde{B} \rangle, view_W^{(\tilde{B})}\langle \tilde{A}, W, \tilde{B} \rangle) \tag{1}$$

$$\text{and} \quad (view_B^{(\tilde{A})}\langle \tilde{A}, B \rangle, view_A^{(\tilde{B})}\langle A, \tilde{B} \rangle). \tag{2}$$

However, the attacker model that seems to underly the literature on divertibility is stronger than expressed by the above requirement. The attackers $\tilde{A}$ and $\tilde{B}$ are usually assumed to know when they engage in a protocol with the warden and so they know which of their views result from the same protocol instances.

A good example to illustrate this difference is protocol *DDH* in Section 2.3. The two ensembles according to (1) and (2) above are equal and thus protocol DDH would have to be regarded as perfectly diverted. This is counterintuitive because the warden in DDH uses less random coins than Alice and Bob together. On the other hand, according to Definition 3, DDH is only computationally diverted, which is the most we would expect.

3 Atomic Proxy Cryptography

A basic goal of public-key encryption is to allow only the key or keys selected at the time of encryption to decrypt the ciphertext. To change the ciphertext to a different key requires re-encryption of the message with the new key, which implies access to the original cleartext and to a reliable copy of the new encryption key. Intuitively, this seems a fundamental, and quite desirable, property of good cryptography; it should not be possible for an untrusted party to change the key with which a message can be decrypted.

Here, on the other hand, we investigate the possibility of *atomic proxy functions* that convert ciphertext for one key into ciphertext for another without revealing secret decryption keys or cleartext messages. An atomic proxy function allows an untrusted party to convert ciphertext between keys without access to either the original message or to the secret component of the old key or the new key. In proxy cryptography, the holders of public-key pairs A and B create and publish a *proxy key* $\pi_{A\to B}$ such that $D(\Pi(E(m, e_A), \pi_{A\to B}), d_B) = m$, where $E(m, e)$ is the public encryption function of message m under encryption key e, $D(c, d)$ is the decryption function of ciphertext c under decryption key d, $\Pi(c, \pi)$ is the atomic proxy function that converts ciphertext c according to proxy key π, and e_A, e_B, d_A, d_B are the public encryption and secret decryption component keys for key pairs A and B, respectively. The proxy key gives the owner of B the ability to decrypt "on behalf of" A; B can act as A's "proxy." In other words, the Π function effectively allows the "atomic" computation of $E(D(c, d_A), e_B)$ without revealing the intermediate result $D(c, d_A)$.

We consider atomic proxy schemes for encryption, identification and signatures. An encryption proxy key $\pi_{A\to B}$ allows B to decrypt messages encrypted for A and an identification or signature proxy key $\pi_{A\to B}$ allows A to identify herself as B or to sign for B (i.e., transforms A's signature into B's signature). Generating encryption proxy key $\pi_{A\to B}$ obviously requires knowledge of at least the secret component of A (otherwise the underlying system is not secure) and similarly generating identification or signature proxy key $\pi_{A\to B}$ requires B's secret, but the proxy key itself, once generated, can be published safely.

Categories of proxy schemes Encryption proxy functions (and similarly but contravariantly, identification or signature proxy functions) can be categorized according to the degree of trust they imply between the two key holders. Clearly, A must (unconditionally) trust B, since the encryption proxy function by definition allows B to decrypt on behalf of A. *Symmetric* proxy functions also imply that B trusts A, e.g., because d_B can be feasibly calculated given the proxy key plus d_A. *Asymmetric* proxy functions do not imply this bilateral trust. (Note that this model implies that proxy cryptography probably makes sense only in the context of public-key cryptosystems. Any secret-key cryptosystem with an asymmetric proxy function could be converted into a public-key system by publishing one key along with a proxy key that converts ciphertext for that key into ciphertext for a second key (which is kept secret.))

We can also categorize the asymmetric proxy schemes that might exist according to the convenience in creating the proxy key. In an *active asymmetric* scheme, B has to cooperate to produce the proxy key $\pi_{A \to B}$ feasibly, although the proxy key (even together with A's secret key) might not compromise B's secret key. In a *passive asymmetric* scheme, on the other hand, A's secret key and B's public key suffice to construct the proxy key. Clearly, any passive asymmetric scheme can be used as an active asymmetric scheme, and any asymmetric scheme can be used as a symmetric scheme.

Finally, we can (informally) distinguish proxy schemes according to the "metadata" they reveal about the identity of the secret-public key-pairs being transformed. *Transparent* proxy keys reveal the original two public keys to a third party. *Translucent* proxy keys allow a third party to verify a guess as to which two keys are involved (given their public keys). *Opaque* proxy keys reveal nothing, even to an adversary who correctly guesses the original public keys (but who does not know the secret keys involved).

Proxy schemes in theory and practice The proxy relationship is necessarily transitive. If there are public proxy keys $\pi_{A \to B}$ and $\pi_{B \to C}$, then anyone can compute a proxy function for $A \to C$. Symmetric proxy schemes further establish equivalence classes of keys where the secret component of any key can be used to decrypt messages for any other key in the same class. Note that creating a single symmetric proxy key between a key in one class and a key in another effectively joins the two classes into one.

The notion of proxy cryptography is a rather natural generalization of public-key cryptography and has some nice theoretical properties. The proxy schemes we consider below have the additional property that anyone can use the proxy key $\pi_{A \to B}$ to transform the public key of A to the public key of B. For such proxy schemes, as we will see in the various examples below, certain aspects of the security of publishing a proxy key actually follow from the fact that anyone, trusted or not, can use a proxy key to transform ciphertext and keys.

For example, suppose random messages m and m' are encrypted with random secret keys a and b as $E(m, a)$, $E(m', b)$. Suppose that knowing the proxy key $\pi_{A \to B}$ enables Eve, who knows neither a nor b, to recover m or m'. Then, ignoring

B altogether and starting with just two (presumably secure) ciphertexts $E(m, a)$ and $E(m', a)$, Eve can pick a random proxy key $r = \pi_{A \to Q}$ for some Q, transform $E(m', a)$ to $E(m', q)$ (where q is the unknown secret key of Q), transform A's public key into Q's public key, and proceed with the hypothesized cryptanalysis. We conclude that if it is safe for A to publish k messages then it is safe for A and B to publish a total of k messages *and* to publish a proxy key, provided only that Eve can successfully *apply* the proxy key to transform ciphertext and public keys.

Because proxy keys are tied to specific key pairs, it is not necessary in many applications to certify or otherwise take special care in distributing them (except to prevent denial-of-service). In particular, it is generally sufficient to rely on the certification and trust established in A (for encryption) or B (for signatures) when using proxy key $\pi_{A \to B}$, since a valid proxy key can by definition only be generated with the cooperation of the owner. Furthermore, the proxy function can be safely applied at any convenient time or place, by the message's sender or receiver, or at any intermediate (and possibly untrusted) point in the network.

Proxy functions potentially also have practical utility for key management in real systems. For example, some pieces of secure hardware (*e.g.*, smartcards) limit the number of secret keys that can be stored in secure memory, while some applications might require the ability to decrypt messages for more keys than the hardware can accommodate. With proxy cryptography, once a new key is created and a corresponding proxy key generated, the secret component of the old (or new) key can be destroyed, with the (public and externally-applied) proxy key maintaining the ability to decrypt for both. In effect, proxy functions allow us to increase the number of public keys without also increasing the number of secret bits or the amount of secret computation. Because proxy functions can be computed anywhere, messaging systems, such as electronic mail, can proxy "forward" messages encrypted with one key to a recipient who holds a different key. Proxy functions make it possible to associate a single key with a network or physical address but still decrypt messages forwarded (and proxied) from other addresses. Finally, proxy functions effectively allow changing or adding a key without obtaining new certificates or altering the distribution channel for the previous public key; this could be useful when it is difficult to distribute or certify new keys (e.g., old keys were published in widely-distributed advertisements or embedded in published software, or the certification authority charges high fees for new certificates).

Security of proxy schemes and ad hoc substitutes If Alice wants Bob to be able to read her mail, instead of issuing a proxy key she might just give Bob her secret key (perhaps, obviating the need to involve Bob, by encrypting it in Bob's public key and publishing it). This would be inferior to using a proxy scheme for several reasons. First, as discussed above, Bob's computing environment may be limited and therefore incapable of automatically processing encrypted secret keys; any new software to decrypt and manage such keys would have to run within the environment trusted by Bob. Proxy processing, on the

other hand, can take place entirely outside of Alice's and Bob's trusted environments and without their active involvement. Furthermore, encrypting one's secret key with another's public key is not in general secure. The cryptosystem we present below, a variant of ElGamal [ElG85], is thought to be secure in part because the cryptanalysis problem is random-self-reducible—which allows one to assert mathematically that recovering m from the public information $\langle e_a, E(m, e_a), e_b \rangle$ is hard on average if it is hard at worst. The task of recovering m from $\langle e_a, E(m, e_a), E(d_a, e_b), e_b \rangle$, however, may be considerably easier since $E(d_a, e_b)$, in the context of e_a and e_b, may leak information about d_a—specifically, the new cryptanalysis problem is probably not random-self-reducible and due to the problem's obscurity it is not clear what, if any, mathematical guarantees of security can be given. By contrast, the proxy scheme we give below is just as strong as the underlying cryptosystem.[5]

Related work A natural question to ask is whether there exist atomic proxy functions (and feasible schemes to generate proxy keys) for any public key cryptosystems.

Previous work on delegating the power to decrypt has focused on developing efficient transformations that allow the original recipient to forward *specific ciphertexts* to another recipient. Mambo and Okamoto [MO97] develop this formulation and give efficient transforms (more efficient than decryption and re-encryption) for ElGamal and RSA. Mambo, Usuda and Okamoto [MUO96] apply a similar notion to signature schemes.

While such schemes have value from the standpoint of efficiency, they are not, however, "atomic proxy cryptosystems" by our definition because the transforming function must be kept secret and applied online by the original keyholder on a message-by-message basis (the schemes are not atomic). The security semantics of these systems are essentially the same as a decryption operation followed by a re-encryption operation for the new recipient. Our formulation of proxy cryptography is distinguished from the previous literature by the ability of the keyholder to publish the proxy function and have it applied by untrusted parties without further involvement by the original keyholder.

3.1 Proxy encryption

Although the problem of proxy cryptography seems like a natural extension of public-key cryptography, existing cryptosystems do not lend themselves to obvious proxy functions. RSA [RSA78] with a common modulus is an obvious candidate, but that scheme is known to be insecure [Sim83,DeL84]. Similarly, there

[5] Note that Bob of this example may be a government mandating that Alice provide him with access to her key. It has been argued that such a scheme makes the system as a whole less trustworthy due to the extra engineering effort involved; we argue here that in the case of random-self-reducible cryptosystems such as ElGamal variants, requiring Alice to encrypt her secret key using the government's public key may also weaken the underlying cryptosystem in the precise mathematical sense of spoiling the random-self-reducibility.

do not appear to be obvious proxy functions for many of the previous discrete-log-based cryptosystems. This is not to say, of course, that proxy functions for existing systems do not exist.

We now describe a new secure discrete-log-based public-key cryptosystem that does have a simple proxy function. The scheme is similar in structure to ElGamal encryption [ElG85], but with the parameters used differently and the inverse of the secret used to recover the message.[6] (This approach has merit beyond proxy encryption; [Hug94] proposed a Diffie-Hellman-like key agreement protocol based on the inverse of the secret, which allows a message's sender to determine the key prior to identifying its recipient).

Cryptosystem $\mathcal{X}$ (encryption) Let p be a prime of the form $2q + 1$ for a prime q and let g be a generator in $\mathbb{Z}_p^*$; p and g are global parameters shared by all users. A's secret key a, $0 < a < p - 1$, is selected at random and must be in $\mathbb{Z}_{2q}^*$, *i.e.*, relatively prime to $p - 1$. (A also calculates the inverse $a^{-1} \bmod 2q$). A publishes the public key $g^a \bmod p$. Message encryption requires a unique randomly-selected secret parameter $k \in \mathbb{Z}_{2q}^*$. To encrypt m with A's key, the sender computes and sends two ciphertext values (c_1, c_2):

$$c_1 = mg^k \bmod p$$
$$c_2 = (g^a)^k \bmod p$$

Decryption reverses the process; since

$$c_2^{(a^{-1})} = g^k \pmod p$$

it is easy for A (who knows a^{-1}) to calculate g^k and recover m:

$$m = c_1((c_2^{(a^{-1})})^{-1}) \bmod p$$

The efficiency of this scheme is comparable to standard ElGamal encryption.

Symmetric proxy function for $\mathcal{X}$ Observe that the c_1 ciphertext component produced by Cryptosystem $\mathcal{X}$ is independent of the recipient's public key. Recipient A's key is embedded only in the c_2 exponent; it is sufficient for a proxy function to convert ciphertext for A into ciphertext for B to remove A's key a from c_2 and replace it with B's key b. Part of what a proxy function must do, then, is similar to the first step of the decryption function, raising c_2 to a^{-1} to remove a. The proxy function must also contribute a factor of b to the exponent. Clearly, simply rasing c_2 to a^{-1} and then to b would accomplish this, but obviously such a scheme would not qualify as a secure proxy function; anyone who examines the proxy key learns the secret keys for both A and B.

This problem is avoided, of course, by combining the two steps into one. Hence, the proxy key $\pi_{A \to B}$ is $a^{-1}b$ and the proxy function is simply $c_2^{\pi_{A \to B}}$.

[6] David Wagner notes that this proxy scheme can be extended to work with standard ElGamal encryption.

Note that this is a symmetric proxy function; A and B must trust one another bilaterally. B can learn A's secret (by multiplying the proxy key by b^{-1}), and A can similarly discover B's key. Observe that applying the proxy function is more efficient than decryption and re-encryption, in that only one exponentiation is required.

Security of $\mathcal{X}$ First, we show that $\mathcal{X}$ is secure—that cleartext and secret keys cannot be recovered from ciphertext and public keys. Beyond that, we also show that publishing the proxy key compromises neither messages nor secret keys. Since recovering a secret key enables an adversary to recover a message and since cryptanalysis is easier with more information (i.e., a proxy key), it is sufficient to show that no cleartext is recoverable from ciphertext, public keys, and proxy keys. Specifically, we will show that the problem of recovering m from

$$(g^a, g^b, g^c, \ldots, mg^k, g^{ak}, a^{-1}b, a^{-1}c, \ldots).$$

is at least as hard as Diffie-Hellman.

Theorem 6. *Suppose there exists a randomized algorithm f that with probability $\epsilon > 1/|p|^{O(1)}$ succeeds in recovering m from the public information*

$$(g^a, g^b, \ldots, mg^k, g^{ak}, b/a, \ldots)$$

where the probability is taken over f's random choices as well as over m and the parameters a, b, and k. Then, for each $\eta = 2^{-|p|^{O(1)}}$, there exists a randomized polynomial-time algorithm for Diffie-Hellman that succeeds with probability $1-\eta$.

Proof. The proof is found in [BS98].

Similarly one can show that recovering a from $(g^a, g^b, mg^k, g^{ak}, b/a)$ is as hard as the discrete log, so publishing the proxy key does not compromise a—not even to the level of Diffie-Hellman.

3.2 Proxy identification

In this section we describe a discrete-log-based identification scheme. With p, g, a as before, Alice wishes to convince Charlotte that she controls a; Charlotte will verify using public key g^a. As before, the proxy key $\pi_{A\to B}$ will be a/b—it will be safe to publish a/b and Alice and Charlotte can easily use a/b to transform the protocol so Charlotte is convinced that Alice controls b.

Note that in the case of a secure identification proxy key that transforms identification by A into identification by B, it is B whose secret is required to construct the proxy key because identification as B should not be possible without B's cooperation.

Cryptosystem $\mathcal{Y}$ (identification) Let p and g be a prime and a generator in $\mathbb{Z}_p^*$, respectively. Alice picks random $a \in \mathbb{Z}_{2q}^*$ to be her secret key and publishes g^a as her public key. Each round of the identification protocol is as follows:

- Alice picks a random $k \in \mathbb{Z}_{2q}^*$ and sends Charlotte $s_1 = g^k$.
- Charlotte picks a random bit and sends it to Alice.
- Depending on the bit received, Alice sends Charlotte either $s_2 = k$ or $s_2' = k/a$.
- Depending on the bit, Charlotte checks that $(g^a)^{s_2'} = s_1$ or that $g^{s_2} = g^k$.

This round is repeated as desired. As with existing protocols, there may be ways to perform several rounds in parallel for efficiency [FFS88].

Symmetric proxy function for $\mathcal{Y}$ A symmetric proxy key is a/b. Suppose Charlotte wants to run the protocol with g^b instead of g^a. Either Alice or Charlotte or any intermediary can use the proxy key to convert Alice's responses k/a to k/b.

Security of $\mathcal{Y}$

Theorem 7. *Protocol $\mathcal{Y}$, with or without proxy keys published, is a zero knowledge protocol that convinces the verifier that the prover knows the secret key.*

Proof. The proof is found in [BS98].

3.3 Proxy signature

The concept of proxy cryptography also extends to digital signature schemes. A signature proxy function transforms a message signature so that it will verify with a public key other than that of the original signer. In other words, a signature proxy function $\Pi(s, \pi_{A \to B})$ with proxy key $\pi_{A \to B}$ transforms signature s signed by the secret component of key A such that $V(m, \Pi(S(m, A), \pi_{A \to B}), B)$ returns VALID, where $S(m, k)$ is the signature function for message m by key k and $V(m, s, k)$ is the verify function for message m with signature s by key k.

Again, existing digital signature schemes such as RSA [RSA78], DSA [NIS91], or ElGamal [ElG85], etc. do not have obvious proxy functions (which, again, is not to say that such functions do not exist).

As in the case of proxy identification, in order to construct a proxy key that transforms A's signature into B's signature, B's secret must be required to construct the proxy key because signing for B should not be possible without B's cooperation.

Now we will see how to use the proxy identification scheme to construct a proxy signature scheme. We suppose there exists a hash function h whose exact security requirements will be discussed below. The parameters p, g, a, b are as before.

Cryptosystem $\mathcal{Z}$ (signature) To sign a message m, Alice picks $k_1, k_2, \ldots k_\ell$ at random and computes $g^{k_1}, \ldots g^{k_\ell}$. Next Alice computes $h(g^{k_1}, \ldots g^{k_\ell})$ and extracts ℓ pseudorandom bits $\beta_1, \ldots, \beta_\ell$. For each i, depending on the i'th pseudorandom bit, Alice (who knows a) computes $s_{2,i} = (k_i - m\beta_i)/a$; that is, $s_{2,i} = (k_i - m)/a$ or $s_{2,i} = k_i/a$. The signature consists of two components:

$$s_1 = (g^{k_1}, \ldots, g^{k_\ell})$$
$$s_2 = ((k_1 - m\beta_1)/a, \ldots, (k_\ell - m\beta_\ell)/a)$$

To verify the signature, first the β_i's are recovered using the hash function. The signature is then verified one "round" at a time, where the i'th round is $(g^{k_i}, (k_i - m\beta_i)/a)$. To verify $(g^k, (k - m\beta)/a)$ using public key g^a, the recipient Charlotte raises (g^a) to the power $(k - m\beta)/a$ and checks that it matches $g^k / g^{m\beta}$.

Symmetric proxy function for $\mathcal{Z}$ A symmetric proxy key $\pi_{A \to B}$ for this signature scheme is a/b. The proxy function Π leaves s_1 alone and maps each component $s_{2,i}$ to $s_{2,i}\pi_{A \to B}$.

Security of $\mathcal{Z}$ This scheme relies on the existence of a "hash" function h. Specifically (Hash Assumption), we assume there exists a function h such that:

- On random input (g^a, m), it is difficult to generate $\{r_i\}$ and $\{\beta_i\}$ such that
$$h(g^{ar_1 + m\beta_1}, \ldots, g^{ar_\ell + m\beta_\ell}) = \langle \beta_1 \ldots, \beta_\ell \rangle.$$
- More generally, it is difficult to generate such $\{r_i\}$ and $\{\beta_i\}$ on input g^a, m, and samples of signatures on random messages signed with a.

It is not our intention to conjecture about the existence of such functions h. In particular, we do not know the relationship between the hash assumption and assumptions about collision freedom or hardness to invert.[7] We note that this generic transformation of a protocol to a signature scheme has appeared in the literature [FS87].

We now analyze the hash assumption. Note that in order to produce a legitimate signature on m that verifies with g^a, a signer needs to produce $\langle g^{k_i} \rangle$ and $\langle (k_i - m\beta_i)/a \rangle$. Thus, putting $\langle \beta_i \rangle = h(\langle g^{k_i} \rangle)$ and then $\langle r_i \rangle = \langle (k_i - m\beta_i)/a \rangle$, it is straightforward to see that the signer could actually produce r_i's and β_i's of the stated type in the course of producing the signature.

While we do not address the security of h, we can state that issuing proxy keys does not weaken the system.

Theorem 8. *Suppose h satisfies the hash assumption. Then, for most b, it is also hard to produce $\{r_i\}$ and $\{\beta_i\}$ given additional input $a/b, g^b$, and samples of messages signed with b.*

Proof. The proof is found in [BS98].

[7] The hash assumption *does* imply that, on random input g^a, it is hard to find $\langle r_i \rangle$ making all the β_i's zero, i.e., such that $h(g^{ar_1}, \ldots, g^{ar_\ell}) = 0$.

4 Conclusions

Conceptually, divertibility and proxiability of protocols are both defined in terms of an effectiveness property and one or two security properties. The effectiveness property is basically the same in both cases, namely extensibility as in Definition 3. Our more recent work shows that a proxy key can be naturally incorporated into (and makes sense for) divertibility as well. In the case of divertibility, the security requirement is that Alice and Bob cannot communicate any subliminal message through the warden. In the case of proxiability, the security requirement is that the proxy key releases no more information than what either Alice or Bob would release in the original protocol. A complete unifying framework remains as future work.

We have introduced the notion of perfect and computational protocol divertibility, and have given a sufficiency criterion for the former. All existing diverted protocols we have found in the literature turned out to satisfy this criterion. The first example of a diverted key distribution protocol was given. This is also the first computationally divertible protocol we know of.

Intuitively, atomic proxy cryptography is a fairly natural extension of the basic notion of public-key cryptography. It surely seems plausible, given that there exist cryptosystems that can grant the ability to encrypt without granting the ability to decrypt, that there might also exist cryptosystems that can grant the ability to *re-encrypt* without granting the ability to decrypt. However, it is not at all obvious whether there exist atomic proxy schemes in general.

Indeed, while this paper demonstrates that there do exist efficient and secure public-key encryption and signature schemes with symmetric atomic proxy functions, this observation probably raises more new questions than it answers. In particular, do proxy functions exist for public-key cryptosystems based on problems other than discrete-log? (One possibility is that, for some cryptosystems, proxy functions do exist but it is infeasible to find a proxy key.) More importantly, we have yet to discover a secure *asymmetric* proxy function of any kind; asymmetric proxy functions are probably much more useful in practice, since there are likely many situations where trust is only unidirectional. Are there cryptosystems for which asymmetric proxy functions exist?

5 Acknowledgements

We thank Steve Bellovin, Matthew Franklin, Jack Lacy, Dave Maher, Andrew Odlyzko and David Wagner for helpful discussions and comments.

References

[BS98] Matt Blaze, Martin Strauss. Atomic Proxy Cryptography. AT&T Labs-Research TR98.5.1 <http://www.research.att.com/library/trs>

[Bleu97] Gerrit Bleumer. On Protocol Divertibility. AT&T Labs-Research TR97.34.2 <http://www.research.att.com/library/trs>

[BD91] Mike V. D. Burmester, Yvo Desmedt. All languages in NP have divertible zero-knowledge proofs and arguments under cryptographic assumptions. *Eurocrypt '90* LNCS 473, Springer-Verlag 1991, 1–10.

[CEG88] David Chaum, Jan-Hendrik Evertse, Jeroen van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. *Eurocrypt '87*. LNCS 304, Springer-Verlag 1988, 127–141.

[CPS95] Jan L. Camenisch, Jean-Marc Piveteau, Markus A. Stadler. Blind Signatures Based on the Discrete Logarithm Problem. *Eurocrypt '94*. LNCS 950, Springer-Verlag 1995, 428–432.

[DeL84] John M. DeLaurentis. A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm. *Cryptologia* 8/3 (1984) 253–259.

[DH76] Whitfield Diffie, Martin E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*. 22/6 (1976) 644–654.

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*. 31/4 (1985) 469–472.

[FFS88] Uriel Feige, Amos Fiat, Adi Shamir. Zero-Knowledge Proofs of Identity. *Journal of Cryptology* 1/2 (1988) 77–94.

[FS87] Amos Fiat, Adi Shamir. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. *Crypto '86*. LNCS 263, Springer-Verlag 1987, 186–194.

[GMR89] Shafi Goldwasser, Silvio Micali, Charles Rackoff. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Computing*. 18/1 (1989) 186–207.

[HMP95] Patrick Horster, Markus Michels, Holger Petersen. Meta-Message Recovery and Meta-Blind Signature Schemes Based on the Discrete Logarithm Problem and Their Applications. *Asiacrypt '94*. LNCS 917, Springer-Verlag 1995, 224–237.

[Hug94] Eric Hughes. An encrypted key transmission protocol. *Crypto '94 Rump Session* presentation, August 1994.

[ISS91] Toshija Itoh, Kouichi Sakurai, Hiroki Shizuya. Any Language in IP has a Divertible ZKIP. *AsiaCrypt '91*. Springer-Verlag 1993, 382–396.

[MO97] Masahiro Mambo, Eiji Okamoto. Proxy cryptosystems: delegation of the power to decrypt ciphertexts. *IEICE Trans. Fund. Electronics Communications and Comp Sci.* E80-A/1 (1997) 54–63.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures: delegation of the power to sign messages. *IEICE Trans. Fund. of Electronic Communications and Comp Sci.* E79-A/9 (1996) 1338–1354.

[NIS91] NIST. A proposed federal information processing standard for digital signature standard (DSS). *Draft Tech. Rep. FIPS PUB XXX*, August 1991. Standards Publication (FIPS

[OO90] Tatsuaki Okamoto, Kazuo Ohta. Divertible zero-knowledge interactive proofs and commutative random self-reducibility. *Eurocrypt '89* LNCS 434, Springer-Verlag 1990, 134–149.

[RSA78] Ronald L. Rivest, Adi Shamir, Leonhard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *CACM* 21/2 (1978) 120-126, reprinted: 26/1 (1983) 96–99.

[Sim83] Gustavus J. Simmons. A "Weak" Privacy Protocol Using the RSA Crypto Algorithm. *Cryptologia* 7/2 (1983) 180–182.

[Sim84] Gustavus J. Simmons. The Prisoners' Problem and the Subliminal Channel. *Crypto '83*. Plenum Press, New York 1984, 51–67.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 :	A1	(11) International Publication Number: WO 94/03003
H04L 9/00		(43) International Publication Date: 3 February 1994 (03.02.94)

(21) International Application Number: PCT/US93/04340

(22) International Filing Date: 4 May 1993 (04.05.93)

(30) Priority data:
07/917,598 23 July 1992 (23.07.92) US

(71) Applicant: CREST INDUSTRIES, INC. [US/US]; 201 Frontage Road North, Suite B, Pacific, WA 98047 (US).

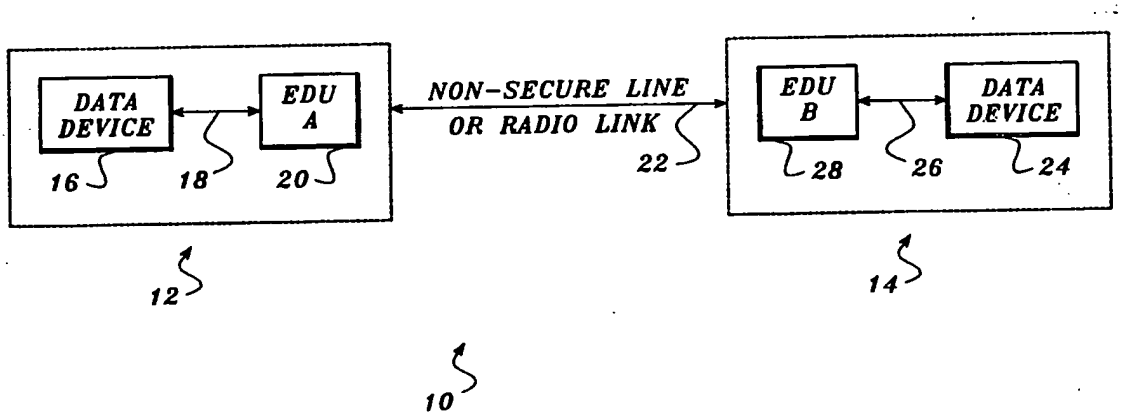
(72) Inventors: RASMUSSEN, Harry, Ronald ; 6105-4th Street Court Northeast, Tacoma, WA 98422 (US). LaBOUNTY, Jack, Daley ; 16931 Southeast 32nd Place, Bellevue, WA 98008 (US). ROSENOW, Michael, James ; 1420 Northwest Gillman, Suite 2305, Issaquah, WA 98027 (US).

(74) Agent: ANDERSON, Ronald, M.; Christensen, O'Connor, Johnson & Kindness, 2800 Pacific First Centre, 1420 Fifth Avenue, Seattle, WA 98101-2347 (US).

(81) Designated States: AT, AU, BB, BG, BR, CA, CH, CZ, DE, DK, ES, FI, GB, HU, JP, KP, KR, KZ, LK, LU, MG, MN, MW, NL, NO, NZ, PL, PT, RO, RU, SD, SE, SK, UA, 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.

(54) Title: ENCRYPTION/DECRYPTION APPARATUS WITH NON-ACCESSIBLE TABLE OF KEYS



(57) Abstract

An encryption/decryption unit (EDU) that handles management of encryption keys used in the secure exchange of data over non-secure communication links. Each EDU includes a central processing unit (CPU) that controls its operation, random access memory (RAM) in which tables of key exchange keys (KEKs) are stored, and a data encryption standard (DES) coprocessor that implements a data encryption algorithm developed by the U.S. National Bureau of Standards - all comprising a module that is embedded in a potting material. Attempts to remove the potting material either by mechanical or solvent means are likely to result in loss of the data and program code stored in the module. The CPU includes special circuitry enabling it to operate in an encrypted mode so that it can not be interrogated to discover the program or data stored therein. This program enables the EDU (20) to establish secure communications with another similar EDU (28) over a non-secure link. Each EDU establishing a secure communications session randomly generates a portion of a session data encryption key (DEK) that is encoded by using a KEK from either a public or private table of keys stored in the embedded RAM. The two EDUs exchange the encrypted portions of the DEK, decrypt the portions, and then logically combine them to determine the current session DEK. Use of a stored EDU ID in each EDU comprising the link prevents a third EDU from bridging the link to tap into the communications between two stations.

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	**FR**	France	**MR**	Mauritania
AU	Australia	**GA**	Gabon	**MW**	Malawi
BB	Barbados	**GB**	United Kingdom	**NE**	Niger
BE	Belgium	**GN**	Guinea	**NL**	Netherlands
BF	Burkina Faso	**GR**	Greece	**NO**	Norway
BG	Bulgaria	**HU**	Hungary	**NZ**	New Zealand
BJ	Benin	**IE**	Ireland	**PL**	Poland
BR	Brazil	**IT**	Italy	**PT**	Portugal
BY	Belarus	**JP**	Japan	**RO**	Romania
CA	Canada	**KP**	Democratic People's Republic of Korea	**RU**	Russian Federation
CF	Central African Republic			**SD**	Sudan
CG	Congo	**KR**	Republic of Korea	**SE**	Sweden
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovak Republic
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LU**	Luxembourg	**TD**	Chad
CS	Czechoslovakia	**LV**	Latvia	**TG**	Togo
CZ	Czech Republic	**MC**	Monaco	**UA**	Ukraine
DE	Germany	**MG**	Madagascar	**US**	United States of America
DK	Denmark	**ML**	Mali	**UZ**	Uzbekistan
ES	Spain	**MN**	Mongolia	**VN**	Viet Nam
FI	Finland				

ENCRYPTION/DECRYPTION APPARATUS WITH NON-ACCESSIBLE TABLE OF KEYS

Field of the Invention

The present invention generally pertains to apparatus for encrypting and decrypting data, and more specifically, to apparatus for implementing the encryption and decryption process with secret encryption keys.

Background of the Invention

Procedures for encrypting and decrypting data for transmission over non-secure radio or telephone links have been highly refined to meet the needs of the military and industry. An encryption algorithm that is virtually unbreakable in any reasonable time frame, by even the most powerful of high-speed computers, has been developed and published by U.S. National Bureau of Standards and sanctioned for use by industry in this country as an acceptable method for protecting computerized data conveyed over non-secure channels. In fact, integrated circuits designed specifically for encryption and decryption of data in accordance with this Data Encryption Algorithm (DEA) are readily available from several vendors, such as Western Digital™. The algorithm, like most encryption schemes, uses an encryption key to encrypt data. Successful use of the DEA, and almost any other encryption/decryption algorithm commonly employed, requires that the station receiving the encrypted transmission have the same key used to encrypt the data in order to decrypt it. Accordingly, no unauthorized party should know or have access to the encryption key that is being used.

Unfortunately, for any prior art encryption/decryption system using the DEA or similar algorithms, extensive security measures are required for managing and

periodically changing the encryption keys that are used. Any third party that gains access to the encryption key being used to encrypt data can tap into a non-secure line over which encrypted messages are transmitted and then use the key to decrypt messages that are intercepted. Even if knowledge of the encryption key used is limited to those operating the encryption/decryption equipment, there can be no assurance that others outside an organization will not breach security and learn the encryption key due to failure of someone in the organization to follow security procedures. As the size of a network over which secure communications must be maintained expands, the difficulty in managing the encryption keys used on the network grows exponentially.

Since any person with access to the encryption keys can breach the security of encrypted communications between members of the network, encryption keys must be changed on a regular basis. Frequent changes in the encryption keys in use minimizes the risk of disclosure by individuals that previously had access to the keys. However, any such change requires that the new encryption keys be distributed to all stations in the network. Typically, the new encryption keys are hand carried to each station site by bonded couriers; nevertheless, it is possible that a courier may compromise security. Even if a security breach does not occur, the cost of regularly distributing encryption keys to each station of a large network in this manner may be prohibitive.

For these reasons, it is preferable to use encryption keys at each station in a network that are not known to anyone, even those operating the encryption/decryption apparatus. Various techniques have been developed to access encryption keys stored in an electronic memory for this purpose. For example, a new encryption key can be selected for subsequent encryption of communications between stations based on the last encryption key that was used, by applying a secret formula to generate the new key. However, if the formula is discovered or otherwise becomes known by someone who is outside the organizational network, security of the encryption system is breached, since that person can generate the encryption keys that will subsequently be used, simply by applying the formula to any previously discovered key.

Clearly, it would be preferable to randomly generate the encryption key that is used to encrypt data transmitted to another station each time that communications are initiated. Yet, random generation of an encryption key at one station inherently renders the receiving station unable to decrypt the message, because it does not have the encryption key used. What is therefore required are means for transmitting the encryption key from one station to another in an encrypted form, with some provision

that enables the receiving station to decrypt the encryption key. Prior art encryption/decryption apparatus do not provide means to accomplish this task in an efficient manner that is not easily circumvented. Any key exchange key (KEK) that is used in the process of transferring an encryption key for encrypting and decrypting the message to the other station must be available to both stations, but can not be available to anyone outside the secure network of stations. Even if the encryption apparatus is available to someone outside the organization, it should be virtually impossible to discover the KEKs used by stations comprising the network, if secure communications are to be maintained.

The foregoing aspects and many of the attendant advantages of this invention over the prior art will become more readily appreciated as the same becomes better understood by reference to the following detailed description, when taken in conjunction with the accompanying drawings.

Summary of the Invention

In accordance with the present invention, encryption/decryption apparatus for ensuring secure communications between two stations include encryption processor means for encrypting and decrypting data using a session data encryption key (DEK) that is input thereto. Control means coupled to the encryption processor means are provided for controlling the operation of the encryption processor means. The control means supply the encryption processor means with the data for encryption and decryption and with an encryption key for use in encrypting and decrypting the data to produce an output signal in response to programmed instructions. These programmed instructions cause the control means to automatically randomly select a part of a session DEK and to combine it with another part of the session DEK received from the other station to determine the session DEK that will be used by the encryption processor means to encrypt data. Non-volatile memory means that are coupled to the control means store a plurality of key encryption keys that are used by the encryption processor means in encrypting a part of the session DEK for transmission to the other station. The control means select the key encryption key from the plurality of key encryption keys as a function of a check value determined with the part of the session key.

Within the non-volatile memory means is disposed an internal power source that provides electrical power to maintain storage of the plurality of key encryption keys. Potting means encapsulate the encryption processor means, the control means, and the non-volatile memory means in a radio and light wave opaque material that is sufficiently hard and resistant to dissolution by solvents to prevent its removal without

damage to interconnections coupling the non-volatile memory means to the control means and damage to interconnections supplying electrical power to the non-volatile memory means from the internal power source. Such damage causes erasure of the plurality of key encryption keys stored in the non-volatile memory means. In addition, the control means respond to any attempt to externally interrogate the non-volatile memory means by causing erasure of the key encryption keys stored therein.

Multiplexer means are coupled to the control means to receive a data signal and a select signal therefrom, and are also coupled to the encryption processor means, an output port, and the memory means; the multiplexer means selectively convey the data signal to one of the encryption processor means, the output port, and the memory means, in response to the select signal. The control means include a non-volatile memory for retaining program steps and a unique identification code that identifies a specific encryption/decryption apparatus. In addition, the control means include means for locking the control means and its non-volatile memory to prevent data and program steps from being read externally after storage of the program steps in the non-volatile memory is complete. The means for locking include means for encrypting data and memory addresses defining memory storage locations within the non-volatile memory of the control means and within the non-volatile memory means.

Brief Description of the Drawings

FIGURE 1 is a block diagram of a communications network comprising two stations, each provided with an encryption/decryption unit (EDU) in accordance with the present invention, thereby enabling the stations to establish secure communications over a non-secure line or radio link;

FIGURE 2 is a schematic block diagram of one of the EDUs shown in FIGURE 1;

FIGURE 3 is a flow chart illustrating the logical steps implemented at one station by the EDU in selecting and encrypting a first portion of a session encryption key for transmittal to another station;

FIGURE 4 is a flow chart illustrating the logical steps implemented by the EDU at the other station in decrypting the first portion of the session encryption key, and in selecting and encrypting a second portion of the session encryption key for transmittal to the one station; and

FIGURE 5 is a flow chart illustrating the logical steps implemented by the EDU at the one station to decrypt the second portion of the session encryption key.

Detailed Description of the Preferred Embodiment

As noted above, one of the more difficult problems in establishing and maintaining an encrypted communication network is distributing secure DEKs to each station in the network on a regular basis. In FIGURE 1, a simple network for carrying out encrypted communications is shown generally at reference numeral 10. Network 10 is shown simply as two stations, including a station 12 and a station 14, but it will be appreciated that the network can comprise many other such stations.

Both stations 12 and 14 use similar components for encrypting and decrypting communications. For example, station 12 includes a data device 16, which may, for example, comprise a facsimile machine or personal computer (neither shown separately). Data device 16 is connected through lines 18 to an EDU A 20. Station 12 uses EDU A 20 to establish secure communications over a non-secure line (or radio link) 22 with station 14, which includes an EDU B 28. EDU B 28 is connected to a data device 24 over lines 26. Data device 24 is the same type of device as data device 16. Thus, if data devices 16 and 24 are facsimile machines, communications network 10 permits secure communication of facsimile information in an encrypted form between stations 12 and 14 over non-secure line 22.

Because of the manner in which secure communications are established between EDU A 20 and EDU B 28, tapping into non-secure line 22 using a similar EDU (not shown) would NOT enable a third party to breach secure communications between stations 12 and 14. In the preferred form of the present invention, communications between EDU A 20 and EDU B 28 are carried out using a session encryption key that is changed with each session and comprises two parts, one part randomly selected by EDU A 20, and the other part randomly selected by EDU B 28. Thus, the present invention comprises the EDU at each of the communicating stations 12 and 14. In establishing secure communications between two stations 12 and 14, the EDU at each station randomly select its respective portion of the session encryption key, encrypts that portion of the session encryption key, and transmits the encrypted respective portion of the session encryption key to the other station. Once both station 12 and station 14 have decrypted the portion of the session encryption key developed by the other station, the two portions are logically combined at each station to produce the complete or final session encryption key used for encrypting data transmitted between stations 12 and 14 during the current session. In addition, the EDUs are preprogrammed to ensure that the intended station in a two-way communication link is actually receiving or transmitting the encrypted data, to guard against a third party tapping into non-secure line 22 with another EDU. The EDUs

also ensure that the two portions of the session encryption key that are exchanged between stations 12 and 14 are correctly received and decrypted, thereby protecting against data errors that might have arisen in the transmission of the encrypted portions of the session encryption key between the two stations or in their decryption.

A block diagram of EDU 20 is shown in FIGURE 2; EDU 28 is exactly the same, except for having a different EDU identification number stored within it. EDU 20 includes a potted module 30 and an external input/output (I/O) bus 32 for providing interconnections between the EDU and the data device (or to other components, if the EDU is used as an element of a more extensive data encryption apparatus) that will provide the data to be encrypted or will receive the data that is decrypted by the EDU. Module 30, which comprises virtually the entire EDU, is encapsulated within a radio opaque and light opaque potting compound 34 to prevent discovery of the internal circuitry and to prevent forced electromagnetic or visual tapping, monitoring, or other forms of penetration that might be attempted to uncover encryption keys and other information included therein. The potting compound is sufficiently hard and resistant to abrasion to prevent its removal without damaging the components comprising the EDU or at least causing loss of important data stored therein. Of greatest sensitivity to maintaining the security of communications between EDUs comprising a network is the need to protect against discovery of KEKs that are encrypted using a key that is unique to each EDU and is assigned to it when it is initialized. The encrypted KEKs are stored as tables within each EDU and are utilized for encrypting portions of the session encryption key that are exchanged between two stations and subsequently logically combined at each EDU to produce a session DEK that is used for encryption of data exchanged over non-secure line 22. To avoid breaching the security of communications on network 10, it is absolutely imperative that these KEKs not become publicly known.

In the preferred form of module 30, two sets or tables of KEKs are stored in encrypted form in a random access memory (RAM) 42. One set is called a "public" set, since each EDU that will be sold will include this set. The other set is a "private" set of KEKs, which optionally may be randomly generated by a user for distribution to and storage in those EDUs comprising a private network of stations. The significance of the KEKs will be apparent from the description that follows. Any attempt to expose the internal circuitry of module 30 by use of a chemical, solvent, or mechanical means in order to access RAM 42 electronically or physically so as to access these data will cause loss of the KEKs that are stored therein. RAM 42 preferably comprises a Dallas Semiconductor™ type DS 1213 smart socket in which is installed

a memory integrated circuit (not separately shown) comprising 128K x 8 bits of storage, i.e., yielding 1,048,576 bits of non-volatile static RAM organized as 131,072 words by 8 bits. This memory integrated circuit is a dual in-line package (DIP) package configuration of generally conventional design, but the smart socket contains an internal battery supply (not separately shown) sufficient to maintain data integrity in the absence of externally applied power for a period in excess of 10 years. Dallas Semiconductor also supplies an integrated circuit non-volatile memory device that includes an integral internal battery supply, and this type of device can be used in place of the smart socket and more conventional memory device combinations. In the event a chemical solution is used to dissolve potting compound 34 in an attempt to discover the KEKs stored in RAM 142, the material comprising RAM 142 (smart socket or memory device that includes the integral internal battery supply) will also be dissolved, thereby disconnecting the internal battery supply and erasing the KEKs stored therein.

Operation of module 30 to establish and conduct secure communications is controlled by a CPU 36, which includes 32K of embedded RAM (not separately shown). In the preferred embodiment, a Dallas Semiconductor™ type DS 5000 microchip integrated circuit is used for CPU 36. The DS 5000 integrated circuit includes non-volatile embedded RAM (not separately shown) and all information and programming stored therein are preserved in the absence of an externally applied voltage for up to 10 years. In addition, the internal data registers and key configuration registers of the DS 5000 integrated circuit are non-volatile. Data stored within the embedded RAM that comprise program steps carried out by CPU 36 in establishing secure communications can be modified after encapsulation of module 30 has been accomplished with potting material 34; however, initial loading of the embedded RAM within the DS 5000 microchip comprising CPU 36 is accomplished with a conventional universal asynchronous receiver/transmitter (UART) interface (not shown) that is connected through external I/O bus 32 by lines 76. In addition, control lines 50 connect CPU 36 to external I/O bus 32 and convey write, read, interrupt, and signals for ports 0-3 (P1.0-P1.3) of the CPU.

Data lines (D0-D7) 54 interconnect CPU 36 with RAM 42 and with a buffer 46. Buffer 46 comprises an SN 74HCT245 octal bus transceiver with a three-state output that is used to block external access to internal data transfers occurring within module 30, thereby preventing an external device from accessing KEKs stored in RAM 42 and other data transferred between components of the module. Buffer 46 is enabled via control signals supplied over a line 74 by CPU 36 when it is appropriate

to allow bi-directional data transfer to and from external I/O bus 32 through lines 52, and therefore to and from an external device.

To provide additional security, CPU 36 operates in an encrypted mode. The encrypted mode is deactivated prior to the initial loading of program steps and data into the embedded RAM of CPU 36. Before the initial loading of program code and data begins during manufacture of the EDU, a 40-bit encryption mode key is selected for use by CPU 36 in the encrypted mode. A data encryptor circuit and an address encryptor circuit (neither separately shown) within CPU 36 respectively control the form in which the program code is stored in the embedded RAM of the CPU and the addresses at which it is stored. As the initial loading of program steps is performed, the data encryptor circuit uses the 40-bit encryption mode key to transform or encrypt opcodes, operands, and data bytes at each memory location defined by the software. Similarly, the address encryptor circuit uses the encryption mode key in a different encryption algorithm to translate or encrypt a logical address of each data byte location into an encrypted address at which the data are actually stored. The contents of the embedded RAM are then verified, and the encrypted mode is enabled by setting a security lock bit. After the security lock bit is set to enable operation in the encrypted mode, the contents of the CPU's embedded RAM is unintelligible to an observer that might attempt to tap into its circuitry to discover the program code and other data stored therein. The address and data encryptor circuitry provides real time translation or decryption of program code and address locations to CPU 36 during subsequent operation of the EDU. Only program code and data stored in the CPU's embedded RAM that does NOT affect secure operation of the EDU can be changed after the security lock bit is set. Any attempt to externally interrogate the CPU to discover the 40-bit encryption key causes its erasure, rendering the contents of the embedded RAM useless. Even if the encrypted program code and data are thereafter read back from the embedded RAM in CPU 36, they can not be decrypted without the 40-bit encryption mode key, which is lost.

CPU 36 selects a specific storage location for a KEK within RAM 42 by setting 16 address bits. Lines 58 connect CPU 36 to a latch 44, and lines 60 connect latch 44 to RAM 42. To minimize the total number of pins required on CPU 36, the first eight address bits (A0-A7) and eight bits of data (D0-D7) use the same pins on CPU 36. These address bits and data are alternatively passed between CPU 36, latch 44, and RAM 42 over lines 58 and 60, respectively. The eight most significant bits of the address are conveyed on lines 56b directly from CPU 36 to RAM 42 and to external I/O bus 32. The least significant eight address bits (A0-A7) are carried on

lines 56a. In the preferred embodiment, the 16 address bits are available on lines 56 at external I/O bus 32 to address the embedded RAM in CPU 36 when it is initially loaded or subsequently modified.

Although CPU 36 controls the operation of module 30, the actual encryption and decryption of data is implemented by a data encryption standard (DES) coprocessor 38. DES coprocessor 38 is designed to encrypt and decrypt 64-bit blocks of data using the algorithm specified in the Federal Information Processing Data Encryption Standard (No. 46). A transfer rate of 807 kilobytes per second is implemented by DES coprocessor 38 under the control of a 10 megahertz clock circuit 48, to which the DES coprocessor is connected through lines 70. Data are transferred between CPU 36 and DES coprocessor 38 over lines 72. In the preferred embodiment, a Western Digital™ type DES WD20C03A integrated circuit is used for DES coprocessor 38, although other such devices are available from other suppliers. A decoder/multiplexer (MUX) 40 is connected through lines 68 to DES coprocessor 38 and through lines 66 to CPU 36. Decoder/MUX 40 is a three-line to eight-line circuit that decodes one of eight lines, dependent upon three binary select inputs and three enable inputs. Lines 66 carry the three binary select signals and the output signal from decoder/MUX 40 and line 68 carries selectable input 7. In addition, lines 62 carry selectable inputs 5 and 6 from RAM 42, while lines 64, which extend between decoder/MUX 40 and external I/O bus 32 convey selectable inputs 0-4.

The embedded non-volatile RAM in CPU 36 is loaded with the appropriate program steps for controlling the operation of EDU 20 at the time module 30 is manufactured. In addition, RAM 42 is loaded with a set of 65,535 public KEKs that are randomly generated from over 72 quadrillion possibilities. Each EDU that is thus produced stores the same table of 65,535 randomly generated public encryption keys. Any EDU can establish secure encrypted communications with any other EDU using the public KEKs. Also stored in RAM 42 is a user-generated table of over 65,535 randomly generated private encryption keys. These private KEKs are used for initiating secure communications with another EDU in the private network that has the same table of private KEKs stored within its RAM 42.

The steps involved in establishing secure communications between two EDUs are shown in FIGURES 3, 4, and 5. Not shown are any handshaking steps necessary to connect two EDUs in communication with each other so that data for a specific device can be transmitted between them. Preferably, such handshaking steps are

implemented by transmitting predefined data blocks between the two devices, but do not necessarily require action by either EDU.

In FIGURE 3, a flow chart 100 identifies the steps taken by EDU B 28, acting as the intended recipient, to establish secure communications. It will be apparent the steps in flow chart 100 could also be carried out by EDU A 20; however, the choice was made in the preferred embodiment to have the receiving station start the process of determining a session data encryption key, thereby avoiding the possibility that a third party posing as another station might tap into the unsecured line with an EDU to initiate the secure communications link. The method begins with a start block 102. In a block 104, EDU B 28 generates a 64-bit random data encryption key 1 (DEK1), which is one of over 72 quadrillion possible data encryption keys (i.e., all possible combinations of 56 bits).

The DEK1 is the first portion of a session data encryption key that will be subsequently used for transmitting encrypted data between the two EDUs. In a block 106, EDU B 28 then uses the DEK1 as the encryption key in implementing the DEA to encrypt one block of data. The use of the DEA to encrypt a single block of data is referred to as an electronic code book (ECB) method and is carried out by DES coprocessor 38 under the control of CPU 36. The ECB method employs the key (DEK1) to encrypt a 64-bit zero function, i.e., a function comprising 64 logical zeros, the result being used to determine a check value.

In a block 108, a KEK table entry value KEK1 comprising the 16 least significant bits (LSBs) of the 64-bit check value from block 106 is determined. The EDU uses the public or private table for KEKs, as specified by EDU A 20 during the handshaking that preceded establishing the secure communications link. The public table and private table of KEKs each represent a linear array of data, that can be taken in groups of four 16-bit words or 64-bits at a time, to define a KEK. The 16 LSBs of the check value determine the starting point or table entry value in the selected table to determine the 64 bits used as a KEK, as indicated in a block 110. Using the 64-bit KEK selected from the table as the encryption key, the EDU encrypts the value DEK1 using the ECB method in a block 112. A cyclic redundancy check (CRC) value for the KEK table that was selected is then determined in the conventional manner.

In a block 114, the EDU encrypts the KEK table CRC, its own EDU ID number (which is stored in within module 30 and is not user modifiable), and the KEK1 entry value using a predefined header encryption key and the ECB method to produce an encrypted key header. The header encryption key is stored in the embedded RAM within CPU 36 at the time that its programming is initially loaded

and is the same for each EDU. In a block 116, the EDU transmits the encrypted key header and encrypted DEK1 to EDU A 20, which initiated the communication. Although both parts of this transmission are encrypted, they are encrypted at different levels of security, since the encrypted key header is always sent encrypted with the same predefined (although inaccessible) key and the encrypted DEK1 uses a different key with virtually every communication session between two EDUs. The method for establishing secure communications continues with the other EDU, at a block B1 118.

In FIGURE 4, a flow chart 120 shows the steps carried out by EDU A 20 (the EDU that initiated the communication). Flow chart 120 begins at block B1 118 and proceeds to a block 122 wherein the encrypted key header and encrypted DEK1 received from EDU B 28 are parsed. In a block 124, the encrypted key header is decrypted using the predefined header encryption key with the ECB method, enabling the EDU to determine the KEK table CRC, the encoded EDU ID number of the EDU that transmitted the encrypted header, and KEK1.

A decision block 126 causes the CPU to determine if the KEK table CRC is correct, thereby ensuring that the KEK table used to encrypted the header is the same as the KEK table that will be used by EDU A 20. This step prevents two EDUs from attempting to communicate if they are using different private KEK tables or if the public table in used by one has become corrupted or is different than the normal public table of KEKs for some other reason. If the CRC value does not match the expected value, a block 128 stops communication between the EDUs. Under most circumstances, however, the KEK table CRC is correct and the logic proceeds to a block 130.

In block 130, EDU A 20 determines the 64-bit KEK that was previously selected from the public or private table by EDU B 28, using the KEK1 value that it just received as an offset to enter the table. The 64-bit KEK is then used with the ECB method to decrypt the value DEK1, as shown in a block 132.

In a block 134, a validity check is made to ensure that the decryption process was carried out correctly and that the encrypted data were not affected by noise or other problems during transmission. The validity check is carried out by using the decrypted DEK1 value and the ECB method to encrypt the 64-bit zero function. The result provides a check value, the 16 LSBs of which are a value KEK1'. The accuracy of the encryption/decryption process and transmission is confirmed in a decision block 136 if the EDU determines that KEK 1 equals KEK 1'. If not, a block 138 provides for indicating that an error has occurred in establishing secure communications, which leads to a stop block 140.

On the other hand, assuming that KEK 1 equals KEK 1', a block 142 directs the EDU to generate a 64-bit random value, DEK2, which is the second portion of the session data encryption key that will be used to encrypt data transmissions between the two EDUs. In a block 144, EDU A 20 performs a logical XOR to combine the first portion of the session key, DEK1, and the second portion, DEK2, to determine the final session data encryption key DEK.

In a block 146, DEK2 is used with the ECB method to encrypt the 64-bit zero function in order to determine a second check value. Using the 16 LSBs of the check value in a block 148, the EDU determines a table entry value KEK2. By entering the specified public or private table at the address offset determined by KEK2, four consecutive 16-bit words comprising a 64-bit KEK are determined in a block 150. The EDU uses the value of KEK from the table and the ECB method to encrypt DEK2 in a block 152.

With the predefined header encryption key, the EDU A 20 encrypts the KEK table CRC, its own EDU ID, and the table entry value KEK2, producing an encrypted key header in a block 154. The encrypted key header just produced and the encrypted DEK2 will be transmitted to EDU B 28 only if the next test is passed in a decision block 155.

Decision block 155 now determines whether the EDU ID that was decrypted from the header received from EDU B 28 in block 124 matches that of the EDU that was initially called, i.e., confirms that the intended recipient has responded. Since the encryption of the EDU ID is carried out automatically by EDU B 28, and can not be modified or affected by external signals, it is virtually impossible for a third party to use another EDU to break into a communications link and take part in establishing secure communications, since the encrypted EDU ID that is returned to the station that initiated the communication would then not match the expected EDU ID. A negative response to decision block 155 causes the process for establishing secure communications to be halted at a stop block 157. Otherwise, the process for establishing a secure communications link proceeds to a block 156. Block 156 provides for transmitting the encrypted key header and encrypted DEK2 to the other EDU, i.e., to EDU B 28, which is the intended recipient for subsequent encrypted communications. Thereafter, the logic proceeds to a block A2 158 in FIGURE 5.

FIGURE 5 illustrates a flow chart 160 defining the steps next implemented by EDU B 28. Following block 158, a block 162 provides for parsing the encrypted key header and encrypted DEK2. The encrypted key header is then decrypted in a block 164 using the ECB method in connection with the predefined header encryption

key, enabling EDU B 28 to determine the KEK table CRC, the EDU ID of the transmitting station, and the KEK2 table entry value. In a decision block 166, EDU B 28 determines if the KEK table CRC value is correct, i.e., confirms that the public or private table of KEKs used by EDU A 20 is the same as that being used by EDU B 28. If not, the communication process is halted at a block 168. Otherwise, the process continues with a block 170.

Block 170 provides for selecting a 64-bit KEK from the designated table of KEKs using the entry value KEK2 as an offset. In a block 172, the EDU uses the selected KEK value in connection with the ECB method to decrypt the encrypted DEK2. It then performs a validity check in a block 174, by using the DEK2 value in connection with the ECB method to encrypt the 64-bit zero function, thereby determining a check value and a table entry value KEK 2' that is based upon the 16 LSBs of the check value. A decision block 176 causes CPU 36 to determine if the decrypted KEK2 equals KEK2' that was just determined in block 174. If not, a block 178 provides for indicating that an error has occurred, leading to a stop block 180.

However, assuming that the validity check has a positive response, in a block 182, the EDU logically XORs DEK1 and DEK2 to determine the value of DEK for this session. At this point, both the receiving and transmitting station EDUs have established the current session data encryption key DEK. Before the communication session can proceed, one final check is made in a decision block 183.

Decision block 183 determines if the EDU ID sent by EDU A 20 in the key header that was decrypted in block 164 by EDU B 28 matches an expected EDU ID. If not, block 180 stops the process of establishing secure communications between the two EDUs. Decision block 183 thus determines if a third EDU has been used to intercept communications between EDU A 20 and EDU B 28; if not, the communication of encrypted data proceeds at a block 184.

The session DEK is used in a block 184 by EDU A 20 to encrypt data (such as facsimile or computer data) for transmission to EDU B 28, which then decrypts it using the same DEK. When EDU B 28 determines that the last of the data to be transmitted has been received and decrypted, a block 186 provides for resetting both EDUs to await the next communication. Thereafter, a stop block 188 terminates further communication between the two stations.

During the process of establishing secure communications, neither of the EDUs linking together transmits DEK1 or DEK2 in the clear. Either the public or private table of KEKs is used for encrypting the first and second portions of the

current session DEK. Consequently, only another EDU provided with the same control program and the same table of KEKs (producing the same CRC value) would be able to decrypt either the encrypted first or second portions of the DEK. Since the software program controlling the operation of the EDUs requires that the EDU ID number of the stations be encrypted as part of the key header information that is exchanged, a third EDU cannot be used to surreptitiously substitute for the intended receiving station or transmitting station during the establishment of the secure communication link. Consequently, only the two EDUs at the receiving and transmitting stations comprising a link are able to communicate to establish a session DEK and thereafter carry on secure communications.

Only an EDU having the same session DEK used to encrypt data can decrypt the data. Furthermore, although any EDU can establish secure communications with any other EDU using the public table of KEKs, only EDUs having the same private table of KEKs (determined from the KEK table CRC value) can establish a session DEK to communicate with each other. As a result, a corporation that generates its own table of private KEKs can ensure that secure communications are initiated only with other stations comprising its private network that include the same table of private KEKs.

While the DES algorithm is used in the preferred form of the present invention, it will be appreciated that other encryption algorithms that use an encryption key can also be employed. Further, when determining a check value, a predefined function other than the zero function can be used. It should also be apparent that the encrypted key header need not include the EDU ID, if a lower level of security is acceptable, for example, in a local network of EDUs exclusively using private KEKs. These and other modifications to the present invention will be apparent to those of ordinary skill in the art. Accordingly, it is not intended that the invention be in any way limited by the description of the preferred embodiment and modifications thereto, but instead that the scope of the invention be determined entirely by reference to the claims that follow.

The embodiments of the invention in which an exclusive property or privilege is claimed are defined as follows:

1. Encryption/decryption apparatus for ensuring secure communications between two stations, said encryption/decryption apparatus disposed at each station comprising:

(a) encryption processor means for encrypting and decrypting data using an encryption key that is input thereto;

(b) control means, coupled to the encryption processor means, for controlling the operation of the encryption processor means, said control means supplying the encryption processor means with data for encryption and decryption and with the encryption key for use in encrypting and decrypting the data to produce an output signal in response to programmed instructions that cause it to automatically randomly select a part of a session data encryption key for use by the encryption processor means to encrypt data when combined with another part of the session data encryption key received from the other station; and

(c) non-volatile memory means, coupled to the control means, for storing a plurality of key encryption keys used by the encryption processor means in encrypting the part of the session data encryption key for transmission to the other station, said control means selecting the key encryption key from said plurality of key encryption keys as a function of a check value determined by the control means with the part of the session key.

2. The encryption/decryption apparatus of Claim 1, wherein said non-volatile memory means include an internal power source that supplies electrical power to maintain storage of the plurality of key encryption keys.

3. The encryption/decryption apparatus of Claim 1, further comprising potting means for encapsulating the encryption processor means, the control means, and the non-volatile memory means in a radio and light wave opaque material, said potting means being sufficiently hard and resistant to dissolution by solvents to prevent its removal without causing damage to interconnections coupling the non-volatile memory means to the control means and damage to interconnections supplying electrical power to the non-volatile memory means from the internal power source, such damage causing erasure of the plurality of key encryption keys stored in

the non-volatile memory means, said control means also responding to any attempt to externally interrogate the non-volatile memory means by causing erasure of the key encryption keys stored therein.

4. The encryption/decryption apparatus of Claim 1, further comprising multiplexer means, coupled to the control means to receive a select signal therefrom, and coupled to the encryption processor means, an output port, and the memory means, for selectively conveying a data signal thereto in response to the select signal.

5. The encryption/decryption apparatus of Claim 1, wherein the control means include a non-volatile memory for storing the programmed instructions and for storing a unique identification code that identifies a specific encryption/decryption apparatus.

6. The encryption/decryption apparatus of Claim 5, wherein the control means include means for locking the control means and its non-volatile memory to prevent data and program steps from being read externally or changed after storage of the programmed instructions in said non-volatile memory is complete.

7. The encryption/decryption apparatus of Claim 6, wherein the means for locking include means for encrypting data and memory addresses defining memory storage locations within the non-volatile memory of the control means and within the non-volatile memory means.

8. Encryption/decryption apparatus for ensuring secure communications, comprising:

(a) processor means for randomly selecting a partial session data encryption key;

(b) encryption means for encrypting the partial session data encryption key, producing an encrypted part key and decrypting another partial session data encryption key selected at another location; and

(c) means for conveying the encrypted part key to an output port so that it can be transmitted to the other location and for conveying an encrypted signal from an input port, said encryption means decrypting the other partial session data encryption key received from the other location as the encrypted signal, said

processor means combining the partial session data encryption key to determine the current session data encryption key that is subsequently used by it to encrypt data transmitted to the other location and to decrypt encrypted signals received from the other location.

9. The encryption/decryption apparatus of Claim 8, further comprising memory means for storing a plurality of key encryption keys, wherein the encryption means select a specific key encryption key from the plurality of key encryption keys as a function of a check value, said encryption means encrypting a predefined set of characters with said part of the encryption key to determine the check value.

10. The encryption/decryption apparatus of Claim 9, wherein the means for transmitting also transmit the check value determined by the encryption means.

11. The encryption/decryption apparatus of Claim 10, wherein the decryption means use a check value received from said other location to determine a specific key encryption key that was used to encrypt the other partial session data encryption key.

12. The encryption/decryption apparatus of Claim 11, wherein:

(a) the encryption means use the other partial session data encryption key decrypted by the decryption means to encrypt the predefined set of characters, producing a test value;

(b) said processor means compare the test value with a check value received from the other location and detect an error if the test value differs from said check value received from the other location; and

(c) if an error is detected in (b), said processor means halt communications with said other location.

13. The encryption/decryption apparatus of Claim 9, wherein said memory means store a unique identification code for that apparatus.

14. The encryption/decryption apparatus of Claim 9, wherein the memory means comprise a non-volatile memory circuit including an internal power source, said internal power source supplying electrical current to the non-volatile memory circuit to retain data stored therein, said memory means being encapsulated in a material that precludes physical inspection of the memory circuit, preventing discovery of the data stored therein, further comprising means for interrupting electrical current supplied from the internal power source to the memory circuit so that the data stored therein are erased if the material encapsulating the memory means is removed therefrom.

15. The encryption/decryption apparatus of Claim 8, wherein the processor means comprise a central processing unit that is programmed to control the encryption means and the decryption means according to a predefined set of instructions.

16. The encryption/decryption apparatus of Claim 8, wherein the encryption means comprise an integrated circuit that implements encryption and decryption of data from a plurality of sources in response to signals from the processor means, using the current session data encryption key, in accordance with a predefined encryption algorithm and a corresponding predefined decryption algorithm.

17. The encryption/decryption apparatus of Claim 9, wherein the memory means store a plurality of sets of key exchange keys, further comprising means for selecting one of the sets of key exchange keys from which the specific key exchange key is determined.

18. Encryption/decryption apparatus for ensuring secure communications, comprising:

(a) a sealed circuit encapsulated in a material opaque to radio and light waves, said sealed circuit comprising:

(i) a central processing unit that receives and transmits data in both an encrypted and decrypted form;

(ii) a memory circuit coupled to the central processing unit, at least one predefined set of key exchange keys being stored in the memory circuit,

said key exchange keys stored in the memory circuit being externally inaccessible, both physically by inspection and by downloading through the central processing unit;

(iii) an encryption/decryption coprocessor coupled to the central processing unit to receive data therefrom, said encryption/decryption coprocessor encrypting and decrypting the data under control of the central processing unit based upon a specified encryption key, the encryption/decryption coprocessor selectively generating a second set of key exchange keys that are also stored in the memory circuit;

(b) connector means for interconnecting the sealed circuit with external data input and output lines, the encryption/decryption coprocessor selectively encrypting the second set of key exchange keys and the connector means conveying the second set of key exchange keys in an encrypted form to an external device for distribution to other encryption/decryption apparatus comprising a limited network, whereby only encryption/decryption apparatus comprising the limited network can securely communicate with each other using the second set of key exchange keys, but can securely communicate with other like encryption/decryption apparatus that do not comprise the limited network using the predefined set of key exchange keys.

19. The encryption/decryption apparatus of Claim 18, further comprising memory means coupled to the central processing unit, for storing program steps controlling automatic determination of a session data encryption key for use in encrypting and decrypting data, said session data encryption key being determined in part by the central processing unit logically combining a first randomly selected portion of the session data encryption key that is received in an encrypted form from another location with a second randomly selected portion of the session data encryption key that the central processing unit transmits to the other location in an encrypted form.

20. The encryption/decryption apparatus of Claim 19, wherein one of the predefined set and the second set of key exchange keys is selectively used for encrypting said other portion of the session data encryption key.

21. The encryption/decryption apparatus of Claim 18, wherein the memory circuit stores a unique identification code for the sealed circuit that can not be changed, said central processing unit halting operation of the sealed circuit if data are received from the other location that specify a different identification code, thereby

preventing secure communications with an unintended encryption/decryption apparatus.

22. Encryption/decryption apparatus for ensuring secure communications between two stations, comprising:

(a) first processor means at one of the stations for randomly selecting a first part encryption key and second processor means at the other of the two stations for randomly selecting a second part encryption key;

(b) encryption means at said one station for encrypting the first part encryption key, producing an encrypted first part key;

(c) means for transmitting the encrypted first part key to said other station;

(d) decryption means at said other station for decrypting the encrypted first part key to determine the first part encryption key;

(e) encryption means at said other station for encrypting the second part encryption key, producing an encrypted second part key;

(f) means for transmitting the encrypted second part key to said one station; and

(g) decryption means at said one station for decrypting the encrypted second part key to determine the second part encryption key, said first processor means at said one station and said second processor means at said other station then combining the first part encryption key and the second part encryption key to determine an encryption key that is used to encrypt and decrypt subsequent communications between the two stations.

23. The encryption/decryption apparatus of Claim 21, further comprising memory means for storing a plurality of key encryption keys at each of the two stations, wherein the encryption means at each station select a specific key encryption key from the plurality of key encryption keys as a function of a first check value and a second check value, respectively, said encryption means at said one station encrypting a predefined set of characters with said first part encryption key to determine the first check value, and said encryption means at said other station encrypting the predefined set of characters with said second part encryption key to determine said second check value.

24. The encryption/decryption apparatus of Claim 22, wherein the means for transmitting from each station also transmit the respective first or second check value determined by the encryption means at each station.

25. The encryption/decryption apparatus of Claim 23, wherein the decryption means at said other station use the first check value received from said one station to determine the specific key encryption key that was used by the encryption means at said one station to encrypt the first part encryption key, and wherein the decryption means at said one station use the second check value received from said other station to determine the specific key encryption key that was used by the encryption means at said other station to encrypt the second part encryption key.

26. The encryption/decryption apparatus of Claim 24, wherein:

(a) the encryption means at said other station uses the first encryption key decrypted by the decryption means to encrypt the predefined set of characters, producing a test check value;

(b) said second processor means compares the test check value with the first check value and detects an error if the test check value differs from the first check value;

(c) the encryption means at said one station uses the second encryption key decrypted by the decryption means to encrypt the predefined set of characters, producing a test check value;

(d) said first processor means at said one station compares the test check value with the second check value and detects an error if the test check value differs from the second check value; and

(e) if an error is detected in (b), said second processor means halt communications with said one station, and if an error is detected in (d), said first processor means halt communications with said other station.

27. The encryption/decryption apparatus of Claim 22, wherein said memory means at each station store a unique identification code for that station.

28. The encryption/decryption apparatus of Claim 26, wherein the encryption means at said one station encrypt the unique identification code of said other station, the means for transmitting then transmitting an encrypted identification code to said other station, said decryption means at said other station decrypting the unique identification code.

29. The encryption/decryption apparatus of Claim 27, wherein said second processor means compare the decrypted unique identification code with the unique identification code stored in the memory means and if not identical, halt communications with said one station.

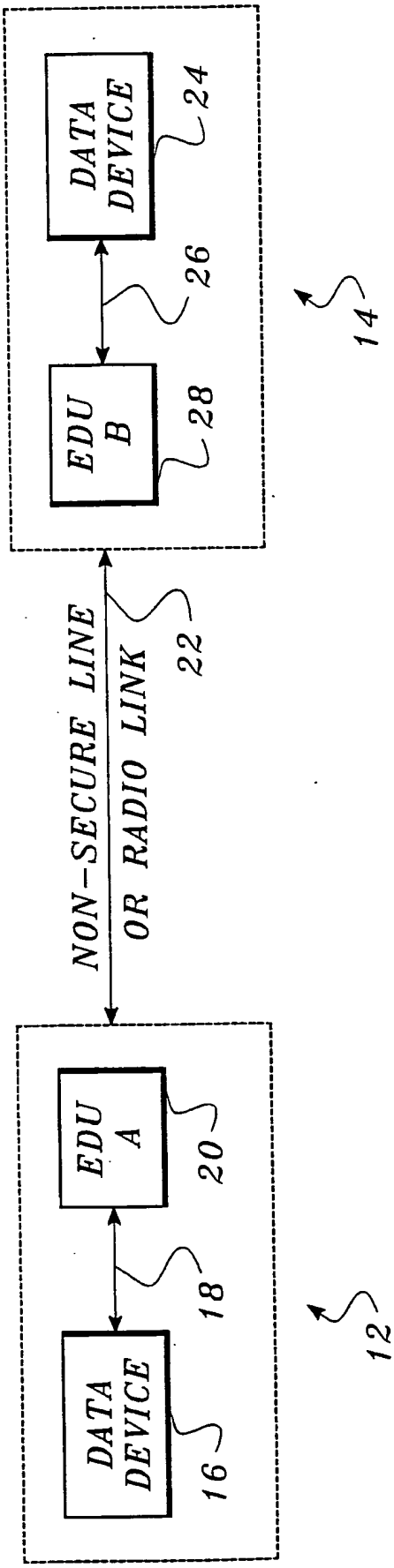

DATA DEVICE
24
26
EDU B
28
14
NON-SECURE LINE OR RADIO LINK
22
10
EDU A
20
18
DATA DEVICE
16
12

FIG. 1.

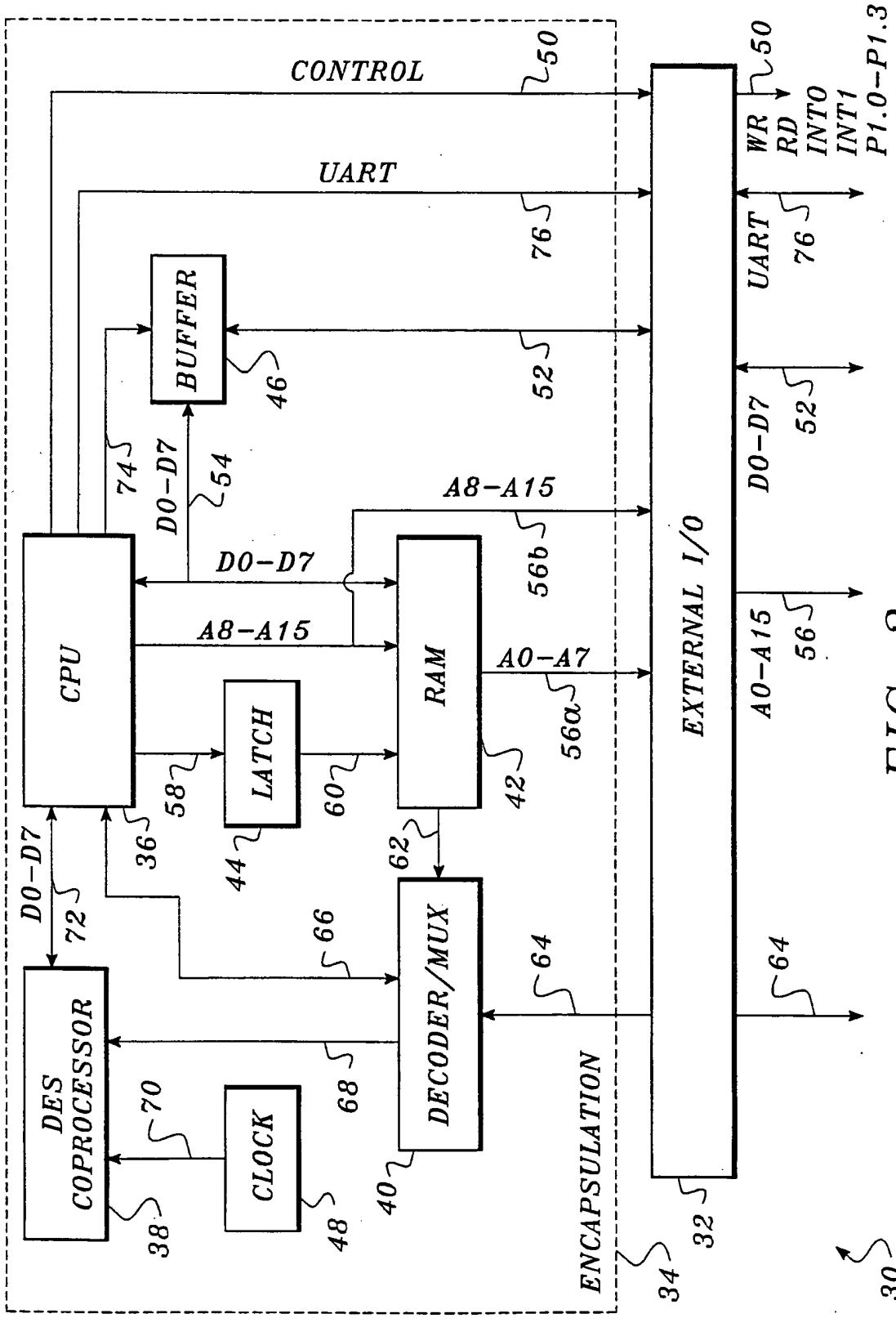

CONTROL
50
UART
76
BUFFER
46
52
74
D0-D7
54
A8-A15
56b
D0-D7
CPU
A8-A15
RAM
A0-A7
56a
LATCH
58
60
42
44
36
62
D0-D7
72
66
64
DES COPROCESSOR
DECODER/MUX
70
68
CLOCK
40
48
38
ENCAPSULATION
34
32
30
EXTERNAL I/O
50
WR
RD
INT0
INT1
P1.0-P1.3
UART
76
D0-D7
52
A0-A15
56
64

FIG. 2.

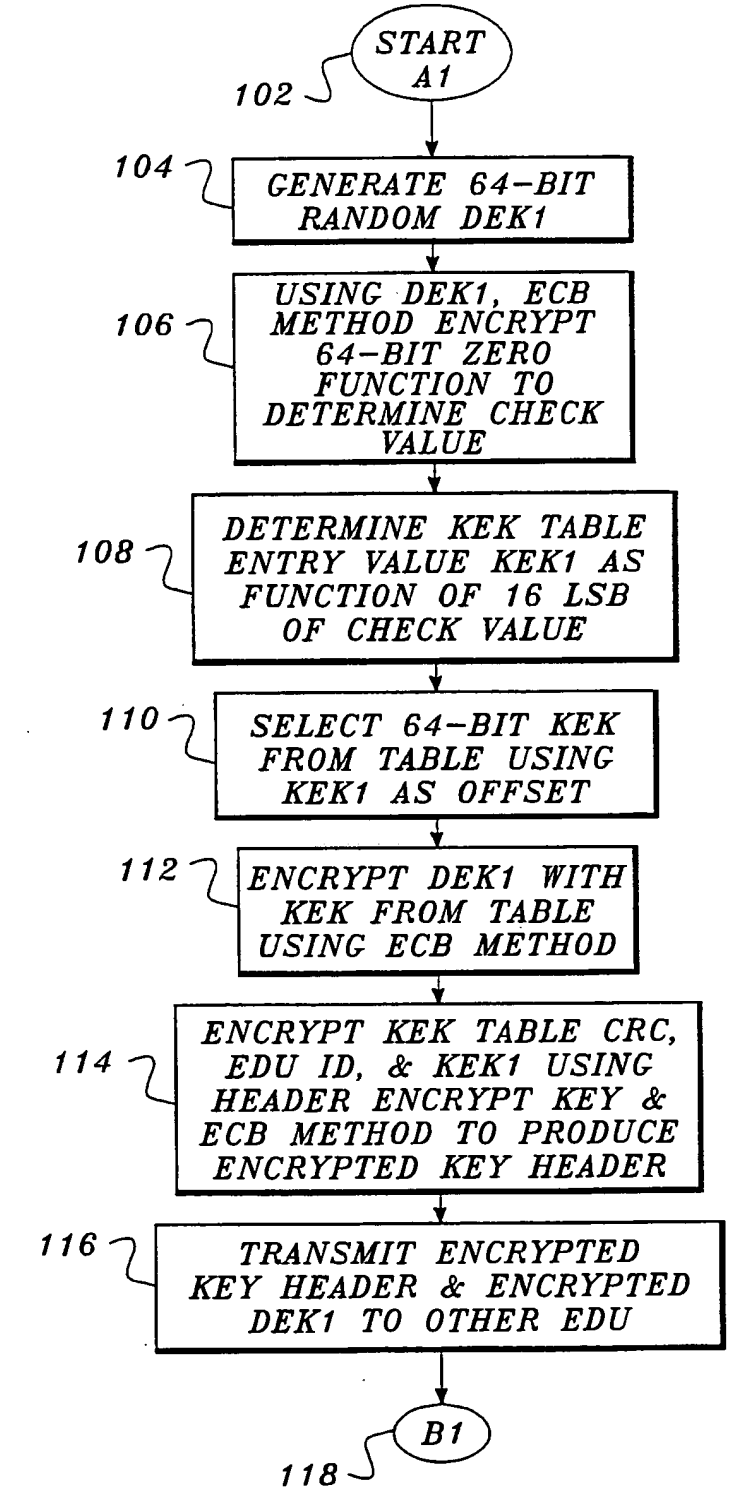

START
A1
102
104
GENERATE 64-BIT RANDOM DEK1
106
USING DEK1, ECB METHOD ENCRYPT 64-BIT ZERO FUNCTION TO DETERMINE CHECK VALUE
108
DETERMINE KEK TABLE ENTRY VALUE KEK1 AS FUNCTION OF 16 LSB OF CHECK VALUE
110
SELECT 64-BIT KEK FROM TABLE USING KEK1 AS OFFSET
112
ENCRYPT DEK1 WITH KEK FROM TABLE USING ECB METHOD
114
ENCRYPT KEK TABLE CRC, EDU ID, & KEK1 USING HEADER ENCRYPT KEY & ECB METHOD TO PRODUCE ENCRYPTED KEY HEADER
116
TRANSMIT ENCRYPTED KEY HEADER & ENCRYPTED DEK1 TO OTHER EDU
B1
118
100

FIG. 3.

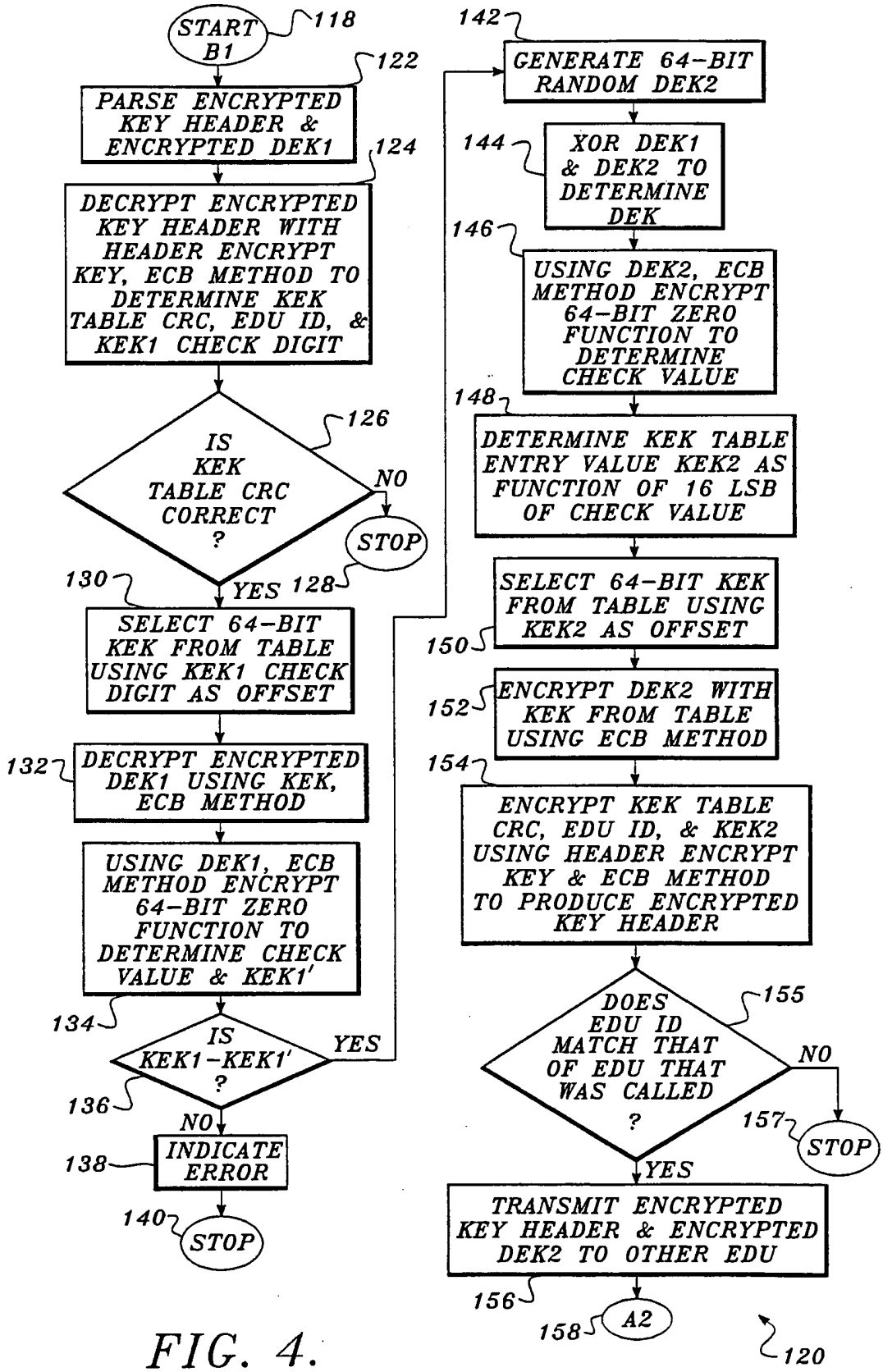

START B1
118
PARSE ENCRYPTED KEY HEADER & ENCRYPTED DEK1
122
DECRYPT ENCRYPTED KEY HEADER WITH HEADER ENCRYPT KEY, ECB METHOD TO DETERMINE KEK TABLE CRC, EDU ID, & KEK1 CHECK DIGIT
124
IS KEK TABLE CRC CORRECT ?
126
NO
STOP
128
YES
130
SELECT 64-BIT KEK FROM TABLE USING KEK1 CHECK DIGIT AS OFFSET
DECRYPT ENCRYPTED DEK1 USING KEK, ECB METHOD
132
USING DEK1, ECB METHOD ENCRYPT 64-BIT ZERO FUNCTION TO DETERMINE CHECK VALUE & KEK1'
134
IS KEK1-KEK1' ?
YES
136
NO
INDICATE ERROR
138
STOP
140
142
GENERATE 64-BIT RANDOM DEK2
144
XOR DEK1 & DEK2 TO DETERMINE DEK
146
USING DEK2, ECB METHOD ENCRYPT 64-BIT ZERO FUNCTION TO DETERMINE CHECK VALUE
148
DETERMINE KEK TABLE ENTRY VALUE KEK2 AS FUNCTION OF 16 LSB OF CHECK VALUE
SELECT 64-BIT KEK FROM TABLE USING KEK2 AS OFFSET
150
ENCRYPT DEK2 WITH KEK FROM TABLE USING ECB METHOD
152
154
ENCRYPT KEK TABLE CRC, EDU ID, & KEK2 USING HEADER ENCRYPT KEY & ECB METHOD TO PRODUCE ENCRYPTED KEY HEADER
DOES EDU ID MATCH THAT OF EDU THAT WAS CALLED ?
155
NO
157
STOP
YES
TRANSMIT ENCRYPTED KEY HEADER & ENCRYPTED DEK2 TO OTHER EDU
156
158
A2
120

FIG. 4.

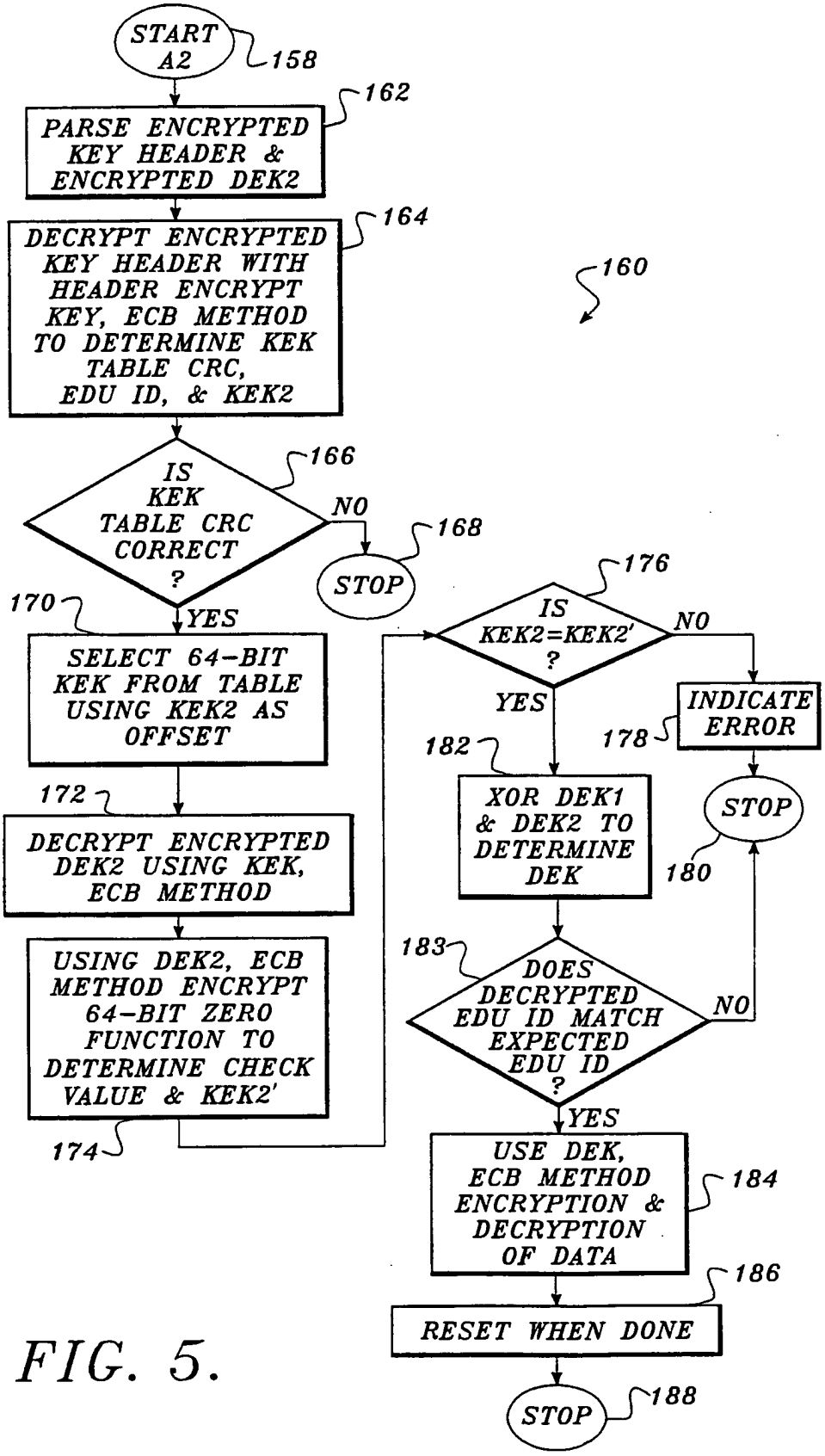

START A2
158
162
PARSE ENCRYPTED KEY HEADER & ENCRYPTED DEK2
164
DECRYPT ENCRYPTED KEY HEADER WITH HEADER ENCRYPT KEY, ECB METHOD TO DETERMINE KEK TABLE CRC, EDU ID, & KEK2
160
166
IS KEK TABLE CRC CORRECT ?
NO
168
STOP
YES
170
SELECT 64-BIT KEK FROM TABLE USING KEK2 AS OFFSET
172
DECRYPT ENCRYPTED DEK2 USING KEK, ECB METHOD
USING DEK2, ECB METHOD ENCRYPT 64-BIT ZERO FUNCTION TO DETERMINE CHECK VALUE & KEK2'
174
176
IS KEK2=KEK2' ?
NO
INDICATE ERROR
178
STOP
180
YES
182
XOR DEK1 & DEK2 TO DETERMINE DEK
183
DOES DECRYPTED EDU ID MATCH EXPECTED EDU ID ?
NO
YES
USE DEK, ECB METHOD ENCRYPTION & DECRYPTION OF DATA
184
186
RESET WHEN DONE
STOP
188

FIG. 5.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/04340

A. CLASSIFICATION OF SUBJECT MATTER

IPC(5) :H04L 9/00
US CL :380/21, 49, 28

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. : 380/21, 49, 28

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

380/52, 46

Electronic data base consulted during the international search (name of data base and, where practicable, 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.
A	US, A 5,029,208 (Tanaka) 02 July 1991	1-29
A	US, A 5,124,117 (Tatebayashi, et al.) 23 June 1992	1-29
A, P	US, A 5,144,665 (Takaragi, et al.) 01 September 1992	1-29
A	US, A 4,607,137 (Jansen, et al.) 19 August 1986	1-29
A	US, A RE33189 (Lee, et al.) 27 March 1990	1-29
A	US, A 4,578,531 (Everhart, et al.) 25 March 1986	1-29
A	US, A 4,876,716 (Okamoto) 24 October 1989	1-29

☐ 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 10 August 1993	Date of mailing of the international search report 26 AUG 1993
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. NOT APPLICABLE	Authorized officer David Cain for Telephone No. 308-0463

Form PCT/ISA/210 (second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 1/00, 11/34 — A1

(11) International Publication Number: WO 93/11480

(43) International Publication Date: 10 June 1993 (10.06.93)

(21) International Application Number: PCT/US92/10215

(22) International Filing Date: 24 November 1992 (24.11.92)

(30) Priority data:
07/798,934 27 November 1991 (27.11.91) US

(71) Applicant: INTERGRAPH CORPORATION [US/US]; One Madison Industrial Park, Huntsville, AL 35894 (US).

(72) Inventors: BAINS, Jeffrey, E. ; 134 Michli Road, Madison, AL 35758 (US). CASE, Willard, W. ; 104 Arden Avenue, Madison, AL 35758 (US).

(74) Agents: SUNSTEIN, Bruce, D. et al.; Bromberg & Sunstein, 10 West Street, Boston, MA 02111 (US).

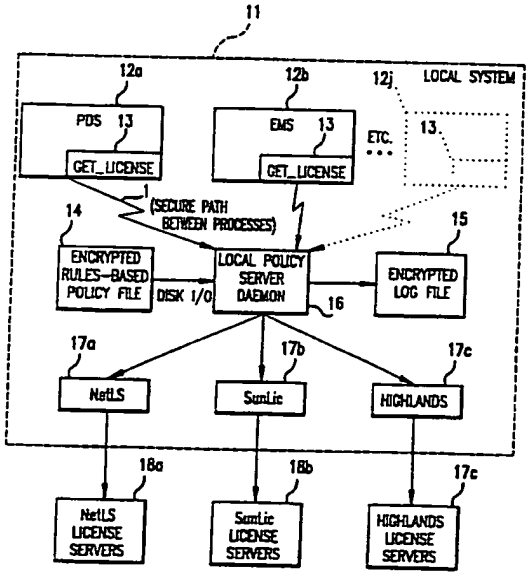
(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM AND METHOD FOR NETWORK LICENSE ADMINISTRATION

(57) Abstract

Disclosed is a system for administration, on a computer network, of license terms (a so-called license server) for a software product (12a, 12b... 12j) provided to said network. Said license server (17c, 18a, 18b) being realized by one of the network computers and which tasks comprise e.g. tracking of a software product (12a, 12b... 12j) usage in the system, issuing usage permits (licenses) to the different network users in accordance with predefined conditions, monitoring expirations and violations (e.g. the maximum number of users simultaneously using a software product) of issued licenses and when necessary, withdrawing issued software product licenses. In one embodiment, the system includes a policy server database (14) maintained on each node (11) of the system, where said predefined conditions are specified under which usage of a software product (12a, 12b... 12j) is permitted at the respective system nodes (11). Each node also has a policy server "daemon" (16) in association with said policy server database (14) for interaction with the license server (17c, 18a, 18b) in order to enforce license terms for a software product.



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
AU	Australia
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CS	Czechoslovakia
CZ	Czech Republic
DE	Germany
DK	Denmark
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IT	Italy
JP	Japan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakhstan
LI	Liechtenstein
LK	Sri Lanka
LU	Luxembourg
MC	Monaco
MG	Madagascar
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SK	Slovak Republic
SN	Senegal
SU	Soviet Union
TD	Chad
TG	Togo
UA	Ukraine
US	United States of America
VN	Viet Nam

SYSTEM AND METHOD FOR NETWORK LICENSE ADMINISTRATION

Technical Field

The present invention relates to systems and methods for controlling and monitoring the use, on computers and computer networks, of computer programs.

Background Art

As computers have proliferated in availability, the investment in computer software has also grown, and there have been developed various methods for charging the computer user for use of computer software products. Typically computer software products are licensed, rather than sold, to the computer user under various arrangements. The simplest common license arrangement gives the user the right to use a software product on a single computer, i.e., to employ only one central processing unit (CPU) in connection with operation of the software product. Although many such licenses are for indefinite periods of time, a license may also be for a limited duration and extendable, so that the entity marketing the product can charge a periodic fee (for example, annually) for use of the software product. Or use may be absolutely time-limited (for example, one-day), so that the user may evaluate the software product for possible purchase of a regular license.

Since software can be copied and moved easily from one like machine to another, companies have invented methods to prevent unauthorized use of their software products. Some licensors require passwords to activate software on a particular machine. The password may be keyed to the hardware's identification number as a condition for operation of the software. Such systems can effectively lock software to a particular machine, but do not address

software that is licensed for concurrent or simultaneous use. Some licensors use hardware locks that attach to a parallel printer port or a serial port on a machine; each time the software is activated, it looks for a specified code, in the hardware lock, as a condition for operation of the software. Using hardware locks resolves the problem of unauthorized moving of software among machines; however, hardware locks do not handle multiple software products on a single machine, and they require time and expense to deliver to the end user.

When computer software products are used in a network environment (which may include computers running in various roles as workstations and servers of various types linked together over a data path), additional licensing challenges are present. For example, a network may permit a user at one node (which may be a terminal or workstation, for instance) to utilize a software product running at another node (which may be the network server or even another workstation). Consequently, the terms of the single-computer type of software license might not cover the usage of the software product on the network, or worse still (from the point of view of the licensor) might actually permit such a usage without additional compensation to the licensor. One approach to network licensing is to grant permission to use the program based on all of the nodes on the network, and to require a license for each node. Then typically the license fee may be increased as the number of nodes on the network increases. Another approach bases the license fee for a software product running on a network on the total number of individual users who might actually run the software, regardless of the number of nodes either on the network or running the software product at a given time. These approaches, however, have usually required the cooperation of the licensee, because additional nodes may be added to the network, or additional users may utilize the software, without the knowledge of the licensor, who is typically not present on the premises of the licensee. The licensor may

reserve the right to audit the licensee's site, but such an audit is intrusive, expensive, and may alienate potential or actual customers for licenses. Although other approaches exist under which one might charge a single fee per server or per site or per entity, often on an individually negotiated basis, these approaches are often impractical or inflexible, in that they also typically do not take into account the possible wide variation over time in the number of nodes or users and also require reliance on licensee cooperation.

The same circumstances that make license enforcement difficult for the licensors of software products for a network environment also make license compliance difficult for the conscientious administrator, for example, of a Management Information System (MIS) or Computer Aided Design (CAD) department of a company using software products. The administrator may be called upon to ensure that the number of workstations using a variety of software products in a network environment complies with the terms of a variety of license agreements. Such an administrator may have to develop and promulgate a series of directives about the terms of permitted workstation usage and must depend primarily upon the goodwill and voluntary compliance of unit personnel with such directives.

Recently it has become practical in some network environments to determine and limit the number of nodes that may access a software product at a given time, and to charge a license fee based on the maximum number of nodes that are permitted to use the software product concurrently. This is called "concurrent licensing". In these environments, a computer program, acting as "librarian" and running on a computer node designated as a license server, is typically used to distribute license keys (sometimes called "tokens") over the network to nodes requesting access to run a software product; the number of keys is tracked by the librarian; and if at a given time, the permitted maximum number of keys would be exceeded by usage of the software

product on a requesting node, the node can be denied, at such time, access to invoke the software product.

Examples of software-based concurrent licensing arrangements may be found in Unix applications running in connection with software products sold under the trademarks NetLS (available from Gradient Technologies, Inc., 577 Main Street, Suite 4, Hudson, Massachusetts 01749), and SunLic (available from Sun Microsystems, Inc., Mountain View, California), and Flexible License Manager (available from Highland Software, Inc., 1001 Elwell Court, Palo Alto, California 94303). However these arrangements suffer from a number of disadvantages. NetLS, for example, includes mechanisms for tracking which nodes have been given keys to run a given software product and the number of keys available for running such software product. However, it is up to the designers of each software product to program such product to implement the terms of any license agreement, and, in particular, to program into the product calls to the NetLS software to provide information to the computer running the software product and to write code in the applicable product to prevent use of the product when the license terms have not been obeyed. Thus a computer system utilizing ten different software products that rely on NetLS for license enforcement will generally have ten different substantial software portions (one in each computer product) to achieve license enforcement. In addition to this complexity, if the license server running NetLS fails, of if the network itself fails, then a workstation loaded with the software product cannot run the software product, since the product requires NetLS interaction to be activated.

The foregoing difficulties are applicable generally not just to NetLS but to "metering software" generally. The Microcomputer Managers Association has issued a White Paper (October 2, 1991), reprinted in Infoworld, pages 46-42 (October 14, 1991) on the problems of network licensing, Commenting on the problem that each software product requires its own interface to the metering software (as well

as possible input of administrative information), the White Paper suggests that "[i]t makes much more sense to have a single package provide the metering for all application software on the network." Infoworld (October 14, 1991), supra, at page 51, column 4. Such an approach has its own difficulty, however. Each application would still have to interface with the single metering package, and the interface to such a package must somehow deal with the varying licensing terms of each software product. Moreover, with the metering package running on the license server, a failure of the server or the network would prevent all software applications from running anywhere on the network.

Summary of the Invention

In a preferred embodiment, the present invention provides an improved system for administration, on a computer network, of license terms for a software product on the network. The improved system is of the type having an arrangement, such as NetLS, for tracking software product usage, associated with one of the computers acting as a license server. This arrangement permits the license server (i) to identify the current set of nodes that are using the software product at a given time, (ii) to handle license data concerning conditions under which usage of the software product is permitted at any given node, and (iii) to determine whether at any given time the conditions would still be satisfied if a given node is then added to this set of nodes. The software product may thus include instructions to interface with the license server to cause enforcement of the license terms. The improvement, in one embodiment, to the system includes a policy server database maintained on each node, containing data specifying conditions under which usage of the software product is permitted on such node. Each node also has a policy server "daemon" (which may be implemented in software) in association with the corresponding policy server database, for (i) communicating with the license server, (ii) interfacing with both the software product and the

corresponding policy server database, (iii) making a permission-to-run availability determination, with respect to local usage of the software product, on the basis of applicable data from the license server and the corresponding policy server database, so that enforcement of license terms applicable to the software product at a given local node is achieved on the basis of both license policy maintained at such local node as well as applicable data from the license server.

In a further embodiment, each policy server database contains data specifying conditions under which usage of each of a plurality of software products is permitted on the node on which the database is maintained. Additionally, each policy server daemon interfaces with each software product. In this manner, enforcement of license terms applicable to each software product at a given node is achieved on the basis of both locally maintained license policy and applicable data from the license server.

In a further embodiment, each node has a log file maintained, in association with each policy server daemon, to record recent software product usage on that node. The policy server daemon is accordingly configured to handle instances when data from the license server is unavailable -- for example, when the computer acting as the license server is non-operational or when the network is non-operational. In particular, the policy server daemon may permit a node to run a software product, in the absence of license server data, if the node's log file indicates a sufficient level of recent usage of the software product on the node. The circumstances under which such a permission-to-run availability determination is favorable may be established by the node's policy server database.

In yet further embodiments, the policy server database and the log file may be encrypted. Furthermore the interface between the policy server daemon and each software product may be made secure. When one or more of the software products are subject to concurrent licensing

restrictions specified in the policy server databases, the policy server daemon may be permitted to reserve a predetermined time interval over which the applicable node has a guaranteed opportunity to utilize a given software product. The reservation is accomplished by having the node's policy server daemon communicate to the license server over the predetermined time interval that the node is using the given software product, regardless whether the software product is actually being used.

It can be seen that the present invention permits a single database at each node to specify all of the conditions under which the node may access any of the software products on the network. Furthermore, as described in further detail below, in order to invoke the licensing administration function carried out in accordance with the present invention, each software product need contain only a simple and short segment including the instruction:

ILic-get_license

followed by parameters identifying license details for the particular software product. A branching routine (which may made available to all the software products, and called by the particular software product after this instruction) then specifies program flow depending on whether a license is available (the remainder of the program can be run) or not (the program operation is terminated and a message is displayed to the user).

Brief Description of the Drawings

The foregoing features of the invention will be more readily understood by reference to the following description taken with the accompanying drawings in which:

Fig. 1 is a block diagram showing operation of a preferred embodiment of the invention in a network;

Fig. 2 is a block diagram illustrating the interrelation of important modules of the embodiment;

Fig. 3 is a block diagram of the main logical flow of

the computer program used in the embodiment;

Fig. 4 is a block diagram of the main processing of a validated "get license" message;

Fig. 5 is a block diagram of the manner in which the policy server database is structured;

Fig. 6 is a block diagram providing more detail than Fig. 4 of the logical flow of the processing of a "get license" message;

Fig. 7 is a block diagram of license acquisition processing referred to as item 623 in Fig. 6;

Fig. 8 is a block diagram of clock message processing for licenses on the main list of licenses that have been established; and

Fig. 9 is a block diagram of clock message processing for licenses moved to the recovery list by item 88 of Fig. 8.

Detailed Description of Specific Embodiments

The invention is applicable to computer networks of the type having an arrangement, such as NetLS, for tracking software product usage, associated with one of the computers on the network acting as a license server. The present embodiment is described with respect to a Unix network; however, the software used by the license server in implementing such an arrangement, and the particular network type, are a matter of design choice.

Fig. 1 shows the manner in which a preferred embodiment of the invention may be implemented on a Unix network. Each computer node 11 of the network may be running a variety of software products, such as PDS (item 12a), EMS (item 12b), and so forth (shown through item 12j). Each of these products includes a call "get_license" to the local policy server daemon 16 for a determination whether a license is available to run the product in question. As used in this detailed description, the term "license" refers not to a written document between the licensor and the licensee, but rather to the availability of permission to run the software product. The local policy server daemon 16 operates at the

computer node 11 and makes the permission-to-run availability determination by reference to its associated policy server database 14, also located at the node, to identify the rules specifying the circumstances under which a license would be granted. The daemon 16 also communicates over the network with the applicable license server. In this figure, three separate license servers are shown: one (item 18a) running NetLS; another (item 18b), SunLic; and another (item 18c), Highlands. The license server communicates with the daemon 16 using the applicable license software NetLS (item 17a), SunLic (item 17b), or Highlands (item 17c), and informs the daemon whether usage of the software product on the network is such that a license may be granted in accordance with the policy established by the database 14. If so, the daemon 16 reports the license to the applicable software product 12, and to the applicable license server 18. If there is no successful communication with the applicable license server 18, if the database 14 so permits, the daemon 16 will consult a log file 15 recording instances of recent software product usage, and if there has been a sufficient level of recent software product usage that has been licensed, the daemon will grant a temporary user license (TUL) to run the software product.

The communication between the applicable software product 12 and the local policy server daemon 16 is handled as an interprocess communication in Unix. Here the Unix "message" is used as the means of communication, but this is a matter of choice, and other means of communication, such as pipes or shared memories, may be used. In order to reduce to risk of tampering by the licensee with the license availability determination made by the policy server daemon 16, the rules database 14 and the log file 15 may be encrypted using techniques known in the art. Similarly, the message communication from the application to the policy server daemon 16 can be subject to validation using techniques known in the art to assure that the message is indeed from the pertinent software product.

The embodiment described herein has been implemented for use with a variety of types of licenses. (Numerous license types may be created and enforced by the invention, but the following types are illustrative.) One type is the concurrent use license. A concurrent use license is issued, from the server running the Licensing System (sometimes called the "license server" in this description and the claims following), with respect to a software product being used on a node in a network. The license server controls the levels of concurrent usage of the software product. Concurrent use licenses are returned to the license server when they are no longer needed. For example, if the Licensing System on the server permits five concurrent licenses for a given software product, then five users on the network can run the software product concurrently.

The concurrent use license is actually implemented as part of a two-tier structure. The first tier is a "base license," and the second tier is a "version-specific" license. The base license controls the number of simultaneous users of a software product. The version license controls the version of the software product that may be utilized by the user. The base license typically expires at the end of each year, and may be renewed. The version license typically never expires. The version license provides a mechanism for controlling how many base licenses are for a software product that is under a maintenance agreement. As an example, a user may have purchased a license to five copies of version A of a software product, but kept maintenance on only three copies. In such a case the user would receive five base licenses (which expire each year and were replenished unless the applicable computers were sold), plus five version A licenses that never expire. This user would subsequently receive only three version B licenses for the three copies under maintenance. Under such an arrangement, the user could still run five copies of version A of the software product, or a mix of version A and version B software as

long as the mix does not exceed five copies in total and does not exceed three copies of version B.

The policy server database file 14 of a computer node stores the license requirements for each software product to be run at that node. For each software product, the database may identify the number of base license "tokens" and version license "tokens", obtained from the server running the Licensing System, that are necessary for operation of that software product on the particular computer consituting the node. (The particular computer, for example, may be particularly fast in processing, and therefore a higher license fee may be required for running the software product on such computer, resulting here in a larger number of tokens required for the base and version licenses.)

Another type of license is a node-locked license, which is tied to a particular computer node and cannot be used by other nodes. The node-locked license token is designated for a particular node when created. In a further variation of the node-locked license, a "reserved" license may be established, that is, the policy server daemon may be permitted to reserve a predetermined time interval over which the applicable node has a guaranteed opportunity to utilize a given software product. (The reservation is accomplished by having the node's policy server daemon communicate to the license server over the predetermined time interval that the node is using the given software product, regardless whether the software product is actually being used.)

A single use license can be used only for one invocation of the software product. Single use licenses are useful for emergency situations, peaks in usage, or demonstrations. A day use license is similar to a single use license, except that a day use license remains available on the computer node that acquired it for 24 hours after the time of acquisition.

A temporary user license (TUL), described above, is

issued on a temporary basis when the server running the Licensing System becomes unavailable. A TUL is designed for emergency situations and is grated on a per user, per node, per software product, per usage history basis.

Fig. 2 illustrates the structure of a program implementing the embodiment of Fig. 1 for a single license server running one or more Licensing Systems, such as NetLS. The program is written in standard C. A communication module 21 handles communication with the various software products, one of which is here shown as item 26. If the software product includes the "get_license" instruction, the communication module 21 refers to the licensing dispatch module 22. The licensing dispatch module 22, by reference to the policy server database 14 and the applicable Licensing System, makes the license availability determination. The Licensing System shown here is No. 1, and client portion 25 is accessed by licensing dispatch 22, which may access other Licensing Systems depending on the software product 26 and information in the policy server database file 14. The client portion of the Licensing System 25 communicates over the network with the server portion 251. In the event that there is no successful communication with the server portion 26, the communication module may trigger the temporary user license (TUL) module 27 to consult with the history log file 15 to determine if there is a sufficient level of recent licensed usage of the software product at this node to permit the grant of a temporary user license (TUL). In any event, the communication module 21 reports the license availability determination by directing a message to the software product's process. The communication module is also responsible for sending a periodic signal (a "ping") to the license server to indicate continued use of a license. Another module 28 causes recordation of license usage in license usage file 281 for reporting purposes. A file 252 of node-locked licenses is maintained locally. The communication module 21 is controlled by timer interval

handler 29, which in turn receives periodic signals from PS driver 291 that has been incorporated into the operating system.

Fig. 3 illustrates the main logical flow of the program carried out by the communication module 21. After initialization 31, the program gets the next message from any processes, and, in particular, from any software products that may be invoked from the node on which the program is running. Next the message is validated (item 33), and then the message is processed (item 34). After processing of the current message, the program loops to seek the next message again.

The most important message is "get_license", and this message is processed as shown in Fig. 4. The first step 41 is to determine the availability of a license. The license availability determination is made in the licensing dispatch module 22.

After the license availability determination is made as shown in step 41 of Fig. 4, if a license is granted, that fact is reported to the software product in step 43. If the license is not granted because of a lost connection to the server running the Licensing System (determination in step 44), there is a check to see if usage of the software product is possible "under grace", that is, whether there has been sufficient recent licensed usage of the software product at the node to permit granting of a TUL. If so, a TUL is granted (step 47). If not, or if the license was denied for reasons other than a lost connection, the program communicates (step 46) the fact of no license availability to the software product.

Additionally, the communication module of the policy server daemon may reserve a predetermined time interval over which the applicable node has a guaranteed opportunity to utilize a given software product. The reservation is accomplished by having the module communicate to the license server over the predetermined time interval that the node is using the given software product, regardless whether the

software product is actually being used.

The construction of the policy server database is shown in Fig. 5. License prices are initially established by management decision in price book 51, which forms the basis for assigning token values (step 52) required for license grant. The license cost to use a software product can also vary as a function of the hardware platform (i.e., the model of the computer) on which the product is running. Accordingly, the platform indicator data 54 and the rules defining the different types of licenses 53 all form a part of the structure of the policy server database 55. In order to assure integrity of the database, it is encrypted.

Fig. 6 is a block diagram providing more detail than Fig. 4 of the logical flow of the processing of a "get license" message. Initially (step 61), memory is allocated for the structure of the applicable license to be added to the list of license structures in memory. Unless the structure shows a reserved license (tested in block 62), the policy server database file 14 of Figs. 1 and 2 is accessed (step 621) to determine the applicable license terms. If access is successful (tested in block 622), then license acquisition processing (described in connection with Fig. 7) follows (step 623).

If, as a result of license acquisition processing, a license is granted (tested in block 625), the history log file 15 of Figs. 1 and 2 is then updated (step 631) to reflect this event. Thereafter, the policy server driver 291 of Fig. 2 is informed (step 63), the license usage file 281 of Fig. 2 is updated for use in generating later reports (step 64), the return status the operation is checked (step 641), and a status message is built and sent (step 65) to the software product that had included the "get license" call. If the return status is a failure (tested in step 641), the license structure is removed from memory (step 642) before sending the the status message to the software product.

If, as a result of the license acquisition processing

of step 623, a license has not been granted, the error messages produced by the Licensing System are analyzed to a single reason (step 626), and the return status for the software product is determined. If the embodiment described herein is not in the enforcement mode (determined in step 627), then the return status is simply a warning (generated in step 643). If the embodiment is in the enforcement mode, there is a check (step 647) to determine if the connection with the license server is lost. If there is no lost connection, the policy server database file 14 is checked (step 646) for the appropriate license failure conditions, and then the return status is determined (step 644). If there is a lost connection, processing follows (step 645), to determine on the basis of the history log file 15 and data in the policy server database file 14 whether a TUL is available. If a TUL is available, the return status is a warning (step 643), as in the case when the system is not in the enforcement mode. Once the return status has been determined, processing is the same as if a license has been granted; that is, the driver is informed, the license usage file is updated, the return status is checked and if necessary the license structure is removed from memory, and the appropriate status message is built and sent to the software product (steps 63, 64, 641, 642, and 65).

If after the determination (step 647) that there is a lost connection, and a TUL is not available (step 645), processing loops back to license acquistion processing (step 623) to attempt again to get a license from the license server. If the policy server database file 14 cannot be successfully accessed in step 621 to determine the relevant license rules (a matter checked in step 622), the processing goes to determine (in step 627) whether the system is in the enforcement mode and to generate an appropriate return status. If in step 62, the license structure shows a reserved license, access to the policy server database file 14 is skipped altogether, and the driver is informed (step 63) directly.

Fig. 7 is a block diagram of the license acquisition processing referred to as item 623 in Fig. 6. In accordance with this processing, there is first sought a "base" license token and then a "version" license token, where "base" and "version" have the meanings described above following the description of Fig. 1. Initially, the policy server database file 14 is cycled through to determine the enabled base token type (step 71)--for example node-locked, or concurrent access, or use once. The Licensing System on the server is then called (step 711) to seek the designated enabled token. If the base token is granted (checked at step 712), the policy server database file 14 is then cycled through to determine the enabled version token type (step 72). If the version token is granted (checked at step 722), the return is "license granted" (step 73). In each case if processing through the policy server database is not complete (checked for, in the case of the base token at step 713 and in the case of the version token at steop 723), the database in cycled through again, the Licensing System is called to seek an enabled token, and there is a test to see if the token is granted. If the end of the list has been reached (tested at step 713 for the base token and 723 for the version token) and the applicable token has not been obtained, a failure is returned (step 725). If the base token has been granted, but the version token denied, then the base token is first freed (step 724) before the failure is returned in step 725.

Fig. 8 is a block diagram of clock message processing for licenses on the main list of licenses that have been established. First, a license is picked as part of a cycle through the main list of licenses in memory (step 81). Next there is a check whether a process exists for this license (step 82). If there is no process, the license is returned to the Licensing System, and associated housekeeping is done (step 821), and the program then picks the next license (step 81) to begin processing again. If it is determined that there is a process, then it is determined whether the

license needs to be "pinged" to satisfy requirements of the Licensing System to keep the license (step 83). The implementation here generates a ping every 10 minutes. If no ping is currently necessary, the program again picks the next license (step 81) to begin processing again. If a ping is necessary, it is sent (step 84), and if successful (i.e., the Licensing System reports that the license is still valid (tested in step 85), the program again picks the next license (step 81) to begin processing again.

If the ping is unsuccessful, a failure counter is incremented (step 86), and there is a test (step 87) to determine if the failure counter is above an established threshhold. If it is, then the failure counter is cleared (step 88) and the license in question is moved to the recovery list (step 89). If it is not, then the program again picks the next license (step 81) to begin processing again.

Fig. 9 is a block diagram of clock message processing for licenses moved to the recovery list in step 89 of Fig. 8. First, a license is picked as part of a cycle through the recovery list of licenses in memory (step 91). Next there is a check whether a process exists for this license (step 911). If there is no process, any remaining part of the license is returned to the Licensing System, and associated housekeeping is done (step 94), and the program then picks the next license (step 91) to begin processing again. A check (in step 912) is made to determine whether the exit flag had been set in step 935, and if so, the process of the software product (application) is signalled to exit (step 913), the exit counter is decremented (step 914), and a test (step 915) is made to determine if the exit counter has reached zero. If so, the application process is killed (step 916). In either event, the next license is picked from the recovery list (91), and processing for the next license resumes as before.

If the exit flag had not been set, then a replacement license is sought (step 921), and a test (922) is made to

determine whether a license has been granted. If a replacement license has been granted, then

If a replacement license has not been granted, the replacement failure counter is incremented (step 93) and then tested (step 931) to determine if it is above a threshhold (here typically 3). If it is not above the threshhold, then the next license is picked from the recovery list (91), and processing for the next license resumes as before. If it is above the threshhold, the polciy server database file 14 is consulted (step 932) to determine whether running of the software product is permitted (step 933). If not, then the exit counter and exit flag are set up; if running is permitted, the replacement failure counter is decremented (step 934). In either case, the next license is picked from the recovery list (91), and processing for the next license resumes as before.

Many other implementations of the invention described herein are possible. For example, the particular types of licenses described here are merely examples. The use of base and version licenses are thus a matter of design choice. The manner in which the failure to obtain a license is handled can also be tailored to suit the policies of the licensor of the software products in question.

What is claimed is:

1. An improved system for administration, on a computer network, of license terms for use of a software product on the network, the system being of the type wherein the network has a plurality of digital computers, each computer at a node, in communication with each other over a data path, and the system has usage tracking means, associated with one of the computers acting as a license server, for (i) causing the storage of the number of licenses available for running the software product on nodes of the network, (ii) identifying the current set of nodes with respect to which a license has been granted to run the software product at a given time, and (iii) determining whether at any given time any licenses remain to be granted for permitting an additional node to run the software product, so that the software product may include instructions to cause enforcement of the license terms;

wherein the improvement comprises:

(a) a policy server database containing data specifying conditions under which usage of the software product is permitted on any given node; and

(b) policy server means, maintained and operating locally as an independent process, on each computer, with respect to which the license terms are to be enforced, in association with the policy server database, for (i) communicating with the license server, (ii) interfacing with both the software product and the policy server database, and (iii) making a permission-to-run availability determination, with respect to local usage of the software product, on the basis of applicable data from the license server and the policy server database, so that enforcement of license terms applicable to the software product at a given local node is achieved on the basis of both license policy maintained in the policy server database as well as applicable data from the license server.

2. A system according to claim 1, wherein each computer at a node with respect to which license terms are to be

enforced includes means for maintaining locally a policy server database, containing data specifying conditions under which usage of the software product is permitted on such node.

3. A system according to claim 1, further comprising:

(c) log means for recording and maintaining a log file of recent software product usage on each computer at a node with respect to which license terms are to be enforced, such log file being accessible to such policy server means, and wherein such policy server means includes means for making a permission-to-run availability determination in the absence of data from the license server on the basis of data from the policy server database and the log file, so that a favorable determination is possible if the log file indicates a sufficient level of recent usage of the pertinent software product on the computer on which such policy server means is operating.

4. A system according to claim 2, wherein

(i) each policy server database contains data specifying conditions under which usage of each of plurality of software products is permitted on the computer on which the database is maintained, and

(ii) each policy server means includes means for interfacing with each of the software products,

so that enforcement of license terms applicable to each software product at a given local node may be achieved on the basis of both license policy maintained at such local node as well as applicable data from the license server.

5. A system according to claim 4, further comprising:

(c) log means, maintained locally in association with each policy server means, for recording and maintaining a log file of recent software product usage on the computer on which such log means is maintained, such log file being accessible to such policy server means, and wherein such policy server means includes means for making a permission-to-run availability determination in the absence of data from the license server on the basis of data from the policy

server database and the log file, so that a favorable determination is possible if the log file indicates a sufficient level of recent usage of the pertinent software product on the computer on which such policy server means is operating.

6. A system according to claim 1, wherein the policy server database is encrypted.

7. A system according to claim 5, wherein the policy server database and the log file are encrypted.

8. A system according to claim 4, wherein the policy server means include means for maintaining a secure interface with each of the software products.

9. A system according to claim 7, wherein the policy server means includes means for maintaining a secure interface with each of the software products.

10. A system according to claim 4, wherein one of the policy server databases includes a limit on the number of nodes that may simultaneously use a given software product and wherein the corresponding policy server means associated with such policy server database includes reservation means for informing the license server, over a predetermined time interval, that the node associated with such policy server means is using the given software product, regardless whether such software product is actually being used, so that such node will always be available to use such software product, despite attempts to use the software product at other nodes which if successful would otherwise foreclose use at such node of such software product, with the effect that the reservation means reserves use of such software product at such node over the predetermined time interval.

11. A system according to claim 5, wherein one of the policy server databases includes a limit on the number of nodes that may simultaneously use a given software product and wherein the corresponding policy server means associated with such policy server database includes reservation means for informing the license server, over a predetermined time interval, that the node associated with such policy server

means is using the given software product regardless of whether such software product is actually being used, so that such node will always be available to use such software product despite attempts to use the software product at other nodes which if successful would otherwise foreclose use at such node of such software product, with the effect that the reservation means reserve use of such software product at such node over the predetermined time interval.

12. A system according to claim 9, wherein one of the policy server databases includes a limit on the number of nodes that may simultaneously use a given software product and wherein the corresponding policy server means associated with such policy server database includes reservation means for informing the license server, over a predetermined time interval, that the node associated with such policy server means is using the given software product, regardless whether such software product is actually being used, so that such node will always be available to use such software product, despite attempts to use the software product at other nodes which if successful would otherwise foreclose use at such node of such software product, with the effect that the reservation means reserves use of such software product at such node over the predetermined time interval.

13. A computer network comprising:

(a) a plurality of digital computers, each computer at a node, in communication with each other over a data path;

(b) usage tracking means, associated with one of the computers acting as a license server, for (i) causing the storage of the number of licenses available for running the software product on nodes of the data path, (ii) identifying the current set of nodes with respect to which a license has been granted to run the software product at a given time, and (iii) determining whether at any given time any licenses remain to be granted for permitting an additional node to run the software product;

(c) a policy server database, maintained locally on such computer with respect to which it is desired to enforce

license terms applicable to usage of the software products, containing data specifying conditions under which usage of any given one of the software products is permitted on the computer on which the database is maintained; and

(d) policy server means, maintained and operating locally, on each computer with respect to which it is desired to enforce license terms applicable to usage of the software products, and in association with the corresponding policy server database, for (i) communicating with the license server, (ii) interfacing with both (aa) each of the software products and (bb) the corresponding policy server database, and (iii) making a permission-to-run availability determination, with respect to local usage of any given software product, on the basis of applicable data from the license server and the corresponding policy server database, so that enforcement of license terms applicable to the given software product at a given node is achieved on the basis of the license policy maintained at such local node as well as applicable data from the license server.

14. A computer network according to claim 13, further comprising:

(e) log means, maintained locally in association with each policy server means, for recording and maintaining a log file of recent software product usage on the computer on which such log means is maintained, such log file being accessible to such policy server means, and wherein such policy server includes means for making a permission-to-run availability determination in the absence of data from the license server on the basis of data from the policy server database and the log file, so that a favorable determination is possible if the log file indicates a sufficient level of recent usage of the pertinent software product on the computer on which such policy server is operating.

15. A computer network according to claim 14, wherein the policy server database and the log file are encrypted.

16. A computer network according to claim 12, wherein the policy server means includes means for maintaining a secure

interface with each of the software products.

17. A computer network according to claim 15, wherein the policy server means includes means for maintaining a secure interface with each of the software products.

18. A digital storage medium encoded with instructions for a given computer in a computer network of the type having:

(i) a plurality of digital computers, each computer at a node, in communication with each other over a data path;

(ii) usage tracking means, associated with one of the computers acting as a license server, for (i) causing the storage of the number of licenses available for running the software product on nodes of the network, (ii) identifying the current set of nodes with respect to which a license has been granted to run the software product at a given time, and (iii) determining whether at any given time any licenses remain to be granted for permitting an additional node to run the software product,

the instructions when loaded into the given computer establishing:

(a) data structure for a policy server database, maintained locally on the given computer, containing data specifying conditions under which usage of any given one of the software products is permitted on the given computer; and

(b) policy server means, maintained and operating locally, on the given computer, and in association with the policy server database, for (i) communicating with the license server, (ii) interfacing with both (aa) each of the software products and (bb) the policy server database, and (iii) making a permission-to-run availability determination, with respect to local usage of any given software product, on the basis of applicable data from the license server and the policy server database, so that enforcement of license terms applicable to the given software product at the given computer is achieved on the basis of the license policy maintained at the given computer as well as applicable data from the license server.

19. A system, for administration of license terms for use of a software product on a computer, comprising:

(a) a policy server database containing data specifying conditions under which usage of the software product is permitted on the computer;

(b) policy server means, operating on the computer, in association with the policy server database, for (i) interfacing with the software product and the policy server database and (ii) making a permission-to-run availability determination, with respect to usage of the software product, on the basis of data from the policy server database.

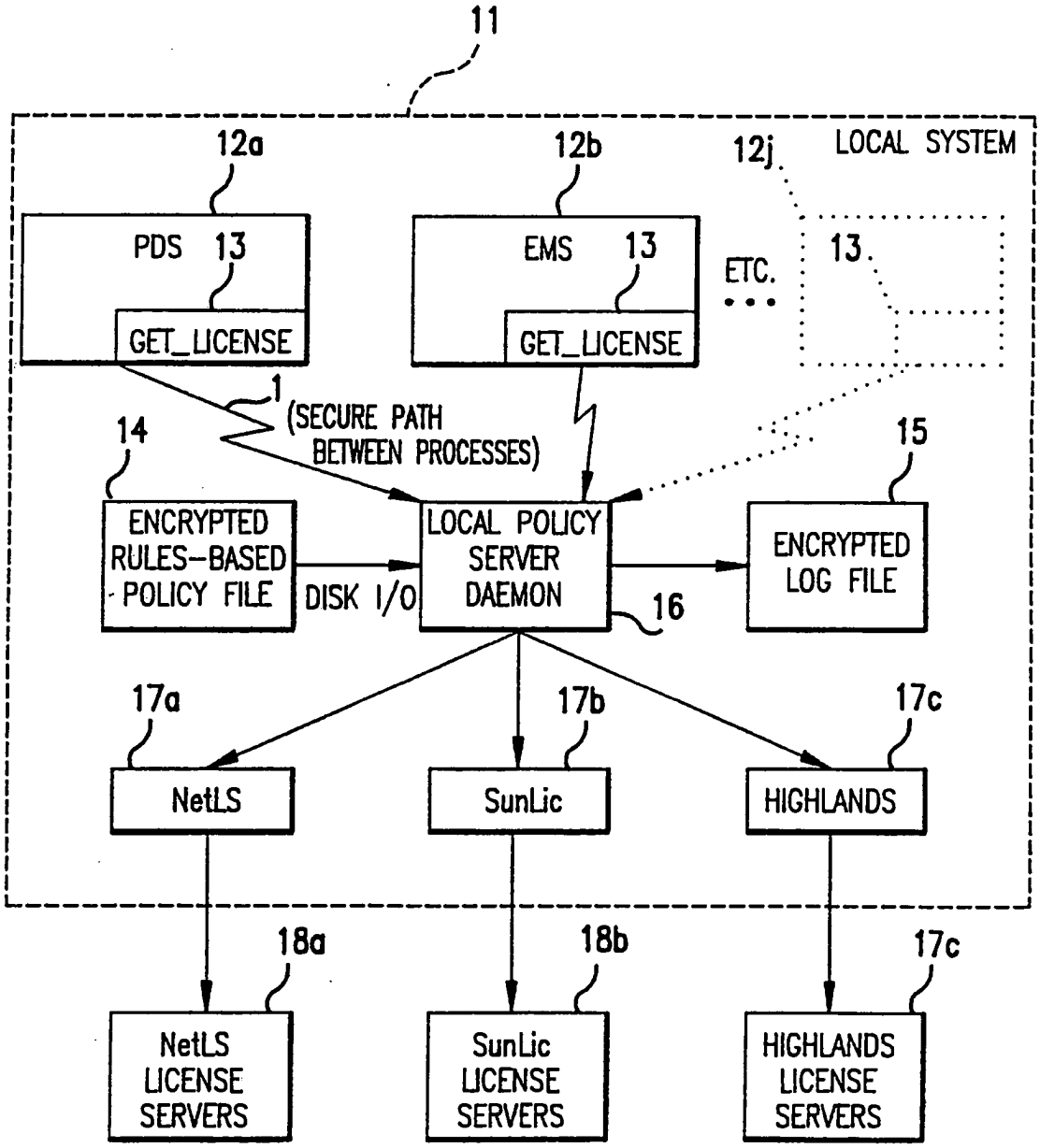


FIG.1

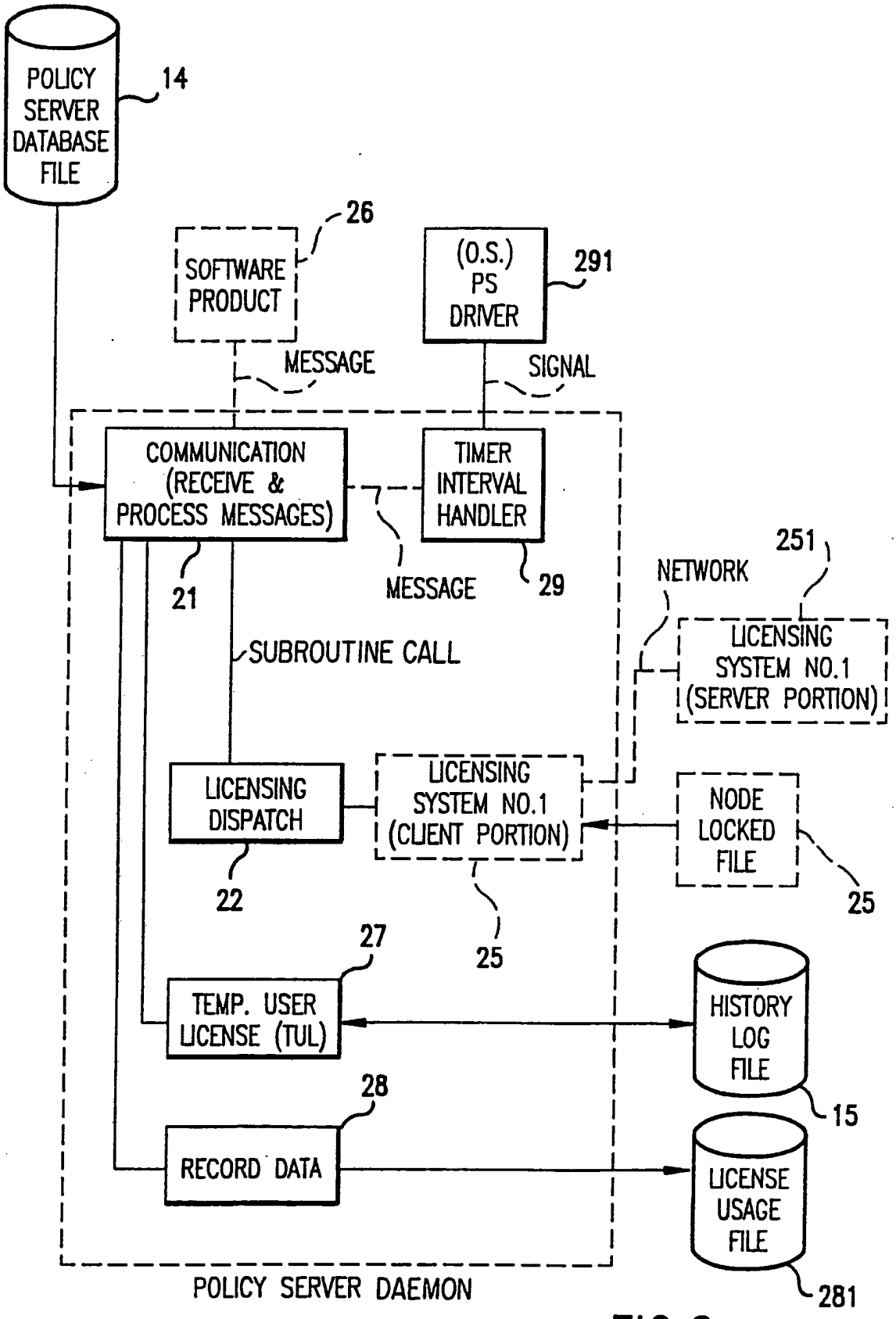


FIG.2

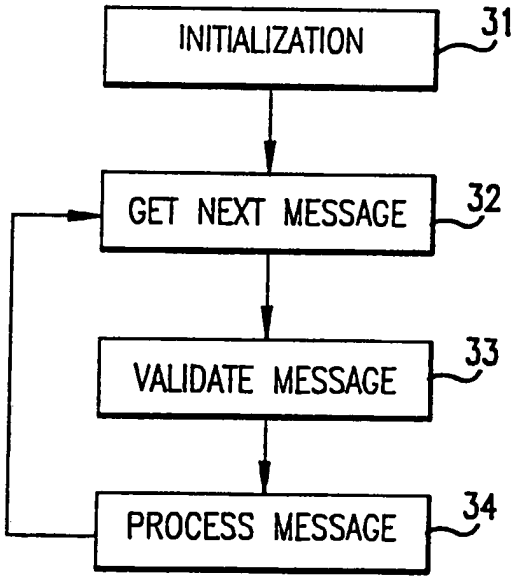


FIG.3

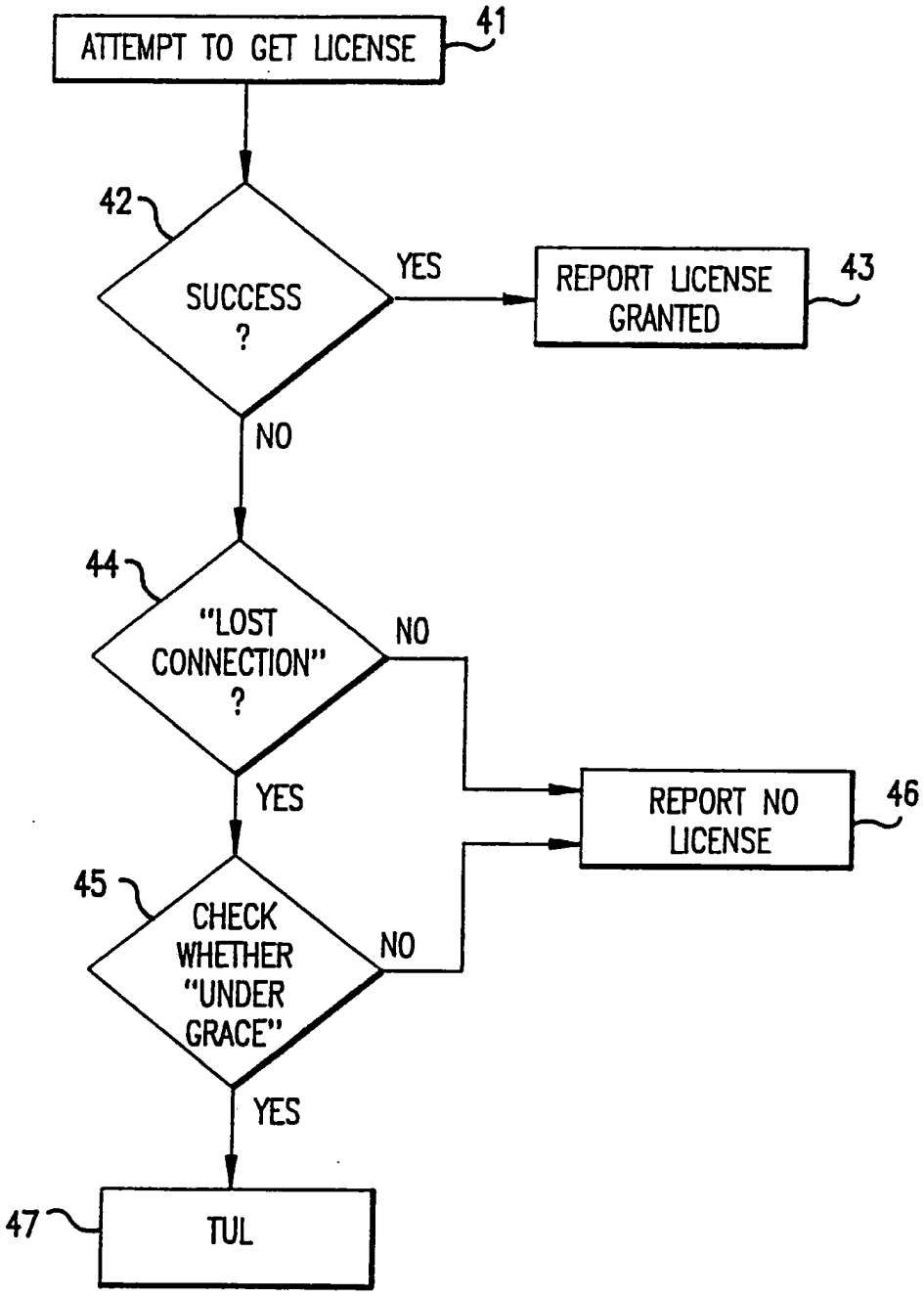


FIG.4

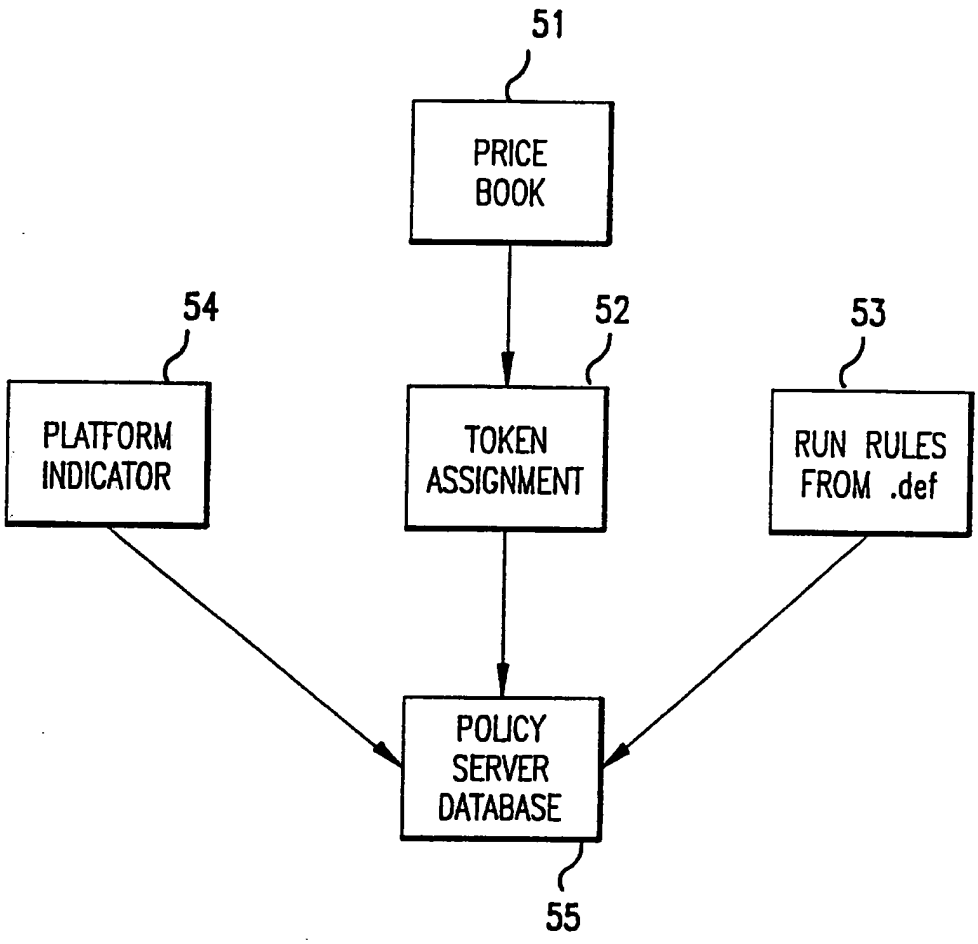


FIG.5

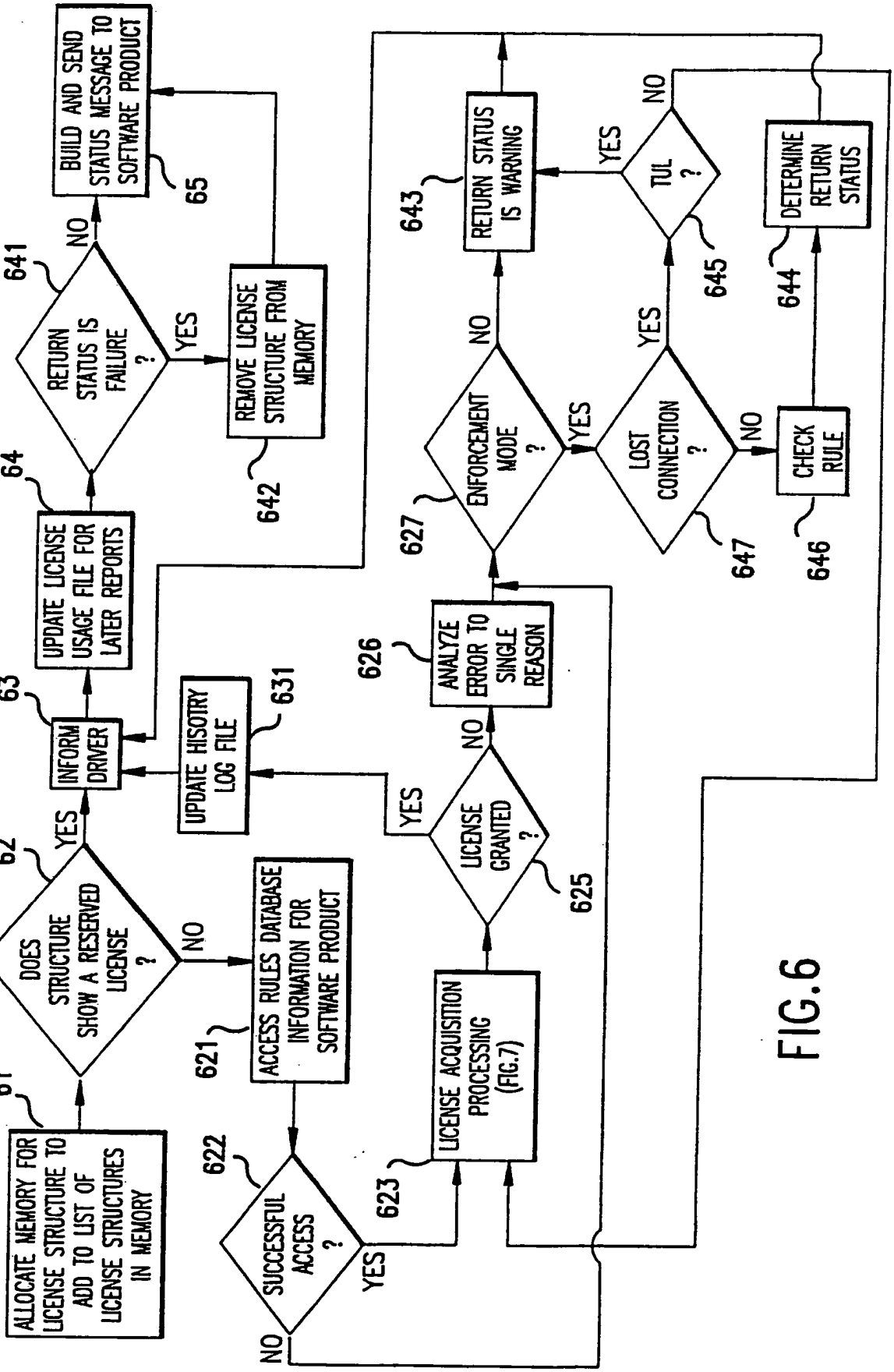


FIG.6