

100404
 13281 U.S. PTO

Approved for use through 10/31/2002. OMB 0651-0032
 PTO/SB/05 (03-01)

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>	Attorney Docket No.	111325-291300
	First Inventor	Mai NGUYEN, et al.
	Title	SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS USING SHARED STATE VARIABLES
	Express Mail Label No.	

17497 U.S. PTO
 10/956121

100404

APPLICATION ELEMENTS	ADDRESS TO: Commissioner for Patents Box Patent Application Washington, DC 20231
----------------------	--

- See MPEP chapter 600 concerning utility patent application contents.
- Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original and a duplicate for fee processing)
 - Applicant claims small entity status.
 See 37 CFR 1.27.
 - Specification [Total Pages 33]
(preferred arrangement set forth below)
 - Descriptive title of the invention
 - Cross Reference to Related Applications *(if applicable)*
 - Statement Regarding Fed sponsored R & D *(if applicable)*
 - Reference to sequence listing, a table, or a computer program listing appendix *(if applicable)*
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings *(if filed)*
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
 - Drawing(s) (35 U.S.C. 113) 17 Figures [Total Sheets 14]
 - Oath or Declaration [Total Pages]
 - Newly executed (original or copy)
 - Copy from a prior application (37 CFR 1.63(d))
(for continuation/divisional with Box 18 completed)
 - DELETION OF INVENTOR(S)**
 Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b)
 - Application Data Sheet. See 37 CFR 1.76

- CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
- Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
 - Computer Readable Form (CRF)
 - Specification Sequence Listing on:
 - CD-ROM or CD-R (2 copies; or
 - paper
 - Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS	
9. <input type="checkbox"/> Assignment Papers (cover sheet & document(s))	
10. <input type="checkbox"/> 37 CFR 3.73(b) Statement <input type="checkbox"/> Power of Attorney <i>(when there is an assignee)</i>	
11. <input type="checkbox"/> English Translation Document <i>(if applicable)</i>	
12. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations	
13. <input type="checkbox"/> Preliminary Amendment	
14. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) <i>(Should be specifically itemized)</i>	
15. <input type="checkbox"/> Certified Copy of Priority Document(s) <i>(if foreign priority is claimed)</i>	
16. <input type="checkbox"/> Nonpublication request under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or its equivalent.	
17. <input type="checkbox"/> Other: _____	

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

Continuation Continuation-in-part (CIP) of prior application No: 10/162,701

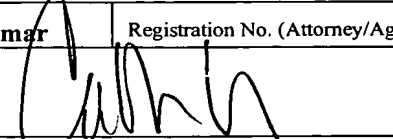
Prior application information: Examiner Not Yet Assigned Group / Art Unit: 2122

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS

Customer Number or Bar Code Label 22204 or Correspondence address below

Name			
Address			
City	State	Zip Code	
Country	Telephone	Fax	

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224
Signature		Date	October 4, 2004

100404

103281 U.S. PATENT AND TRADEMARK OFFICE

FEE TRANSMITTAL FOR FY 2004

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

Complete if Known	
Application Number	Not Yet Assigned
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Not Yet Assigned
Art Unit	Not Yet Assigned
Attorney Docket No.	111325-291300

TOTAL AMOUNT OF PAYMENT **\$1,078.00**

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:

Deposit Account Number: **19-2380**

Deposit Account Name: **Nixon Peabody LLP**

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments

Charge any additional fee(s)

Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1051	130	2051	65	Surcharge - late filing fee or oath
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet
1053	130	1053	130	Non-English specification
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action
1251	110	2251	55	Extension for reply within first month
1252	430	2252	215	Extension for reply within second month
1253	980	2253	490	Extension for reply within third month
1254	1,530	2254	765	Extension for reply within fourth month
1255	2,080	2255	1,040	Extension for reply within fifth month
1401	340	2401	170	Notice of Appeal
1402	340	2402	170	Filing a brief in support of an appeal
1403	340	2403	150	Request for oral hearing
1451	1,510	1451	1,510	Petition to institute a public use proceeding
1452	110	2452	55	Petition to revive - unavoidable
1453	1,370	2453	685	Petition to revive - unintentional
1501	1,370	2501	685	Utility issue fee (or reissue)
1502	490	2502	245	Design issue fee
1503	660	2503	330	Plant issue fee
1460	130	1460	130	Petitions to the Commissioner
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)
1806	180	1806	180	Submission of Information Disclosure Stmt
8021	40	8021	40	Recording each patent assignment per property (times number of properties)
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))
1801	790	2801	395	Request for Continued Examination (RCE)
1802	900	1802	900	Request for expedited examination of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$ 0)

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

Date

Signature

Typed or printed name

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	790.00
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) \$790.00

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Fee from below

Total Claims **36** -20** = **16** X **18.00** = **\$288.00**

Independent Claims **0** -3** = **0** X **0** = **0**

Multiple Dependent X **0** = **0**

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description
1202	18	2202	9	Claims in excess of 20
1201	88	2201	44	Independent claims in excess of 3
1203	300	2203	150	Multiple dependent claim, if not paid
1204	88	2204	44	** Reissue independent claims over original patent
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) \$288.00

**or number previously paid, if greater, For Reissues, see above

SUBMITTED BY

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224	Telephone	(202) 585-8204
Signature		Date	October 4, 2004		

Complete (if applicable)

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS USING
SHARED STATE VARIABLES**

RELATED APPLICATION DATA

[0001] This application is a continuation-in-part application of co-pending application Ser. No. 10/162,701 filed on June 6, 2002, which claims benefit from U.S. provisional applications Ser. Nos. 60/331,624, 60/331,623, and 60/331,621 filed on November 20, 2001, and U.S. provisional applications Ser. Nos. 60/296,113, 60/296,117, and 60/296,118 filed on June 7, 2001, the entire disclosures of all of which are hereby incorporated by reference herein.

FIELD OF THE INVENTION

[0002] The present invention generally relates to rights transfer and more particularly to a method, system and device for managing transfer of rights using shared state variables.

BACKGROUND OF THE INVENTION

[0003] One of the most important issues impeding the widespread distribution of digital works (i.e. documents or other content in forms readable by computers), via electronic means, and the Internet in particular, is the current lack of ability to enforce the intellectual property rights of content owners during the distribution and use of digital works. Efforts to resolve this problem have been termed "Intellectual Property Rights Management" ("IPRM"), "Digital Property Rights Management" ("DPRM"), "Intellectual Property Management" ("IPM"), "Rights Management" ("RM"), and "Electronic Copyright Management" ("ECM"), collectively referred to as "Digital Rights Management (DRM)" herein. There are a number of issues to be considered in effecting a DRM System. For example, authentication, authorization, accounting, payment and financial clearing, rights specification, rights verification, rights enforcement, and document protection issues should be addressed. U.S. patents 5,530,235, 5,634,012, 5,715,403, 5,638,443, and

5,629,980, the disclosures of which are incorporated herein by reference, disclose DRM systems addressing these issues.

[0004] Two basic DRM schemes have been employed, secure containers and trusted systems. A “secure container” (or simply an encrypted document) offers a way to keep document contents encrypted until a set of authorization conditions are met and some copyright terms are honored (e.g., payment for use). After the various conditions and terms are verified with the document provider, the document is released to the user in clear form. Commercial products such as CRYPTOLOPES™ and DIGIBOXES™ fall into this category. Clearly, the secure container approach provides a solution to protecting the document during delivery over insecure channels, but does not provide any mechanism to prevent legitimate users from obtaining the clear document and then using and redistributing it in violation of content owners’ intellectual property.

[0005] In the “trusted system” approach, the entire system is responsible for preventing unauthorized use and distribution of the document. Building a trusted system usually entails introducing new hardware such as a secure processor, secure storage and secure rendering devices. This also requires that all software applications that run on trusted systems be certified to be trusted. While building tamper-proof trusted systems is a real challenge to existing technologies, current market trends suggest that open and untrusted systems, such as PC’s and workstations using browsers to access the Web, will be the dominant systems used to access digital works. In this sense, existing computing environments such as PC’s and workstations equipped with popular operating systems (e.g., Windows™, Linux™, and UNIX) and rendering applications, such as browsers, are not trusted systems and cannot be made trusted without significantly altering their architectures. Of course, alteration of the architecture defeats a primary purpose of the Web, i.e. flexibility and compatibility.

[0006] As an example, U.S. patent 5,634,012, the disclosure of which is incorporated herein by reference, discloses a system for controlling the distribution of digital documents. Each rendering device has a repository associated therewith. A predetermined set of usage transaction steps define a protocol used by the repositories for enforcing usage rights. Usage rights define one or more manners of use of the associated document content and persist with the document content. The usage rights can permit various manners of use such as, viewing only, use once, distribution, and the like. Usage rights can be contingent on payment or other conditions. Further, a party may grant usage rights to others that are a subset of usage rights possessed by the party.

[0007] DRM systems have facilitated distribution of digital content by permitting the content owner to control use of the content. However, known business models for creating, distributing, and using digital content and other items involve a plurality of parties. For example, a content creator may sell content to a publisher who then authorizes a distributor to distribute content to an on-line storefront who then sells content to end-users. Further, the end users may desire to share or further distribute the content. In such a business model, usage rights can be given to each party in accordance with their role in the distribution chain. However, the parties do not have control over downstream parties unless they are privy to any transaction with the downstream parties in some way. For example, once the publisher noted above provides content to the distributor, the publisher cannot readily control rights granted to downstream parties, such as the first or subsequent users unless the publisher remains a party to the downstream transaction. This loss of control combined with the ever increasing complexity of distribution chains results in a situation, which hinders the distribution of digital content and other items. Further, the publisher may want to prohibit the distributor and/or the storefront from viewing or printing content while allowing an end user receiving a license from the storefront to view and print. Accordingly, the

concept of simply granting rights to others that are a subset of possessed rights is not adequate for multi-party, i.e. multi-tier, distribution models.

SUMMARY OF THE INVENTION

[0008] The exemplary embodiments of the present invention are directed to a method, system and device for transferring rights adapted to be associated with items from a rights supplier to a rights consumer, including obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived from the meta-; determining whether the rights consumer is entitled to the derivable rights specified by the meta-rights; and deriving at least one right from the derivable rights, if the rights consumer is entitled to the derivable rights specified by the meta-rights, wherein the derived right includes at least one state variable based on the set of rights and used for determining a state of the derived right.

[0009] Still other aspects, features, and advantages of the present invention are readily apparent from the following detailed description, simply by illustrating a number of exemplary embodiments and implementations, including the best mode contemplated for carrying out the present invention. The present invention is also capable of other and different embodiments, and its several details can be modified in various respects, all without departing from the spirit and scope of the present invention. Accordingly, the drawings and descriptions are to be regarded as illustrative in nature, and not as restrictive.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Exemplary embodiments of this invention will be described in detail, with reference to the attached drawings in which:

[0011] Fig. 1 is a schematic illustration of a rights management system in accordance with the preferred embodiment;

[0012] Fig. 2 is a block diagram of an example distribution chain showing the derivation of rights from meta-rights;

[0013] Fig. 3 is a schematic illustration of a license in accordance with the preferred embodiment;

[0014] Fig. 4 is an example of a license expressed with an XML based rights language in accordance with the preferred embodiment;

[0015] Fig. 5 is a block diagram of the license server of the system of Fig. 1;

[0016] Fig. 6 is a block diagram of a rights label in accordance with the preferred embodiment;

[0017] Fig. 7 is a flow chart of the procedure for transferring and deriving rights in accordance with the preferred embodiment;

[0018] Fig. 8 illustrates an exemplary system including a state-of-rights server;

[0019] Fig. 9 illustrates employing of a state variable in deriving exclusive usage rights;

[0020] Fig. 10 illustrates employing of a state variable in deriving inherited usage rights;

[0021] Fig. 11 illustrates employing of a state variable in deriving rights that are shared among a known set of rights recipients;

[0022] Fig. 12 illustrates employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients;

[0023] Fig. 13 illustrates employing of a state variable in maintaining a state shared by multiple rights;

[0024] Fig. 14 illustrates employing of multiple state variables to represent one state of rights;

[0025] Fig. 15 illustrates a case where not all rights are associated with states;

[0026] Fig. 16 illustrates a case where not all rights which are associated with states are shared or inherited; and

[0027] Fig. 17 illustrates a case of rights sharing based on an offer which does not explicitly include meta-rights.

DETAILED DESCRIPTION

[0028] A DRM system can be utilized to specify and enforce usage rights for specific content, services, or other items. Fig. 1 illustrates DRM System 10 that can be used in connection with the preferred embodiment. DRM System 10 includes a user activation component, in the form of activation server 20, that issues public and private key pairs to content users in a protected fashion, as is well known. During an activation process, some information is exchanged between activation server 20 and client environment 30, a computer or other device associated with a content recipient, and client component 60 is downloaded and installed in client environment 30. Client component 60 preferably is tamper resistant and contains the set of public and private keys issued by activation server 20 as well as other components, such as any component necessary for rendering content 42.

[0029] Rights label 40 is associated with content 42 and specifies usage rights and possibly corresponding conditions that can be selected by a content recipient. License Server 50 manages the encryption keys and issues licenses for protected content. These licenses embody the actual granting of usage rights to an end user. For example, rights label 40 may include usage rights permitting a recipient to view content for a fee of five dollars and view and print content for a fee of ten dollars. License 52 can be issued for the

view right when the five dollar fee has been paid, for example. Client component 60 interprets and enforces the rights that have been specified in license 52.

[0030] Fig. 6 illustrates rights label 40 in accordance with the preferred embodiment. Rights label 40 includes plural rights offers 44 each including usage rights 44a, conditions 44b, and content specification 44c. Content specification 44c can include any mechanism for calling, referencing, locating, linking or otherwise specifying content 42 associated with offer 44. Clear (unprotected) content can be prepared with document preparation application 72 installed on computer 70 associated with a content publisher, a content distributor, a content service provider, or any other party. Preparation of content consists of specifying the rights and conditions under which content 42 can be used, associating rights label 40 with content 42 and protecting content 42 with some crypto algorithm. A rights language such as XrML can be used to specify the rights and conditions. However, the rights can be specified in any manner. Also, the rights can be in the form of a pre-defined specification or template that is merely associated with the content. Accordingly, the process of specifying rights refers to any process for associating rights with content. Rights label 40 associated with content 42 and the encryption key used to encrypt the content can be transmitted to license server 50. As discussed in detail below, rights 44a can include usage rights, which specify a manner of use, and meta-rights, which permit other rights to be derived.

[0031] In some case, license 52 includes conditions that must be satisfied in order to exercise a specified right. For, example a condition may be the payment of a fee, submission of personal data, or any other requirement desired before permitting exercise of a manner of use. Conditions can also be "access conditions" for example, access conditions can apply to a particular group of users, say students in a university, or members of a book club. In other words, the condition is that the user is a particular person or

member of a particular group. Rights and conditions can exist as separate entities or can be combined.

[0032] Labels, offers, usage rights, and conditions can be stored together with content 42 or otherwise associated with content 42 through content specification 44c or any other mechanism. A rights language such as XrML can be used to specify the rights and conditions. However, the rights can be specified in any manner. Also, the rights can be in the form of a pre-defined specification or template that is merely associated with content 42.

[0033] A typical workflow for DRM system 10 is described below. A recipient operating within client environment 30 is activated for receiving content 42 by activation server 20. This results in a public-private key pair (and possibly some user/machine specific information) being downloaded to client environment 30 in the form of client software component 60 in a known manner. This activation process can be accomplished at any time prior to the issuing of a license.

[0034] When a recipient wishes to obtain specific content 42, the recipient makes a request for content 42. For example, a user, as a recipient, might browse a Web site running on Web server 80, using a browser installed in client environment 30, and request content 42. During this process, the user may go through a series of steps possibly including a fee transaction (as in the sale of content) or other transactions (such as collection of information). When the appropriate conditions and other prerequisites, such as the collection of a fee and verification that the user has been activated, are satisfied, Web server 80 contacts license server 50 through a secure communications channel, such as a channel using a Secure Sockets Layer (SSL). License server 50 then generates license 52 for content 42 and Web server 80 causes both the content and license 52 to be downloaded. License 52 includes the appropriate rights, such as usage rights and/or meta-rights, and can be downloaded from license server 50 or an associated device.

Content 42 can be downloaded from computer 70 associated with a vendor, distributor, or other party.

[0035] Client component 60 in client environment 30 will then proceed to interpret license 52 and allow use of content 42 based on the usage rights and conditions specified in license 52. The interpretation and enforcement of usage rights are well known generally and described in the patents referenced above, for example. The steps described above may take place sequentially or approximately simultaneously or in various orders.

[0036] DRM system 10 addresses security aspects of content 42. In particular, DRM system 10 may authenticate license 52 that has been issued by license server 50. One way to accomplish such authentication is for application 60 to determine if license 52 can be trusted. In other words, application 60 has the capability to verify and validate the cryptographic signature, or other identifying characteristic of license 52. Of course, the example above is merely one way to effect a DRM system. For example, license 52 and content 42 can be distributed from different entities. Clearinghouse 90 can be used to process payment transactions and verify payment prior to issuing a license.

[0037] As noted above, typical business models for distributing digital content include plural parties, such as owners, publishers, distributors, and users. Each of these parties can act as a supplier granting rights to a consumer downstream in the distribution channel. The preferred embodiment extends the known concepts of usage rights, such as the usage rights and related systems disclosed in U.S. patents 5,629,980, 5,634,012, 5,638,443, 5,715,403 and 5,630,235, to incorporate the concept of "meta-rights." Meta-rights are the rights that one has to generate, manipulate, modify, dispose of or otherwise derive other rights. Meta-rights can be thought of as usage rights to usage rights (or other meta-rights). This concept will become clear based on the description below.

[0038] Meta-rights can include derivable rights to offer rights, grant rights, negotiate rights, obtain rights, transfer rights, delegate rights, expose rights, archive rights, compile rights, track rights, surrender rights, exchange rights, and revoke rights to/from others. Meta-rights can include the rights to modify any of the conditions associated with other rights. For example, a meta-right may be the right to extend or reduce the scope of a particular right. A meta-right may also be the right to extend or reduce the validation period of a right. Meta-rights can be hierarchical and can be structured as objects within objects. For example, a distributor may have a meta-right permitting the distributor to grant a meta-right to a retailer which permits the retailer to grant users rights to view content. Just as rights can have conditions, meta-rights can also have conditions. Meta-rights can also be associated with other meta-rights.

[0039] The concept of meta-rights can be particularly useful because distribution models may include entities that are not creators or owners of digital content, but are in the business of manipulating the rights associated with the content. For example, as noted above, in a multi-tier content distribution model, intermediate entities (e.g., distributors) typically will not create or use the content but will be given the right to issue rights for the content they distribute. In other words, the distributor or reseller will need to obtain rights (meta-rights) to issue rights. For the sake of clarity, the party granting usage rights or meta-rights is referred to as “supplier” and the party receiving and/or exercising such rights is referred to as “consumer” herein. It will become clear that any party can be a supplier or a consumer depending on their relationship with the adjacent party in the distribution chain. Note that a consumer “consumes” , i.e. exercises, rights and does not necessarily consume, i.e. use, the associated content.

[0040] Fig. 2 schematically illustrates an example of a multi-tier distribution model 200. Publisher 210 publishes content for distribution, by distributor 220 for example. Distributor 220 distributes content to retailers, such as retailer

230 and retailer 230 sells content to users, such as user 240. In model 200, publisher 210 could negotiate business relationships with distributor 220 and distributor 220 could negotiate business relationships with retailer 230. Also, retailer 230 may desire usage rights that are beyond usage rights granted to distributor 220. However, keep in mind that, in a distribution chain that utilizes a DRM system to control use and distribution of content or other items, content can travel from publisher 210 to user 240 through any digital communication channel, such a network or transfer of physical media. When user 240 wishes to use content, a license is obtained, in the manner described above for example. Accordingly, the negotiated relationships can become difficult, if not impossible, to manage.

[0041] In model 200 of Fig. 2, retailer 230 will only grant rights to user 240 that have been predetermined and authorized by the distributor 220, publisher 210 and potentially other parties upstream of the transaction, such as the content creator or owner. The rights are predetermined through, and derived from, meta-rights granted to retailer 230 by distributor 220. Of course, there can be any number of parties in the distribution chain. For example, distributor 220 may sell directly to the public in which case retailer 230 is not necessary. Also, there may be additional parties. For example user 240 can distribute to other users.

[0042] In model 200 publisher grants to distributor 220 usage rights 212 permitting distribution of content, and meta-rights 214. Meta-rights 214 permit distributor 220 to grant to retailer 230 the usage right 214' (derived from meta-rights 214) to distribute or possibly sell content and meta-rights 216 which permit retailer 230 to grant user 240 the right to use content. For example, publisher 210 may specify, through meta-rights 214, that meta-right 216 granted to retailer 230 permits retailer 230 to grant only 500 licenses and usage rights 216' that retailer 230 can grant to a user can only be "view" and "print-once". In other words, distributor 220 has granted meta-rights to retailer 230. Similarly, publisher 210 issues meta-rights 214 to the distributor that will

govern what type, and how many , rights distributor 220 can grant to retailer 230. Note that these entities could be divisions, units or persons that are part of a larger enterprise, which also has other roles. For example, an enterprise might create, distribute, and sell content and carry out those activities using different personnel or different business units within the enterprise. The principles of meta-rights can be applied to an enterprise to determine content usage within that enterprise. Also, retailer 230 could grant meta-rights 218 to user 240 permitting user 240 to share rights or grant usage rights to achieve a super-distribution model. It can be seen that meta-rights of a party are derived from meta-rights granted by an upstream party in the distribution chain.

[0043] For example, a person's medical records can be in digital form managed by a first hospital as publisher 230. In this scenario, the person, as supplier, grants usage rights to the hospital, as consumer, to access and update the medical records. Should that person require treatment at a second hospital and desires to transfer their records to the second hospital, the person can grant to the first hospital the right to transfer the access rights to the new hospital through meta-rights. In other words, the person has specified meta-rights and granted the meta-rights to the first hospital. The meta-rights permit the first hospital to grant rights, as a supplier, to the second hospital, as a consumer. In another example, a person's last will and testament can be in digital form and managed by a law firm as publisher 210. If the person wishes to allow a third party to review the will. The person can grant meta-rights to the law firm permitting the law firm to grant access rights to this third party.

[0044] At a high level the process of enforcing and exercising meta-rights are the same as for usage rights. However, the difference between usage rights and meta-rights are the result from exercising the rights. When exercising usage rights, actions to content result. For example usage rights can be for viewing, printing, or copying digital content. When meta-rights are

exercised, new rights are created from the meta-rights or existing rights are disposed as the result of exercising the meta-rights. The recipient of the new rights may be the same principal (same person, entity, or machine, etc), who exercises the meta-rights. Alternatively, the recipient of meta-rights can be a new principal. The principals who receive the derived rights may be authenticated and authorized before receiving/storing the derived rights. Thus, the mechanism for exercising and enforcing a meta-right can be the same as that for a usage right. For example, the mechanism disclosed in U.S. Patent 5,634,012 can be used.

[0045] Meta-rights can be expressed by use of a grammar or rights language including data structures, symbols, elements, or sets of rules. For example, the XrML™ rights language can be used. As illustrated in Fig. 3, the structure of license 52 can consist of one or more grants 300 and one or more digital signatures 310. Each grant 300 includes specific granted meta-rights 302 such as rights to offer usage rights, grant usage rights, obtain usage rights, transfer usage rights, exchange usage rights, transport usage rights, surrender usage rights, revoke usage rights, reuse usage rights, or management meta-rights such as the rights to backup rights, restore rights, recover rights, reissue rights, or escrow the rights for management of meta-rights and the like.

[0046] Grant 300 can also specify one or more principals 304 to whom the specified meta-rights are granted. Also grants 300 can include conditions 306 and state variables 308. Like usage rights, access and exercise of the granted meta-rights are controlled by any related conditions 306 and state variables 308. The integrity of license 52 is ensured by the use of digital signature 310, or another identification mechanism. Signature 310 can include a crypto-algorithm, a key, or another mechanism for providing access to content 42 in a known manner. The structure of digital signature 310 includes the signature itself, the method of how the code is computed, the key information needed to verify the code and issuer identification.

[0047] State variables track potentially dynamic states conditions. State variables are variables having values that represent status of rights, or other dynamic conditions. State variables can be tracked, by clearinghouse 90 or another device, based on identification mechanisms in license 52. Further, the value of state variables can be used in a condition. For example, a usage right can be the right to print content 42 for and a condition can be that the usage right can be exercised three times. Each time the usage right is exercised, the value of the state variable is incremented. In this example, when the value of the state variable is three, the condition is no longer satisfied and content 42 cannot be printed. Another example of a state variable is time. A condition of license 52 may require that content 42 is printed within thirty days. A state variable can be used to track the expiration of thirty days. Further, the state of a right can be tracked as a collection of state variables. The collection of the change is the state of a usage right represents the usage history of that right.

[0048] Fig. 4 is an example of license 52 encoded in XrML™. The provider grants the distributor a meta right to issue a usage right (i.e., play) to the content (i.e., a book) to any end user. With this meta right, the distributor may issue the right to play the book within the U.S. region and subject to some additional conditions that the distributor may impose upon the user, as long as the distributor pays \$1 to the provider each time the distributor issues a license for an end user. The XrML™ specification is published and thus well known.

[0049] Fig. 5 illustrates the primary modules of license server 50 in accordance with the preferred embodiment. License interpreter module 502 validates and interprets license 52 and also provides the functions to query any or all fields in the license such as meta-rights 302, conditions 306, state variables 308, principle 304, and/or digital signature 310. License manager module 503 manages all license repositories for storing licenses 52, and also provides functions to create licenses 52 for derived rights, verify licenses,

store licenses, retrieve licenses and transfer licenses. State of rights module 504 manages the state and history of rights and meta-rights. The current value and history of the state variables together with the conditions controls the permission to exercise given meta-rights for a given authenticated principal. Condition validator 506 verifies conditions associated with the meta-rights. Together with the state variables, conditions associated with meta-rights define variables whose values may change over the lifetime of the meta-rights. Values of state variables used in conditions can affect the meta-rights at the time and during the time the rights are exercised.

[0050] Authorization module 508 authorizes the request to exercise meta-rights and to store the newly created rights or derived rights as the result of exercising the meta-rights. Authorization module 508 accesses both state of rights manager module 504 and condition validator module 506. Authorization module 508 interacts with license manager module 503 and the list of state variables and conditions and then passes the state variables to state of rights manager module 504 and condition list to condition validator module 506 for authorization.

[0051] A request for exercising a meta-right is passed to meta-rights manager module 510. Assuming that the requesting device has been authenticated, meta-rights manager module 510 requests the license manager module 504 to verify the license for exercising the requested meta-rights. License manager module 504 verifies the digital signature of the license and the key of the signer. If the key of the signer is trusted and the digital signature is verified then license manager module 504 returns "verified" to the meta-rights manager module 510. Otherwise "not verified" is returned.

[0052] Authorization module 508 instructs license manager 503 to fetch state variable 308 and conditions 306 of license 52. Authorization manager 508 then determines which state variables are required to enforce to enforce license 52. State of rights manager 504 then supplies the current value of

each required state variable to authorization module 508. Authorization module 508 then passes conditions 306 and the required state variables to condition validator 506. If all conditions 306 are satisfied, authorization module 508 returns “authorized” to meta-rights manager module 510.

[0053] Meta-rights manager module 510 verifies license 52 and meta-rights 302 therein, to authorize the request to exercise meta-rights 302, to derive new rights from meta-rights 302, and to update the state of rights and the current value of the conditions. Rights manager module 512, on the other hand, manages the new rights created or the derived rights as the result of exercising the meta-rights. Rights manager module 512 uses authorization module 508 to verify that recipient of the newly created rights or derived rights is intended principal 304. If the recipient are authorized then the rights manager module 512 directs license manager 504 to store the newly created rights in a repository associated with the consumer. This is discussed in greater detail below with reference to Fig. 7.

[0054] The authorization process is not limited to the sequence or steps described above. For example, a system could be programmed to allow authorization module 508 to request the state conditions from license manager 504 prior to verification of the digital signature. In such a case it would be possible to proceed subject to a verified license. Further, the various modules need not reside in the license server or related devices. The modules can be effected through hardware and/or software in any part of the system and can be combined or segregated in any manner.

[0055] Once a request to exercise a meta-rights has been authorized, the meta-right can be exercised. Meta-rights manager module 510 informs state of rights module 504 that it has started exercising the requested meta-rights. State of rights module 504 then records the usage history and changes its current value of the state variables. Meta-rights manager module 510 exercises the requested meta-rights in a manner similar to known procedures

for usage rights. If new rights are derived, then meta-rights manager module 510 invokes license manager module 504 to create new rights as the result of exercising the target meta-rights. Each new right is then sent to the corresponding rights manager module 512 of the consumer and stored in a repository associated with the consumer. Rights manager module 512 of the consumer will authenticate and authorize the consumer before receiving and storing the newly created right. New rights can be derived from meta-rights in accordance with a set of rules or other logic. For example, one rule can dictate that a consumed right to offer a license for use will result in the consumer having the right to offer a usage right and grant a license to that usage right to another consumer.

[0056] Fig. 7 illustrates the workflow for transferring meta-rights and deriving new rights from the meta-rights in accordance with the preferred embodiment. All steps on the left side of Fig. 7 relate to the supplier of rights and all steps on the right side of Fig. 7 relate to the consumer of rights. In step 702, principal 304 of license 52 is authenticated in a known manner. In other words, it is determined if the party exercising meta-right 302 has the appropriate license to do so. If the principal is not authorized, the procedure terminates in step 704. If the principal is authorized, the procedure advances to step 706 in which meta right 302 is exercised and transmitted to the consumer in the form of license 52 having derived rights in the manner set forth above. In step 708 the principal of this new license is authenticated. In other words, it is determined if the party exercising the derived rights has the appropriate license to do so. If the principal is not authorized, the procedure terminates in step 710. If the principal is authorized, the procedure advances to step 712 in which the derived right is stored. The procedure then returns to step 708 for each additional right in the license and terminates in step 714 when all rights have been processed.

[0057] Thus, the exemplary embodiments include a method for transferring rights adapted to be associated with items from a rights supplier to a rights

consumer, including obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived therefrom by the rights consumer, determining whether the rights consumer is entitled to derive the derivable rights specified by the meta-rights, and at least one of deriving the derivable rights, and generating a license including the derived rights with the rights consumer designated as a principal if the rights consumer is entitled to derive the derivable rights specified by the meta-rights. The exemplary embodiments further include a license associated with an item and adapted to be used within a system for managing the transfer of rights to the item from a rights supplier to a rights consumer. The license includes a set of rights including meta-rights specifying derivable rights that can be derived therefrom by the rights consumer, a principal designating at least one rights consumer who is authorized to derive the derivable rights, and a mechanism for providing access to the item in accordance with the set of rights. The exemplary embodiments still further include a method for deriving rights adapted to be associated with items from meta-rights, including obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived therefrom by the rights consumer, and generating a license associated with the item and including the derived rights.

[0058] Fig. 8 illustrates an exemplary system including a common state-of-rights server, according to the present invention. In FIG. 8, the exemplary system can include a common state-of-rights server of the system 801, including a state-of-rights manager 809, and one or more state-of-rights repositories 814, and one or more license servers 800, including a meta-rights manager 810, a usage rights manager 812, an authorization component 808, a condition validator 806, a state-of-rights manager 804, one or more state-of-rights repositories 816, a license manager 803, a license interpreter 802, and one or more license repositories 818.

[0059] The common state-of-rights server 801 can be configured as a remote server connected with one or more of the license servers 800. The common state-of-rights server 801 provides comparable services as the state-of-rights manager 804 in the license servers 800 via the state-of-rights manager 809. The services provided by the state-of-rights server 801 are accessible and states that the server 801 manages can be shared by one or more rights suppliers and rights consumers (not shown).

[0060] The state-of-rights server 801 can be configured as a remote server connected with one or more of the license servers 800 via one or more communication links 820, and the like. The services provided by the state-of-rights server 801 also can be integrated within one or more of the license server 800 and such services can be accessible by other rights suppliers, rights consumers, and the like.

[0061] The license manager 803 derives new rights based on an offer, which can include any suitable machine-readable expression, and optionally including meta-rights. While deriving rights, the license manager 803 can create new state variables to be associated with derived rights. The creation of state variables and their scopes can be prescribed in the offer or by some other function in the system. The state variables can be created in one or more instances, for example, prior to rights derivation, during rights derivation, upon fulfillment of conditions, during a first exercise of rights associated with the state variables, and the like. The state variables can be designated exclusively for a specific rights consumer, can be shared among rights consumers, and can be shared among rights consumers and other entities, such as rights suppliers, and the like. The license manager 803 can interact with the state-of-rights manager 804 to associate new state variables with physical addresses in one or more of the state-of-rights repositories 816. The state-of-rights manager 804 can access the one or more state-of-rights repositories 816 and can interact with the state-of-rights server 801 to access shared state variables from one or more of the state-of-rights repositories 814.

[0062] Designated state variables can be used to support a license that grants a recipient of the license a right to print content 5 times, shared state variables can be used to support a site license that grants a group of authorized users a right to print content an aggregated total of 100 times, and the like. A designated state variable can be updated when the corresponding right is exercised, whereas a shared state variable can be updated when an authorized user exercises the corresponding right. In other words, a shared state variable can include a data variable that is updated in response to actions by a plurality of users and which is globally applied to each of the users.

[0063] There are multiple ways to specify the scope of state variables, each of which can affect whether the derivative state variables can be shared, how the derivative state variables can be shared, and the like. For example, a state variable can be local, and solely confined to a recipient or can be global, and shared by a predetermined group of recipients. A global state variable can be shared by a group of recipients not determined when derived rights are issued, but to be specified later, perhaps based on certain rules defined in the license or based on other means. A global state variable can be shared between one or more rights suppliers, predetermined recipients, un-specified recipients, and the like. Advantageously, depending on the sharing employed with a given a business model and the rights granted in the meta-rights, state variables can be created at different stages of the value chain.

[0064] A set of non-exhaustive exemplary usages of state variables will now be described. For example, a state variable can be unspecified in meta-rights, which means the identifier and value of the state variable are yet to be determined by the meta-rights manager module 810 and included in the derived right. If a distinct state variable is assigned to each derived right, the scope of the state variable in the derived right is typically exclusive to the recipient.

[0065] Fig. 9 is used to illustrate employing of a state variable in deriving exclusive usage rights, according to the present invention. In Fig. 9, rights 902 and 903 derived from an offer 901 are exclusive to each respective consumer. The offer 901 is a type of meta-right of which the recipients have the rights to obtain specific derivative rights when the conditions for obtaining such rights are satisfied. Accordingly, the exemplary offer 901 has an unspecified state variable 904. However, specific state variable 905 and 906, each with uniquely assigned identifications (IDs) are included in the derived rights 902 and 903. The derived state variables 905 and 906 are bound to their associated derived rights, e.g., “AlicePlayEbook” (i.e., Alice has the right to play Ebook) is bound to derived right 902, and “BobPlayEbook” (i.e., Bob has the right to play Ebook) is bound to derived right 903. The “AlicePlayEbook” variable can be updated when Alice exercises her play right, whereas the “BobPlayEbook” variable can be updated when Bob exercises his play right.

[0066] Other than deriving rights from an offer, a right can transfer from an entity to a recipient. When a right is transferred, the governing of the associated state variable is also transferred to the recipient. After a right is transferred, the source principal typically can no longer exercise the right, whereas the recipient can exercise the right. The license server governing the exercising of a right of a recipient assumes the responsibility for state management. If, however, the state variables are managed by the common state of right server 801, the state of right server 801 needs to be informed of the transfer of right. Specifically, the state variable can be managed in the context of the recipient after the transfer of right.

[0067] When a right is to be shared between the source principal and the recipient, the associated state variable is referenced in the derived right. If the same right is shared with multiple recipients, then typically all of the recipients share the same state variables with the source principal. In this

case, a shared state can be managed by an entity that is accessible by all sharing principals.

[0068] Fig. 10 is used to illustrate employing of a state variable in deriving inherited usage rights, according to the present invention. In Fig. 10, a derived right can inherit a state variable from meta-rights. For example, a personal computer (PC) of a user, Alice, can be configured to play an e-book according to a license 1003. A personal data assistant (PDA) of Alice also can obtain a right to play the e-book according to offer 1001, if the PC and PDA share the same state variables 1004 and 1005, e.g., "AlicePlayEbook." A derived right 1002 allows Alice also to play the e-book on her PDA as long as the PDA and the PC share a same count limit 1006 of 5 times.

[0069] When a usage right is to be shared among a predetermined set of recipients, a state variable for tracking a corresponding usage right can be specified in a meta-right using a same state variable identification for all recipients. During a process of exercising the meta-right, the same state variable identification is included in every derived right.

[0070] Fig. 11 illustrates the use of state variable in deriving rights that are shared among a known set of rights recipients, according to the present invention. In Fig. 11, a site license 1101 is issued to FooU university. For example, via the site license 1101, a librarian is granted a right to issue rights that allow FooU students to play, view, and the like, corresponding content, such as e-books and the like, as long as such usage is tracked by a state variable 1104, e.g., "www.fooou.edu." Accordingly, rights 1102 and 1103 derived from the site license 1101 include state variables 1105 and 1106, "www.fooou.edu," which can be updated when corresponding students, Alice and Bob, play the e-book.

[0071] When a usage right is to be shared among a dynamic set of recipients, the state variable can stay unspecified in the usage right. When exercising a meta-right and a set of recipients is known, a state variable can

be specified using some identification unique to the known recipients and can be included within a derived right.

[0072] Fig. 12 is used to illustrate employing of a state variable in deriving rights that are shared among a dynamic set of rights recipients, according to the present invention. In Fig. 12, an offer 1201 specifies that a distributor can issue site licenses to affiliated clubs, allowing 5 members of each club to concurrently view, play, and the like, content, such as an e-book. A corresponding state variable 1207 associated with such a right can be unspecified in the offer 1201. When corresponding rights 1202 and 1203 are issued to affiliated clubs, the corresponding club identities are used to specify state variables 1208 and 1209 in the issued rights. The offers 1202 and 1203 are meta-rights derived from the offer 1201, with offer being assigned the distinct state variables 1208 and 1209. Further rights 1204-1206 can be derived from the offers 1202 and 1203 to be shared among members of each respective club. The licenses 1204 and 1205 are examples of rights derived from the offer 1202, and which inherit the state variable 1208, e.g., "urn:acme:club," whereas the license 1206 inherits the state variable 1209, e.g., "urn:foo:club."

[0073] Not only can state variables be shared among principals, such as rights suppliers, consumers, and the like, a state variable can be shared among multiple exercisable rights. Fig. 13 is used to illustrate employing of a state variable for maintaining a state shared by multiple rights, according to the present invention. In Fig. 13, a same state variable 1303 is associated to both a right to print 1302 and the right to play 1301, so that the total number of playing, printing, and the like, can be tracked together.

[0074] The state of rights can depend on more than one state variable. Fig. 14 is used to illustrate employing of multiple state variables to represent one state of rights, according to the present invention. The example described with respect to Fig. 14 builds upon the example described with

respect to Fig. 12. In FIG. 14, a usage right can be tracked by employing multiple state variables 1407 and 1408 in an offer 1401. The state variable 1408, for example, representing a priority level, can stay unspecified in the corresponding offers 1402 and 1403 (e.g., site licenses). The corresponding state variables 1409-1411, for example, used for setting a priority, can be assigned to each member in the corresponding licenses 1404, 1405 and 1406. The corresponding right to view, play, and the like, can now be dependent on two state variables, effectively restricting 5 simultaneous views, plays, and the like, per priority level.

[0075] One state variable can represent a collection of states. For example, a unique identification can be used to represent a state variable, and an appropriate mechanism can be employed to map such unique id to a database of multiple variables, where each variable represents a distinct state.

[0076] The scope of state variables can be used to determine entities by which the state variables can be managed. For example, for a local state variable, usage tracking of associated rights thereof can be managed solely by a trusted agent embedded within a rights consumption environment, such as a media player, and the like. In addition, such usage tracking can be conducted by a trusted remote service, such as the common state-of-rights server 801. Further, shared global state variables can be made accessible by multiple trusted agents. To avoid privacy issues, security issues, trust issues, rights issues, and the like, associated with accessing content, such as data, and the like, included within a peer rights consumption environment, managing of such shared global state variables can be performed by a remote service, such as the state-of-rights server 801.

[0077] A counter is a common form of state variable usage. For example, such state sharing can include counter sharing where a state represents a number of times a right has been exercised, an event has occurred, and the

like. Such counter sharing can be manifested in various forms and occur in many contexts, such as: tracking a number of simultaneous uses, tracking a number of sequential uses, sequencing (e.g., a commercial must be viewed before free content can be accessed), a one-time use constraint, a transaction count, a delegation control level, a super-distribution level, dependency on at least one or more services or devices, and the like.

[0078] In addition, state variables can be incarnated in a wide variety of forms. For example, a state variable can be used to track specific time slots within a period of time, such as used by a movie studio to transfer syndication rights to a specific TV station, to transfer syndication rights shared by a group of stations, to transfer syndication rights assigned through a bidding process, and the like.

[0079] State variables also can be employed, for example, with regional selling or distribution rights, in a statement from a financial clearing house to acknowledge that an appropriate fee has been paid, as a status of whether a commercial has been watched before free content can be accessed, and the like.

[0080] Not all rights need be associated with states. Fig. 15 is used to illustrate a case where not all rights are associated with states, according to the present invention. In Fig. 15, an offer 1501 allows a user, Alice, to grant an unlimited play right, view right, and the like, to her PDA. Such a play right need not be associated with any state. Accordingly, derived right 1502 also has an unlimited play right to the content, as well as the right 1503 for her PC.

[0081] Not all rights which are associated with states are shared or inherited. For example, some rights are meant for off-line usage, can be transferred in whole to another device, and hence are not shared with other devices. Fig. 16 is used to illustrate a case where not all rights which are associated with states are shared or inherited, according to the present invention. In Fig. 16, even though a play right 1603 of a user, Alice, a play

right 1602 of a PDA of Alice, and a play right 1603 of a PC of Alice specify a same state variable identification 1604, a same state need not be shared since each device can track a state thereof locally. Advantageously, such an implementation would allow the PC and the PDA to each play the corresponding content up to 5 times.

[0082] Fig. 17 illustrates a form of an offer which does not explicitly include meta-rights. In Fig. 17, an offer 1701 is configured as a site license written in English. Licenses 1702 and 1703 are instances derived from the offer 1701. In an exemplary embodiment, variables 1704 and 1705 can be created based on interpretation of the offer 1701, for example, by the system of Fig. 8.

[0083] The preferred embodiments are not limited to situations where resellers, distributors or other “middlemen” are used. For example, the preferred embodiment can be applied within enterprises or other organizations, which create and/or distribute digital content or other items to control use of the content within the enterprise or other organization. Meta-rights can also be issued to end-users, when the grant of a right relates to another right. For example, the right to buy or sell securities as it is in the case of trading options and futures. Meta-rights can be assigned or associated with goods services, resources, or other items.

[0084] The invention can be implemented through any type of devices, such as computers and computer systems. The preferred embodiment is implemented in a client server environment. However, the invention can be implemented on a single computer or other device. Over a network using dumb terminals, thin clients, or the like, or through any configuration of devices. The various modules of the preferred embodiment have been segregated and described by function for clarity. However, the various functions can be accomplished in any manner through hardware and/or software. The various modules and components of the preferred embodiment have separate utility and can exist as distinct entities. Various communication

channels can be used with the invention. For example, the Internet or other network can be used. Also, data can be transferred by moving media, such as a CD, DVD, memory stick or the like, between devices. Devices can include, personal computers, workstations, thin clients, PDA's and the like.

[0085] The invention has been described through exemplary embodiments and examples. However, various modifications can be made without departing from the scope of the invention as defined by the appended claims and legal equivalents.

What is claimed is:

1. A method for transferring rights adapted to be associated with items from a rights supplier to a rights consumer, the method comprising:

obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived from the meta-rights;

determining whether the rights consumer is entitled to the derivable rights specified by the meta-rights; and

deriving at least one right from the derivable rights, if the rights consumer is entitled to the derivable rights specified by the meta-rights, wherein the derived right includes at least one state variable based on the set of rights and used for determining a state of the derived right.

2. The method of claim 1, wherein the state variable inherits a state thereof for content usage or rights transfer from the set of rights.

3. The method of claim 1, wherein the state variable shares a state thereof for content usage or rights transfer with the set of rights.

4. The method of claim 1, wherein the state variable inherits a remaining state for content usage or rights transfer from the set of rights.

5. The method of claim 1, wherein the state variable is updated upon exercise of a right associated with the state variable.

6. The method of claim 1, further comprising deriving a plurality of rights from the derivable rights, wherein the state variable is shared by the derived rights.

7. The method of claim 1, wherein the state variable represents a collection of states.

8. The method of claim 1, further comprising a plurality of state variables that determine the state of the derived right.

9. The method of claim 1, wherein the at least one state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

10. The method of claim 1, wherein the state variable is transferred from the derivable rights to the derived right.

11. The method of claim 1, further comprising generating a license including the derived right, if the rights consumer is entitled to the derivable rights specified by the meta-rights.

12. A system for transferring rights adapted to be associated with items from a rights supplier to a rights consumer, the system comprising:

means for obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived from the meta-rights;

means for determining whether the rights consumer is entitled to the derivable rights specified by the meta-rights; and

means for deriving at least one right from the derivable rights, if the rights consumer is entitled to the derivable rights specified by the meta-rights, wherein the derived right includes at least one state variable based on the set of rights and used for determining a state of the derived right.

13. The system of claim 12, wherein the state variable inherits a state thereof for content usage or rights transfer from the set of rights.

14. The system of claim 12, wherein the state variable shares a state thereof for content usage or rights transfer with the set of rights.

15. The system of claim 12, wherein the state variable inherits a remaining state for content usage or rights transfer from the set of rights.

16. The system of claim 12, wherein the state variable is updated upon exercise of a right associated with the state variable.

17. The system of claim 12, further comprising means for deriving a plurality of rights from the derivable rights, wherein the state variable is shared by the derived rights.

18. The system of claim 12, wherein the state variable represents a collection of states.

19. The system of claim 12, including a plurality of state variables that determine the state of the derived right.

20. The system of claim 12, wherein the at least one state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

21. The system of claim 12, wherein the state variable is transferred from the derivable rights to the derived right.

22. The system of claim 12, further comprising means for generating a license including the derived right, if the rights consumer is entitled to the derivable rights specified by the meta-rights.

23. The system of claim 12, wherein the means for obtaining, the means for determining, and the means for deriving comprise at least one of computer-executable instructions, and devices of a computer system.

24. A device for transferring rights adapted to be associated with items from a rights supplier to a rights consumer, the device comprising:

means for obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived from the meta-rights;

means for determining whether the rights consumer is entitled to the derivable rights specified by the meta-rights; and

means for deriving at least one right from the derivable rights, if the rights consumer is entitled to the derivable rights specified by the meta-rights, wherein the derived right includes at least one state variable based on the set of rights and used for determining a state of the derived right.

25. The device of claim 24, wherein the state variable inherits a state thereof for content usage or rights transfer from the set of rights.

26. The device of claim 24, wherein the state variable shares a state thereof for content usage or rights transfer with the set of rights.

27. The device of claim 24, wherein the state variable inherits a remaining state for content usage or rights transfer from the set of rights.

28. The device of claim 24, wherein the state variable is updated upon exercise of a right associated with the state variable.

29. The device of claim 24, further comprising means for deriving a plurality of rights from the derivable rights, wherein the state variable is shared by the derived rights.

30. The device of claim 24, wherein the state variable represents a collection of states.

31. The device of claim 24, including a plurality of state variables that determine the state of the derived right.

32. The device of claim 24, wherein the at least one state variable is unspecified in the derived right, is created during a rights transfer, and is assigned to the derived right.

33. The device of claim 24, wherein the state variable is transferred from the derivable rights to the derived right.

34. The device of claim 24, further comprising means for generating a license including the derived right, if the rights consumer is entitled to the derivable rights specified by the meta-rights.

35. The device of claim 24, wherein the means for obtaining, the means for determining, and the means for deriving comprise at least one of computer-executable instructions, and devices of a computer system.

36. The device of claim 24, wherein one or more of the means for obtaining, the means for determining, and the means for deriving are specified in a license.

ABSTRACT OF THE DISCLOSURE

[0086] A method, system and device for transferring rights adapted to be associated with items from a rights supplier to a rights consumer, including obtaining a set of rights associated with an item, the set of rights including meta-rights specifying derivable rights that can be derived from the meta-; determining whether the rights consumer is entitled to the derivable rights specified by the meta-rights; and deriving at least one right from the derivable rights, if the rights consumer is entitled to the derivable rights specified by the meta-rights, wherein the derived right includes at least one state variable based on the set of rights and used for determining a state of the derived right.

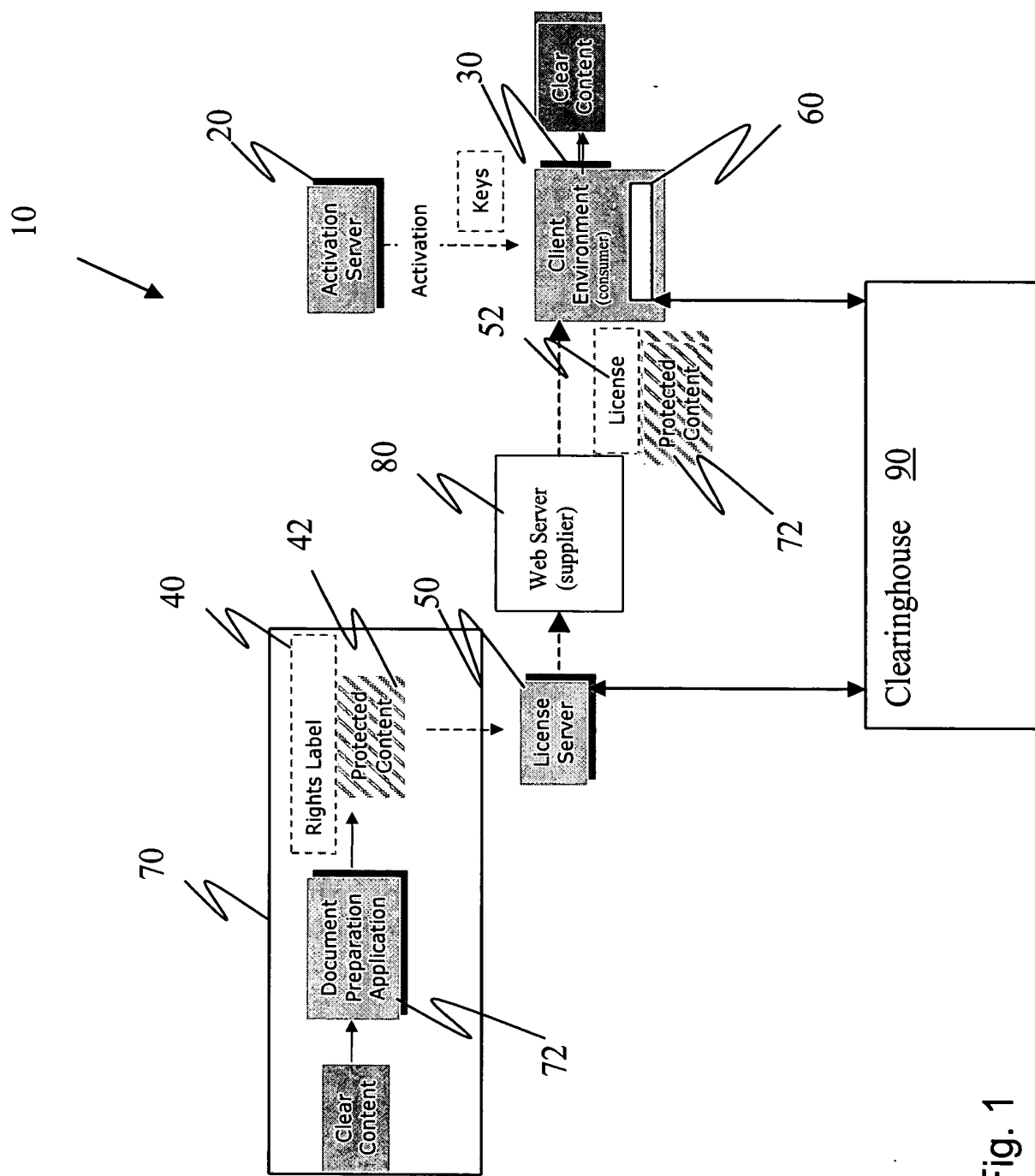


Fig. 1

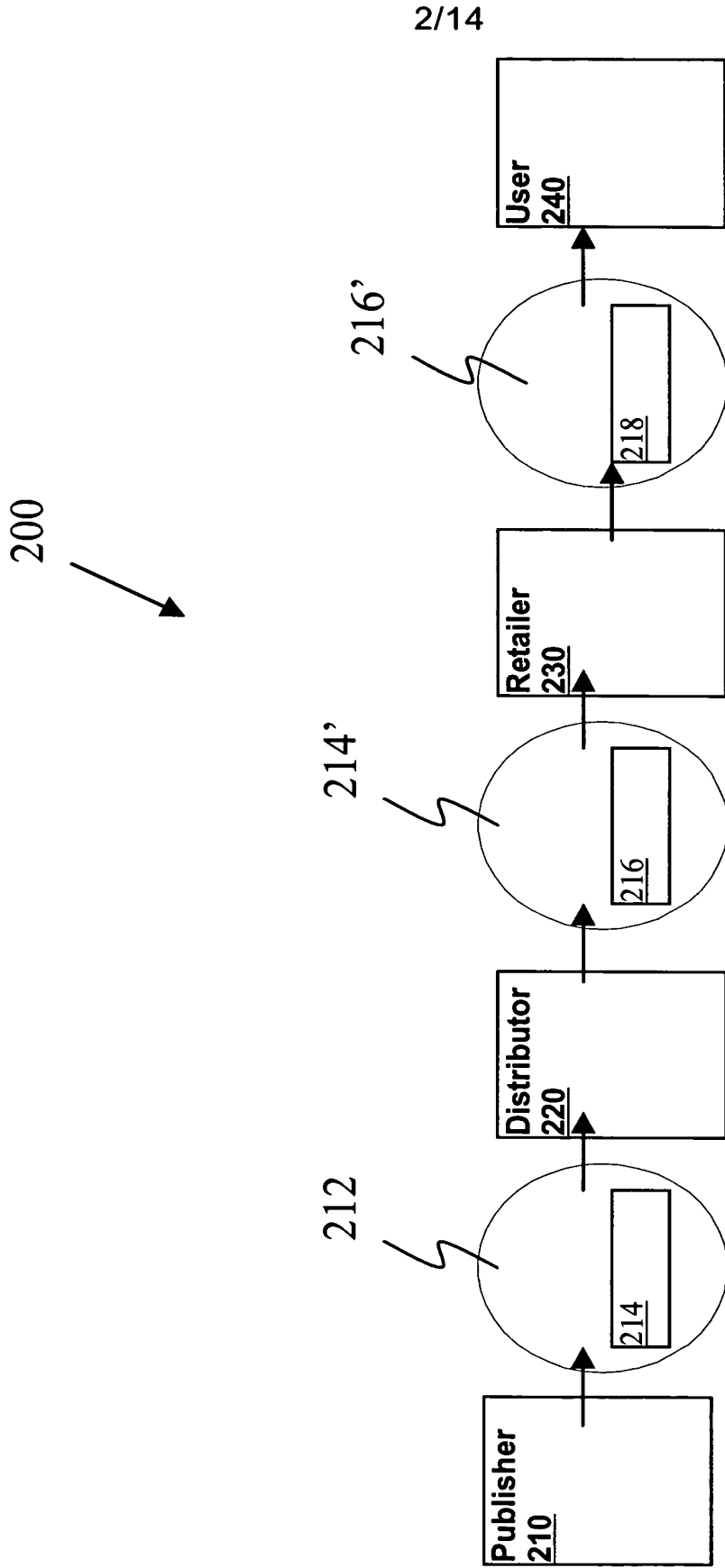


Fig. 2

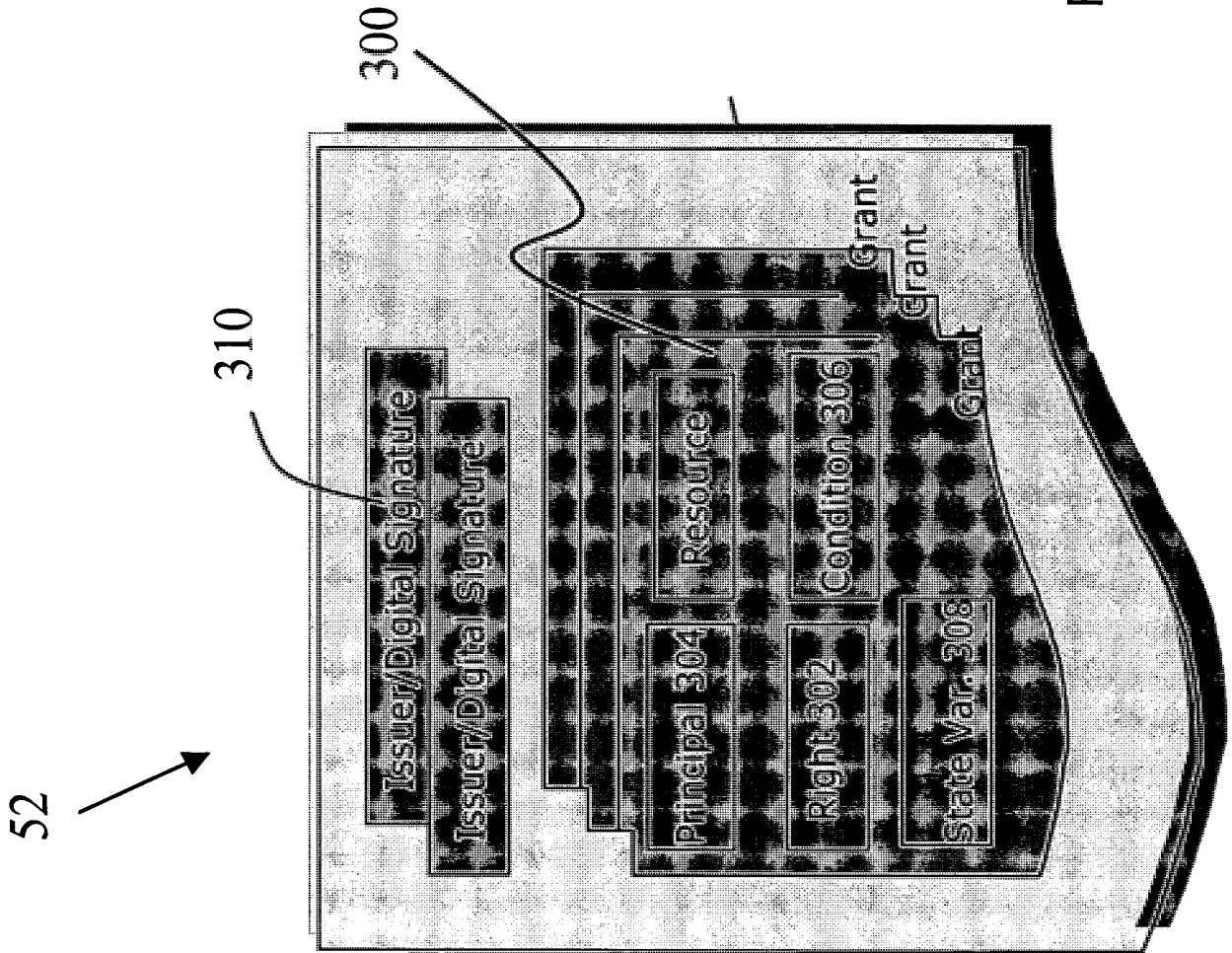


Fig. 3

52

```
</license>
<grant>
  <forAll varName="user"/>
  <forAll varName="distributorConditionForPlay"/>
  <principal id="distributor"/>
  <issue/>
  <grant>
    <principal varRef="user"/>
    <play/>
    <digitalResource licensePartId="book"/>
    <allCondition>
      <region regionCode="US"/>
      <condition varRef="distributorConditionForPlay"/>
    </allCondition>
  </grant>
  <fee>
    <flat currencyCode="USD">1</flat>
    <to licensePartId="provider"/>
  </fee>
  </grant>
  <issuer id="provider"/>
</license>
```

Fig. 4

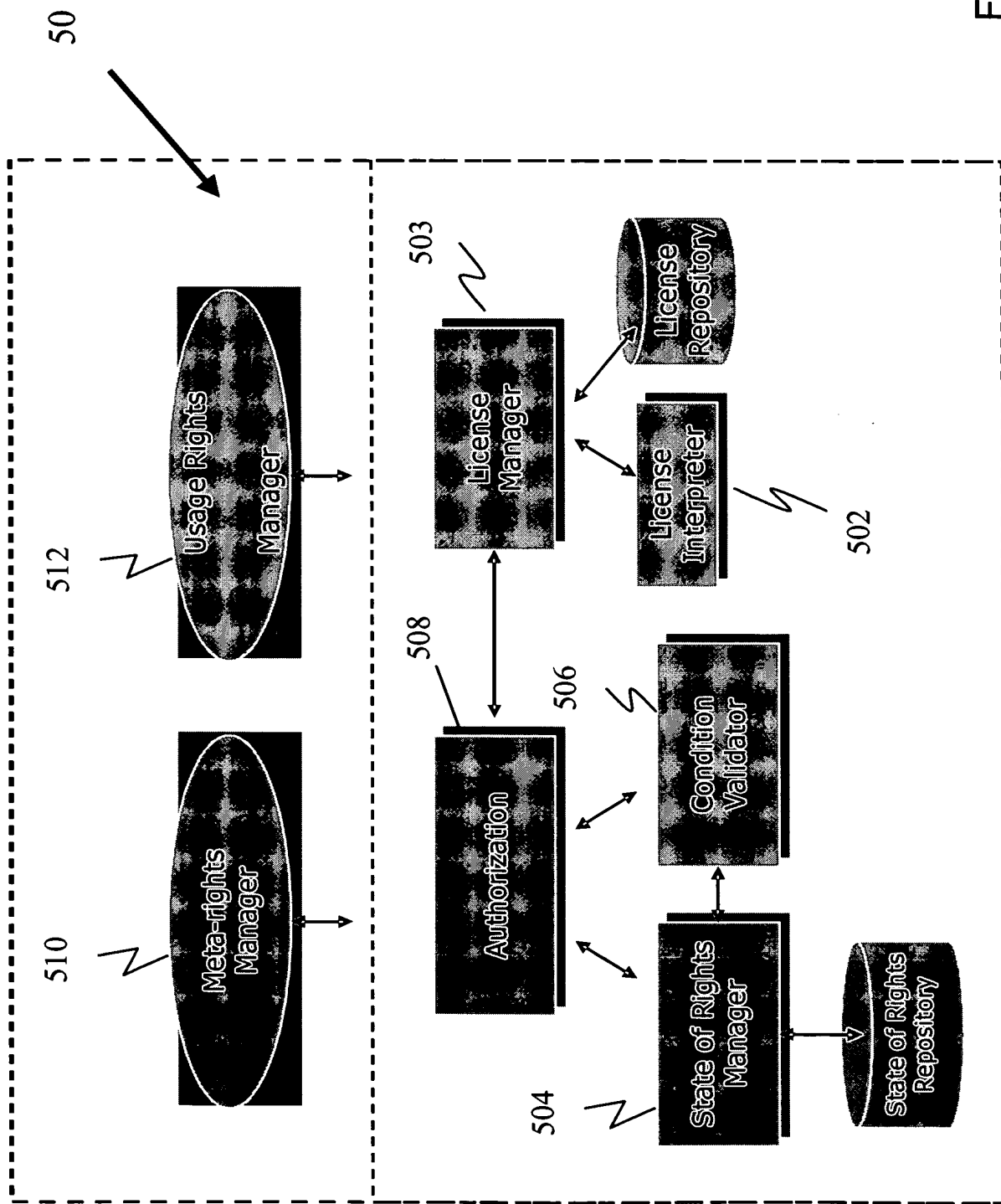


Fig. 5

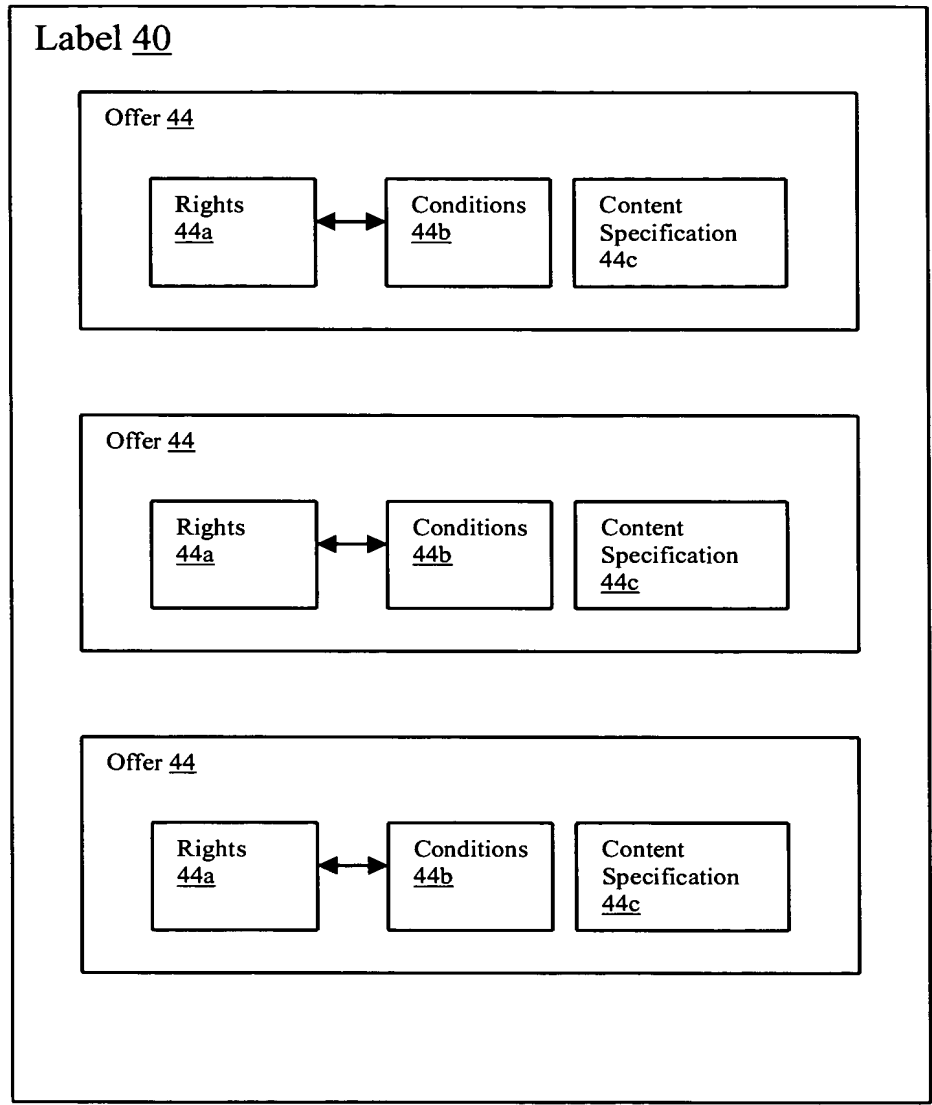


Fig. 6

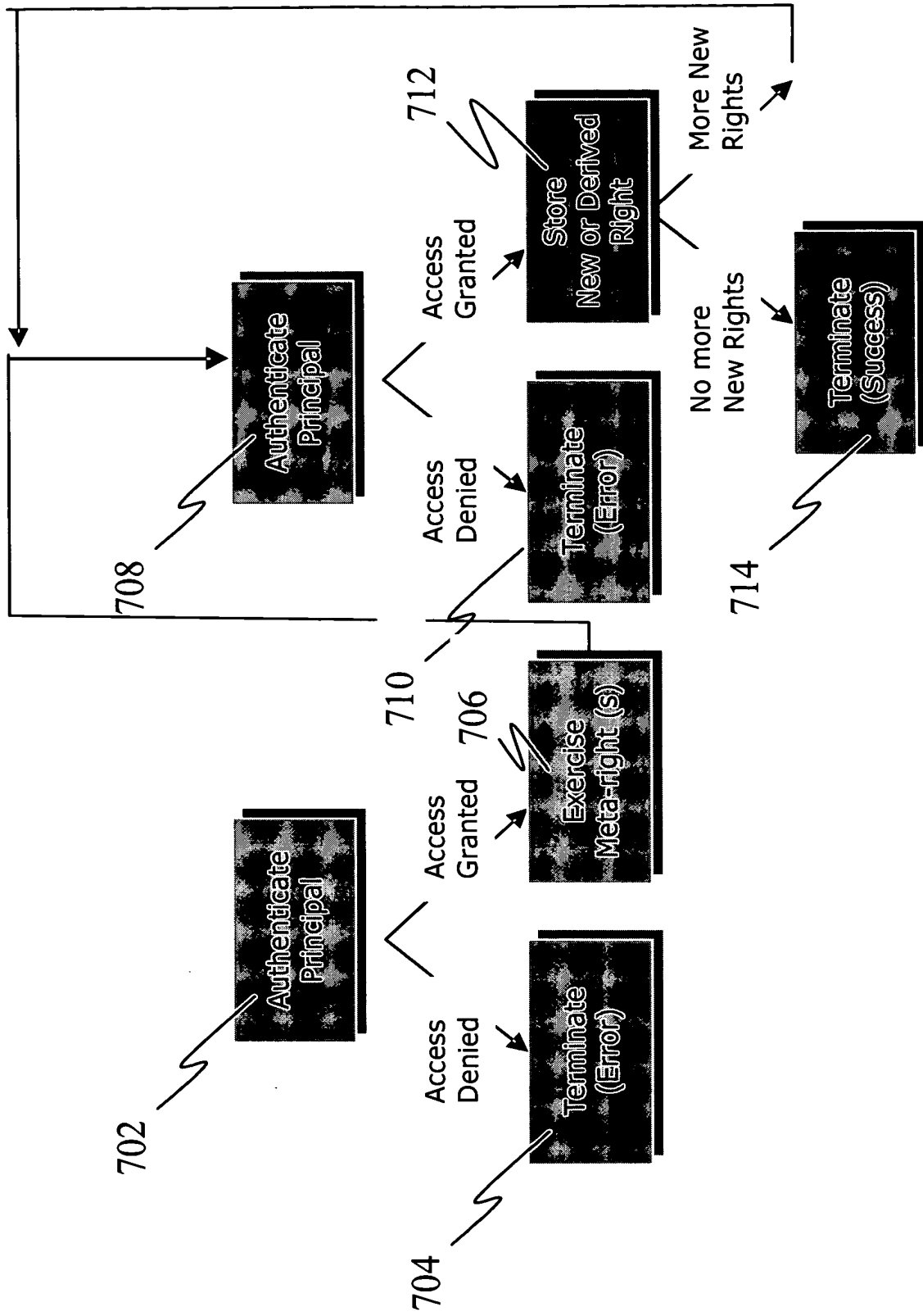


Fig. 7

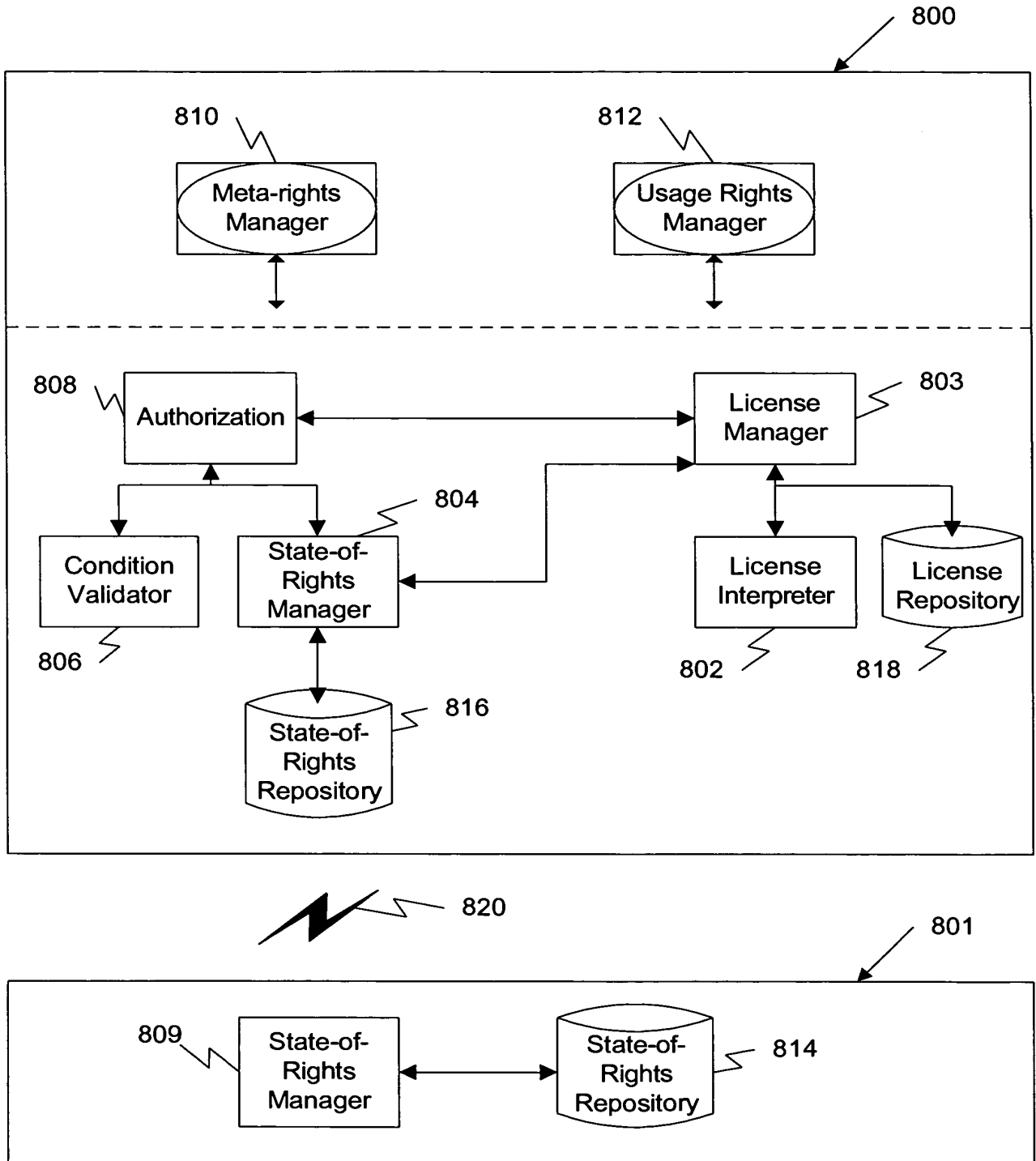


Fig. 8

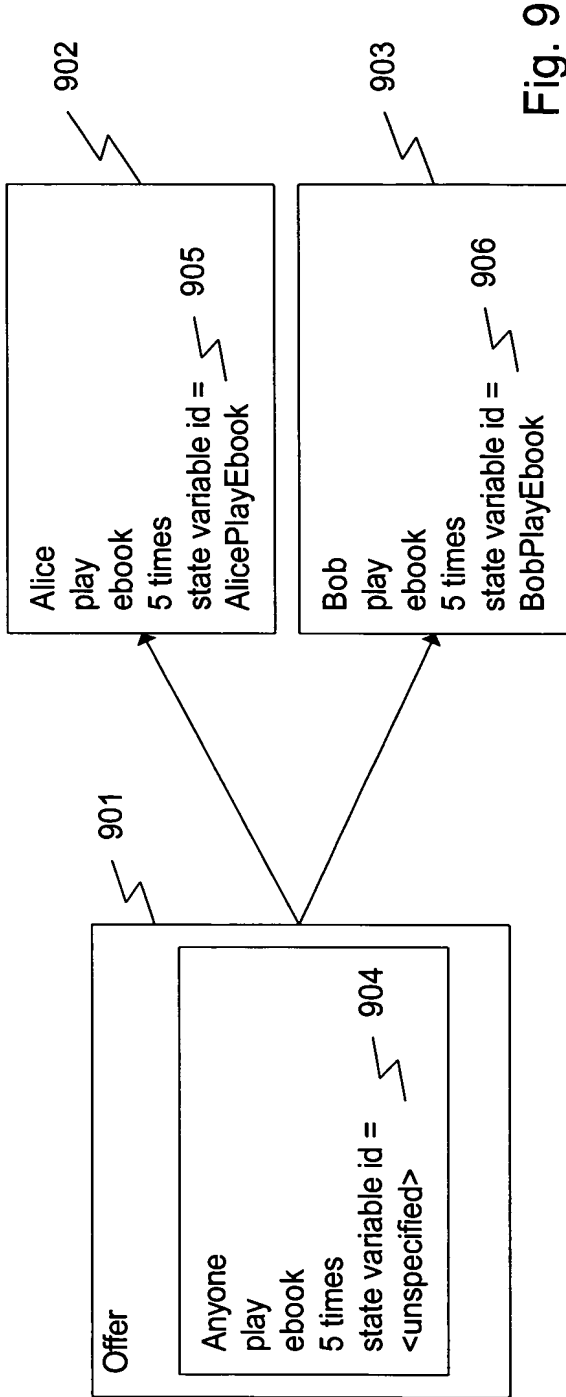


Fig. 9

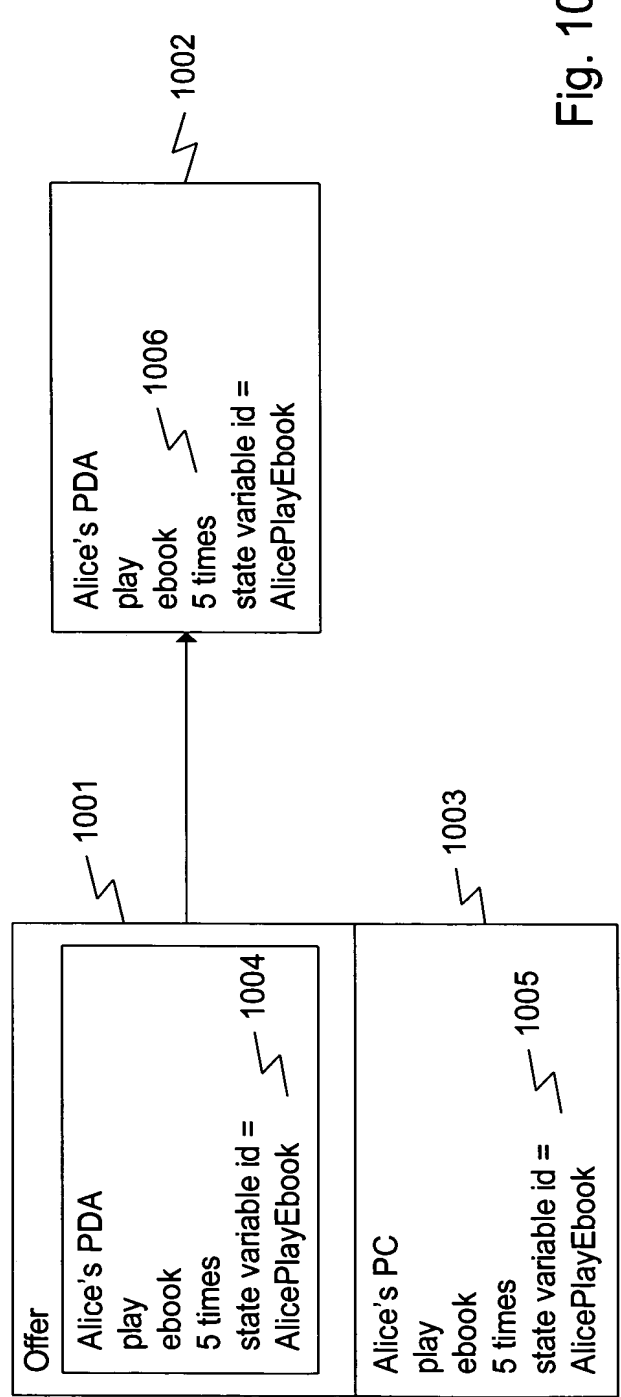


Fig. 10

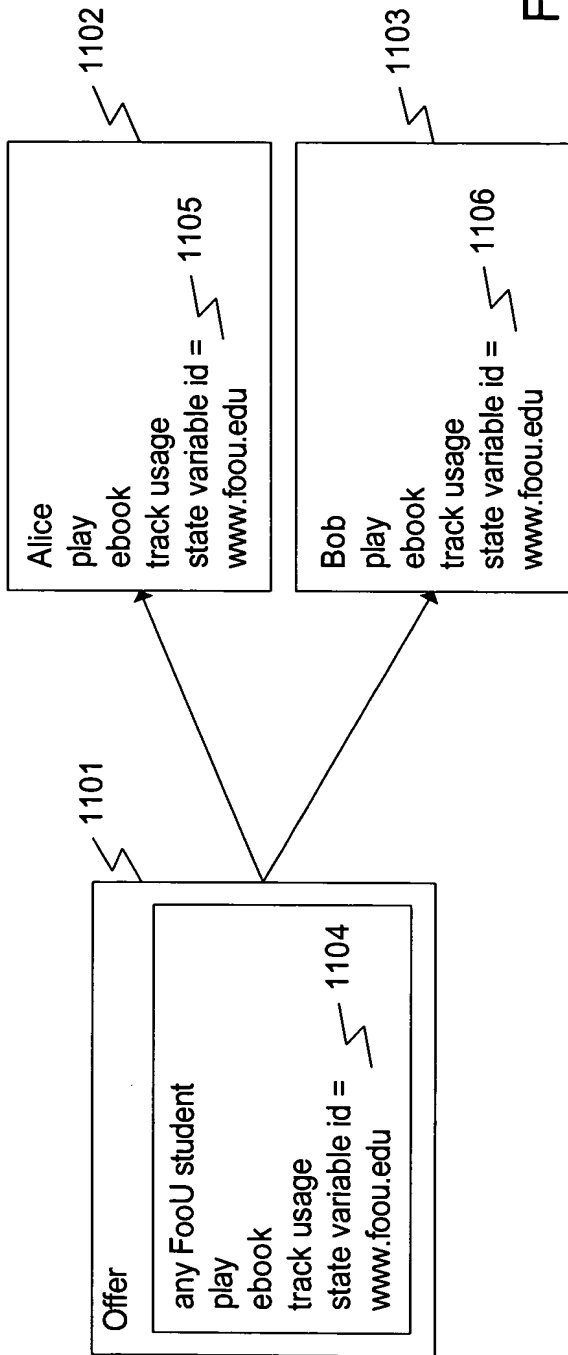


Fig. 11

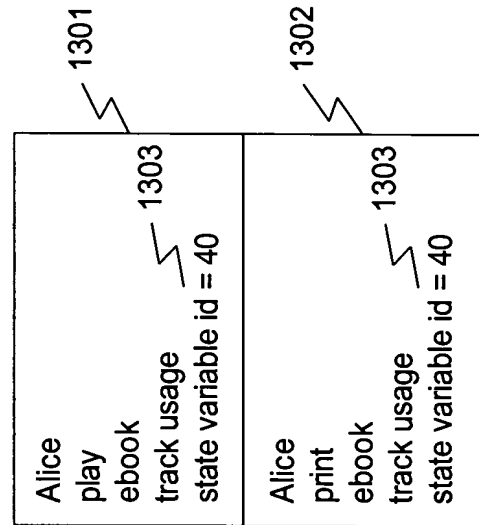


Fig. 13

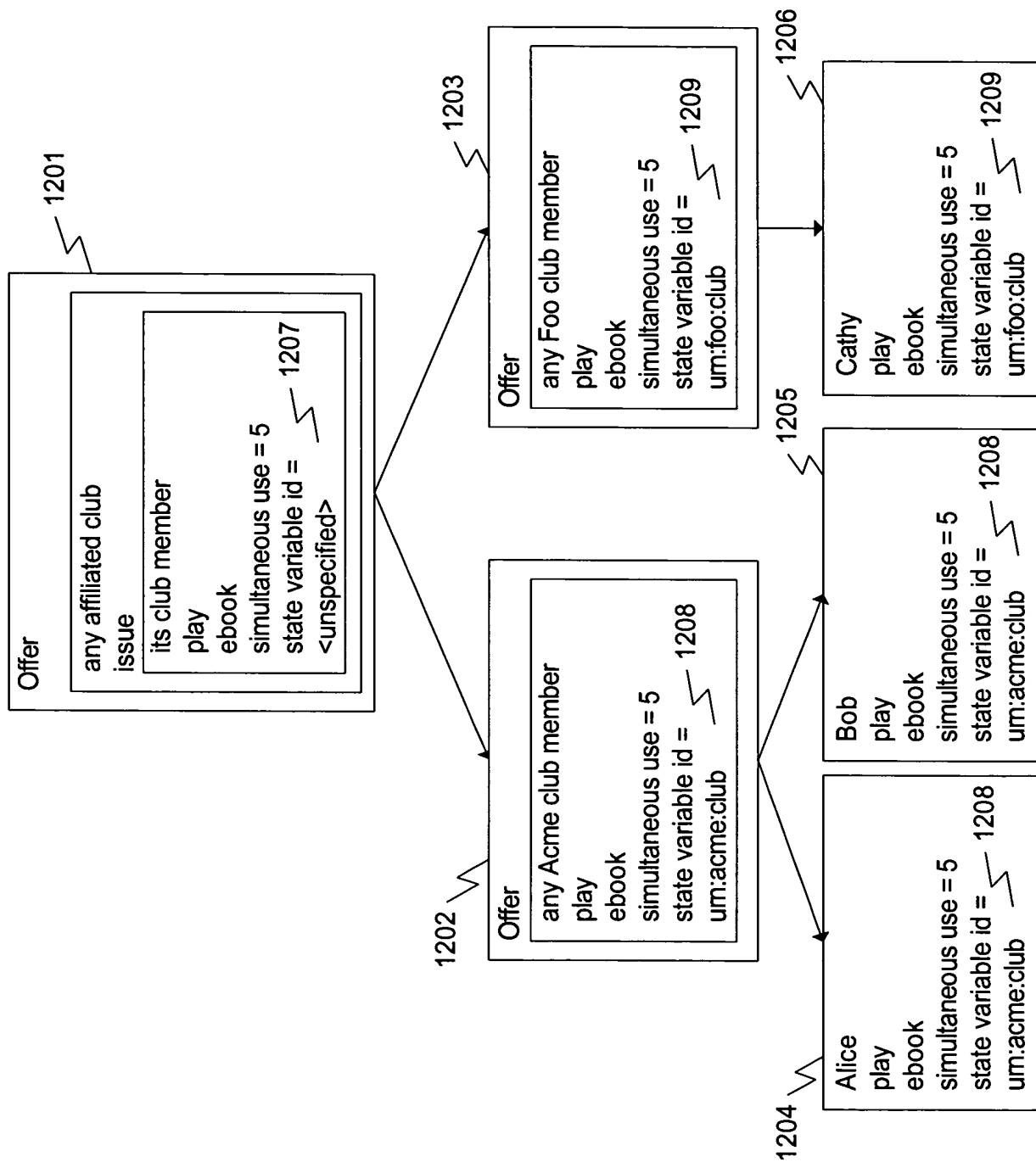


Fig. 12

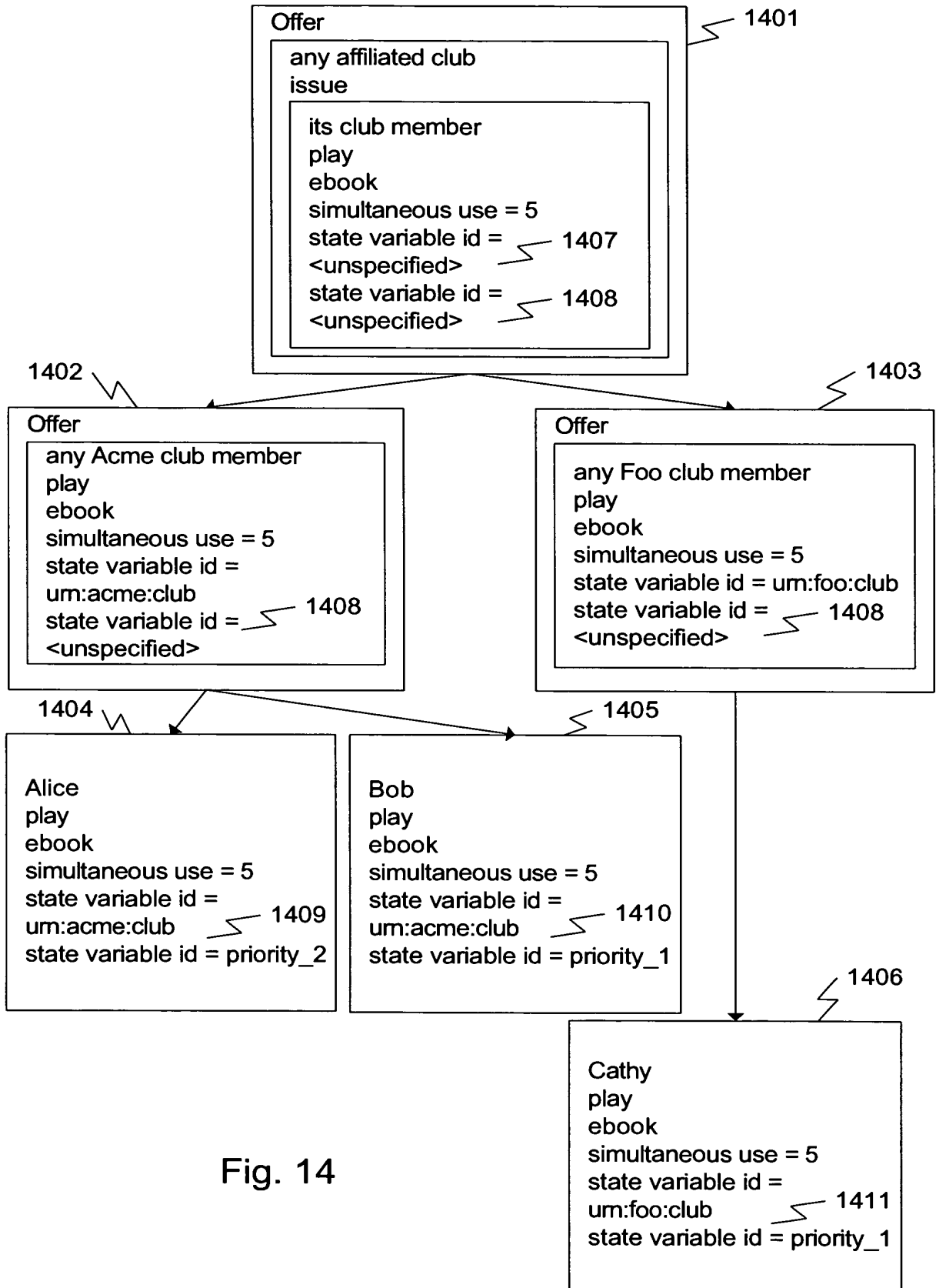


Fig. 14

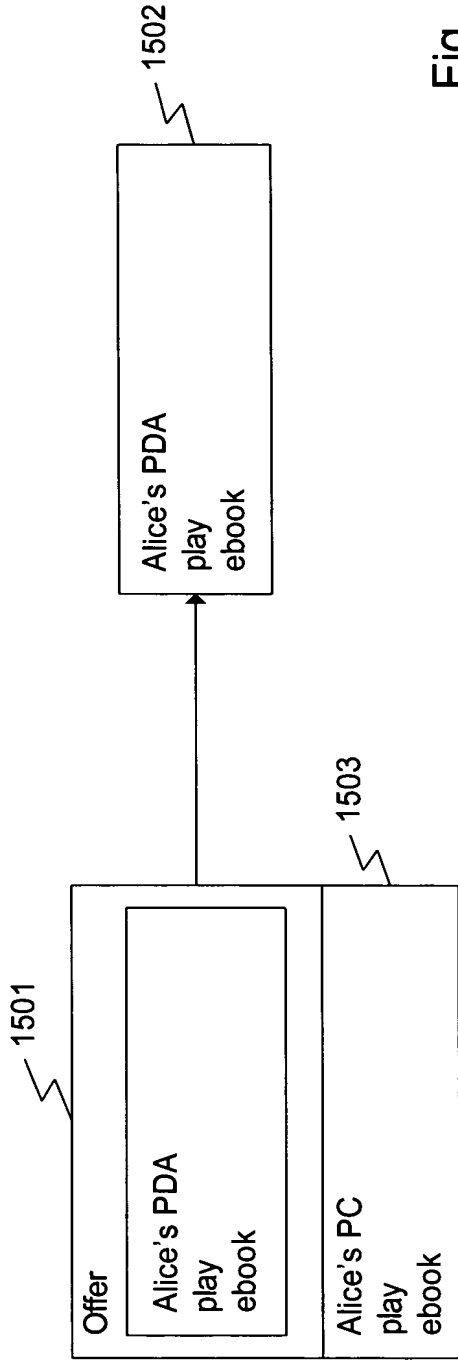


Fig. 15

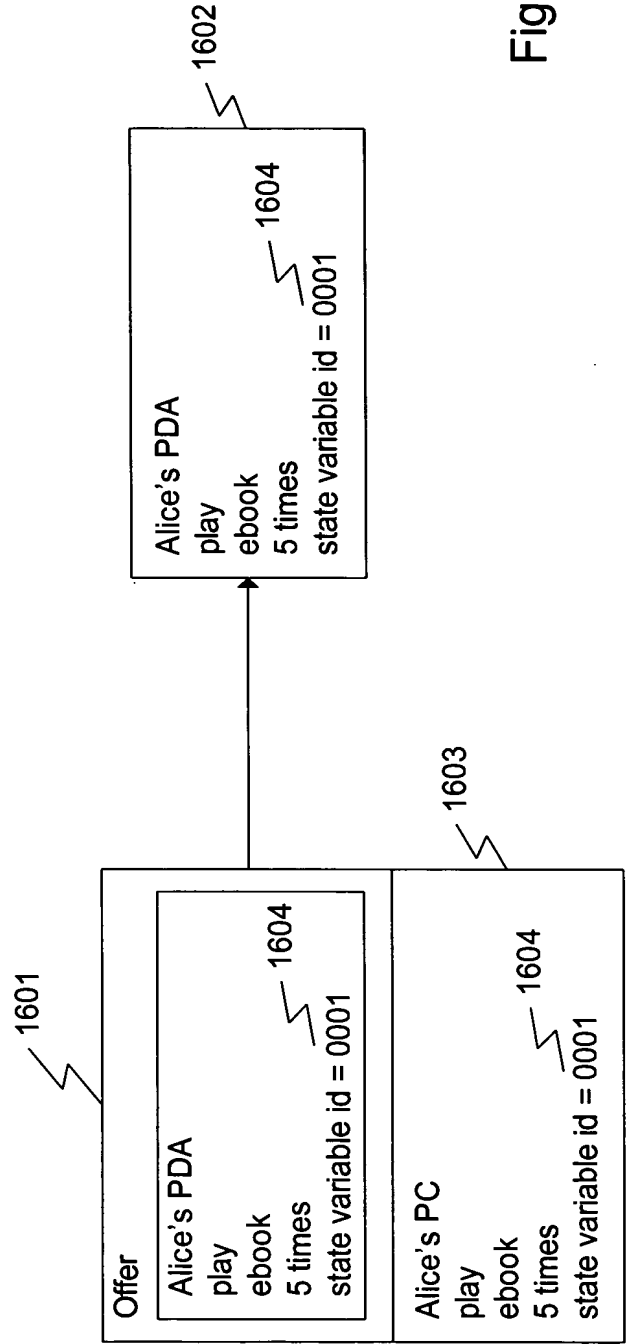


Fig. 16

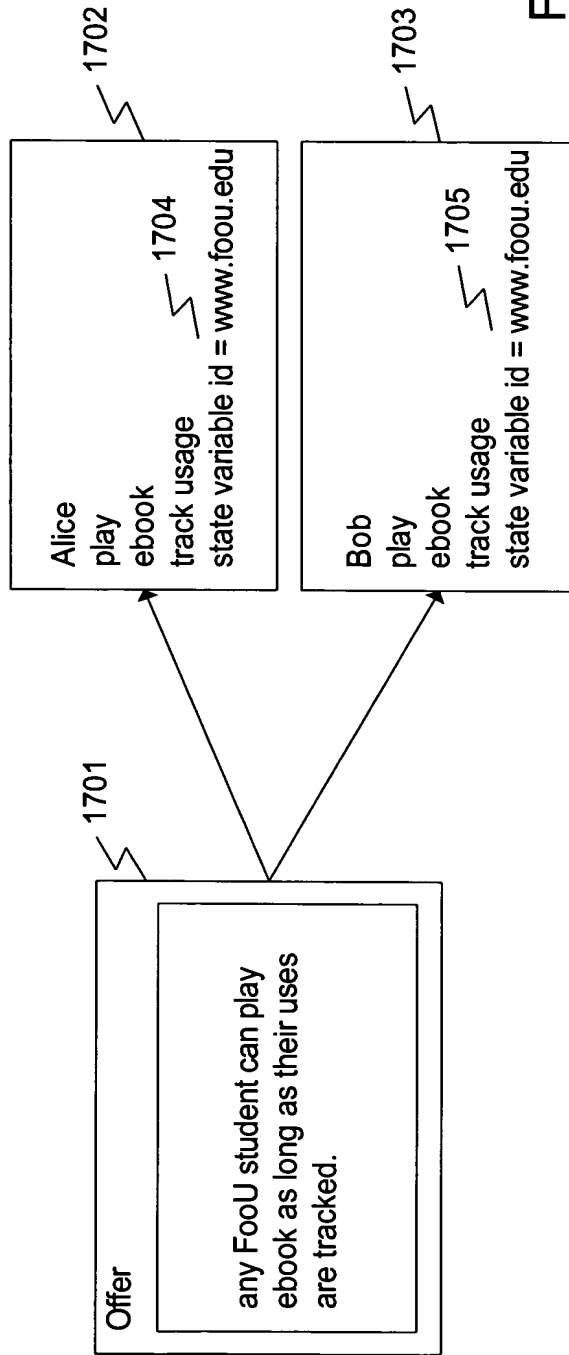


Fig. 17

Please type a plus sign (+) inside this box → [+]

PTO/SB/01 (12-97)

Approved for use through 9/30/00. OMB 0651-0032
Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

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

DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION (37 CFR 1.63)		Attorney Docket Number	111325-113	
		First Named Inventor	Xin Wang et al.	
<input type="checkbox"/> Declaration Submitted With Initial Filing OR <input checked="" type="checkbox"/> Declaration Submitted after Initial Filing (surcharge (37 CFR 1.16(d)) required)		COMPLETE IF KNOWN		
		Application Number	10/162,701	
		Filing Date	June 6, 2002	
		Group Art Unit	2122	
		Examiner Name	Not yet assigned	
		<p>As a below named inventor, I hereby declare that: My residence, post office address, and citizenship are as stated below next to my name. I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled: <u>Method And Apparatus Managing The Transfer Of Rights</u> <i>(Title of the Invention)</i> the specification of which <input type="checkbox"/> is attached hereto OR <input checked="" type="checkbox"/> was filed on (MM/DD/YYYY) <u>06/06/2002</u> As United States Application Number or PCT International Application Number <u>10/162,701</u> And was amended on (MM/DD/YYYY) _____ (If applicable). I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment specifically referred to above. I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56.</p>		
I hereby claim foreign priority benefits under 35 U.S.C. 119(a)-(d) or 365(b) of any foreign application(s) for patent or inventor's certificate, or 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below, by checking the box, any foreign application for patent or inventor's certificate, or of any PCT international application having a filing date before that of the application on which priority is claimed.				
Prior Foreign Application Number(s)	Country	Foreign Filing Date (MM/DD/YYYY)	Priority Not Claimed	Certified Copy Attached YES No
			<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<input type="checkbox"/> Additional foreign application numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.				
I hereby claim the benefit under 35 U.S.C. 119(e) of any United States provisional application(s) listed below.				
Application Number(s)	Filing Date (MM/DD/YYYY)		<input checked="" type="checkbox"/> Additional provisional application Numbers are listed on a supplemental priority data sheet PTO/SB/02B attached hereto.	
60/331,624	11/20/2001			
60/331,623	11/20/2001			
60/331,621	11/20/2001			

[Page 1 of 4]

Burden Hour Statement: This form is estimated to take 0.4 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissions for Patents, Washington, DC 20231.

BEST AVAILABLE COPY

Please type a plus sign (+) inside this box → [+]

PTO/SB/01 (12-97)

Approved for use through 9/30/00. OMB 0651-0032
 Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

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

DECLARATION – Utility or Design Patent Application

I hereby claim the benefit under 35 U.S.C. 120 of any United States application(s), or 365 of any PT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application in the manner provided by the first paragraph of 35 U.S.C. 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

U.S. Parent Application or PCT Parent Number	Parent Filing Date (MM/DD/YYYY)	Parent Patent Number (if applicable)

Additional U.S. or PCT international application are listed on a supplemental priority date sheet PTO/SB/02B attached hereto.

As a named inventor, I hereby appoint the following registered practitioner(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith: Customer Number 22204

OR

Registered practitioner(s) name/registration number listed below.

Name	Registration Number	Name	Registration Number
Stuart J. Friedman	24,312	Daniel S. Song	43,143
Charles M. Leedom, Jr.	26,477	Marc S. Kaufman	35,212
David S. Safran	27,997	Corinne R. Gorski	34,339
Thomas W. Cole	28,290	Jason H. Vick	45,285
Donald R. Studebaker	32,815	Luan C. Do	38,434
Jeffrey L. Costellia	35,483		
Tim L. Brackett, Jr.	36,092		

Direct all correspondence to: Customer Number 22204

Name: Marc S. Kaufman

Firm: NIXON PEABODY LLP

Address: 8180 Greensboro Drive, Suite 800

City: McLean State: VA ZIP: 22102

Country: United States Telephone: (703) 770-9300 FAX: (703) 770-9400

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

Name of Sole or First Inventor: A petition has been filed for this unsigned inventor.

Given Name (first and middle [if any])	Family Name or Surname
XIN	WANG

Inventor's Signature:  Date: 8/15/2002

Mailing Address (Street or P.O. Box): 3720 Emerald Street #V2

City: Torrance State: CA ZIP: 90503 Country: USA

Residence: City: State: Country:

Citizenship: USA

Additional inventors are being named on the _____ Supplemental Additional Inventor(s) sheet(s) PTO/SB/02A attached hereto.

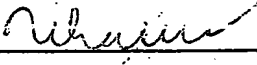
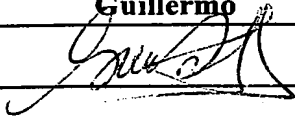
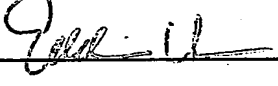
BEST AVAILABLE COPY

Please type a plus sign (+) inside this box → [+]

PTO/SB/02A (10-00)
 Approved for use through 10/31/2002. OMB 0651-0032
 Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

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

DECLARATION	ADDITIONAL INVENTOR(S) Supplemental Sheet Page ___ of ___
--------------------	--

Name of Additional Joint Inventor, if any:		<input type="checkbox"/> A petition has been filed for this unsigned inventor	
Given Name	Thanh	Family Name Or Surname	TA
Inventor's Signature			Date 08/15/2002
Mailing Address (Street or P.O. Box): 18694 Stratton Lane			
City: Huntington Beach	State: CA	ZIP: 92648	Country: USA
Residence: City:	State:	Country:	
Citizenship: Australia			
Name of Additional Joint Inventor, if any:		<input type="checkbox"/> A petition has been filed for this unsigned inventor	
Given Name	Guillermo	Family Name Or Surname	LAO
Inventor's Signature			Date 8/15/2002
Mailing Address (Street or P.O. Box): 5531 Lorna Street			
City: Torrance	State: CA	ZIP: 90503	Country: USA
Residence: City:	State:	Country:	
Citizenship: USA			
Name of Additional Joint Inventor, if any:		<input type="checkbox"/> A petition has been filed for this unsigned inventor	
Given Name	Eddie J.	Family Name Or Surname	CHEN
Inventor's Signature			Date 8/15/2002
Mailing Address (Street or P.O. Box): 6796 Vallon Drive			
City: Ranchos Palos Verdes	State: CA	ZIP: 90275	Country: USA
Residence: City:	State:	Country:	
Citizenship: USA			

Burden Hour Statement: This form is estimated to take 21 minutes to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231. [Page 3 of 4]

BEST AVAILABLE COPY

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

10/06/2004 AOSMAN1 00000012 192380 10956121

01 FC:1001 790.00 DA
02 FC:1202 288.00 DA

PTO-1556
(5/87)

PATENT APPLICATION FEE DETERMINATION RECORD
Effective October 1, 2004

Application or Docket Number

10956121

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)
TOTAL CLAIMS	36	
FOR	NUMBER FILED	NUMBER EXTRA
TOTAL CHARGEABLE CLAIMS	36 minus 20 =	* 16
INDEPENDENT CLAIMS	3 minus 3 =	*
MULTIPLE DEPENDENT CLAIM PRESENT		<input type="checkbox"/>

SMALL ENTITY TYPE

OR OTHER THAN SMALL ENTITY

RATE	FEE
BASIC FEE	395.00
X\$ 9=	
X44=	
+150=	
TOTAL	

RATE	FEE
BASIC FEE	790.00
X\$18=	288
X88=	
+300=	
TOTAL	1078

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

CLAIMS AS AMENDED - PART II

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT A	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR
	Total	* Minus	** =
	Independent	* Minus	*** =
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

SMALL ENTITY

OR OTHER THAN SMALL ENTITY

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X88=	
+300=	
TOTAL ADDIT. FEE	

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT B	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR
	Total	* Minus	** =
	Independent	* Minus	*** =
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X88=	
+300=	
TOTAL ADDIT. FEE	

	(Column 1)	(Column 2)	(Column 3)
AMENDMENT C	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR
	Total	* Minus	** =
	Independent	* Minus	*** =
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM <input type="checkbox"/>			

RATE	ADDITIONAL FEE
X\$ 9=	
X44=	
+150=	
TOTAL ADDIT. FEE	

RATE	ADDITIONAL FEE
X\$18=	
X88=	
+300=	
TOTAL ADDIT. FEE	

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

APPLICATION DATA SHEET

Electronic Version 0.0.11

Stylesheet Version: 1.0

Attorney Docket Number: 111325-291300

Publication Filing Type: new-utility

Application Type: utility

Title of Invention: SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS
USING SHARED STATE VARIABLES

Legal Representative:

Attorney or Agent: Marc S. Kaufman

Registration Number: 35212

Attorney or Agent: Carlos R. Villamar

Registration Number: 43224

Customer Number Correspondence Address: 22204

22204

Continuity Data:

This application is a continuation in part of 10/162,701 2002-06-06 which claims benefit from U.S. Provisional Application Serial Nos. 60/331,624; 60/331,623 and 60/331,621 filed on November 20, 2001 and U.S. Provisional Application Serial Nos. 60/296,113; 60/296,117 and 60/296,118 filed on June 7, 2001

INVENTOR(s):

Primary Citizenship: United States

Given Name: Mai

Family Name: Nguyen

Residence City: Buena Park

Residence State: CA

Residence Country: US

Address: 5611 Cambridge Avenue
Buena Park CA, 96021 US

Primary Citizenship: United States

Given Name: Xin
 Family Name: Wang
 Residence City: Torrance
 Residence State: CA
 Residence Country: US
 Address: 3720 Emerald Street, #V2
 Torrance CA, 90503 US

Primary Citizenship: United States
 Given Name: Thanh
 Family Name: Ta
 Residence City: Huntington Beach
 Residence State: CA
 Residence Country: US
 Address: 18694 Stratton Lane
 Hungtington Beach CA, 92648 US

Primary Citizenship: United States
 Given Name: Guillermo
 Family Name: Lao
 Residence City: Torrance
 Residence State: CA
 Residence Country: US
 Address: 5531 Lorna Street
 Torrance CA, 90503 US

Primary Citizenship: United Sttes
 Given Name: Eddie
 Middle Name: J.
 Family Name: Chen
 Residence City: Rancho Palos Verdes
 Residence State: CA
 Residence Country: US
 Address: 6796 Vallon Drive
 Rancho Palos Verdes CA, 90275 US

100404
13281 U.S. PTO

Approved for use through 10/31/2002. OMB 0651-0032
PTO/SB/05 (03-01)

UTILITY PATENT APPLICATION TRANSMITTAL <i>(Only for new nonprovisional applications under 37 CFR 1.53(b))</i>	Attorney Docket No. 111325-291300	
	First Inventor Mai NGUYEN, et al.	
	Title	SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS USING SHARED STATE VARIABLES
	Express Mail Label No.	

17497 U.S. PTO
10/956121

100404

APPLICATION ELEMENTS	ADDRESS TO: Commissioner for Patents Box Patent Application Washington, DC 20231
----------------------	--

- See MPEP chapter 600 concerning utility patent application contents.
- Fee Transmittal Form (e.g., PTO/SB/17)
(Submit an original and a duplicate for fee processing)
 - Applicant claims small entity status.
See 37 CFR 1.27.
 - Specification [Total Pages 33]
(preferred arrangement set forth below)
 - Descriptive title of the invention
 - Cross Reference to Related Applications *(if applicable)*
 - Statement Regarding Fed sponsored R & D *(if applicable)*
 - Reference to sequence listing, a table, or a computer program listing appendix *(if applicable)*
 - Background of the Invention
 - Brief Summary of the Invention
 - Brief Description of the Drawings *(if filed)*
 - Detailed Description
 - Claim(s)
 - Abstract of the Disclosure
 - Drawing(s) (35 U.S.C. 113) 17 Figures [Total Sheets 14]
 - Oath or Declaration [Total Pages]
 - Newly executed (original or copy)
 - Copy from a prior application (37 CFR 1.63(d))
(for continuation/divisional with Box 18 completed)
 - DELETION OF INVENTOR(S)**
Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b)
 - Application Data Sheet. See 37 CFR 1.76

- CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
- Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
 - Computer Readable Form (CRF)
 - Specification Sequence Listing on:
 - CD-ROM or CD-R (2 copies; or
 - paper
 - Statements verifying identity of above copies

ACCOMPANYING APPLICATION PARTS	
9. <input type="checkbox"/> Assignment Papers (cover sheet & document(s))	
10. <input type="checkbox"/> 37 CFR 3.73(b) Statement <input type="checkbox"/> Power of Attorney <i>(when there is an assignee)</i>	
11. <input type="checkbox"/> English Translation Document <i>(if applicable)</i>	
12. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations	
13. <input type="checkbox"/> Preliminary Amendment	
14. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) <i>(Should be specifically itemized)</i>	
15. <input type="checkbox"/> Certified Copy of Priority Document(s) <i>(if foreign priority is claimed)</i>	
16. <input type="checkbox"/> Nonpublication request under 35 U.S.C. 122(b)(2)(B)(i). Applicant must attach form PTO/SB/35 or its equivalent.	
17. <input type="checkbox"/> Other: _____	

18. If a CONTINUING APPLICATION, check appropriate box, and supply the requisite information below and in a preliminary amendment, or in an Application Data Sheet under 37 CFR 1.76:

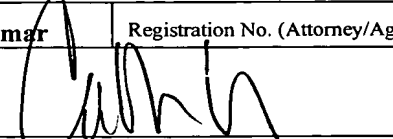
Continuation Continuation-in-part (CIP) of prior application No: 10/162,701

Prior application information: Examiner Not Yet Assigned Group / Art Unit: 2122

For CONTINUATION OR DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

19. CORRESPONDENCE ADDRESS	
<input checked="" type="checkbox"/> Customer Number or Bar Code Label	22204
	or <input type="checkbox"/> Correspondence address below

Name			
Address			
City	State	Zip Code	
Country	Telephone	Fax	

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224
Signature		Date	October 4, 2004

100404

103281 U.S. PATENT & TRADEMARK OFFICE

FEE TRANSMITTAL FOR FY 2004

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

Complete if Known	
Application Number	Not Yet Assigned
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Not Yet Assigned
Art Unit	Not Yet Assigned
Attorney Docket No.	111325-291300

TOTAL AMOUNT OF PAYMENT **\$1,078.00**

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:

Deposit Account Number: **19-2380**

Deposit Account Name: **Nixon Peabody LLP**

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments

Charge any additional fee(s)

Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity		Small Entity		Fee Description
Fee Code	Fee (\$)	Fee Code	Fee (\$)	
1051	130	2051	65	Surcharge - late filing fee or oath
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet
1053	130	1053	130	Non-English specification
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action
1251	110	2251	55	Extension for reply within first month
1252	430	2252	215	Extension for reply within second month
1253	980	2253	490	Extension for reply within third month
1254	1,530	2254	765	Extension for reply within fourth month
1255	2,080	2255	1,040	Extension for reply within fifth month
1401	340	2401	170	Notice of Appeal
1402	340	2402	170	Filing a brief in support of an appeal
1403	340	2403	150	Request for oral hearing
1451	1,510	1451	1,510	Petition to institute a public use proceeding
1452	110	2452	55	Petition to revive - unavoidable
1453	1,370	2453	685	Petition to revive - unintentional
1501	1,370	2501	685	Utility issue fee (or reissue)
1502	490	2502	245	Design issue fee
1503	660	2503	330	Plant issue fee
1460	130	1460	130	Petitions to the Commissioner
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)
1806	180	1806	180	Submission of Information Disclosure Stmt
8021	40	8021	40	Recording each patent assignment per property (times number of properties)
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))
1801	790	2801	395	Request for Continued Examination (RCE)
1802	900	1802	900	Request for expedited examination of a design application

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) (\$ 0)

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____

Date

Signature

Typed or printed name

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	790	2001	395	Utility filing fee	790.00
1002	350	2002	175	Design filing fee	
1003	550	2003	275	Plant filing fee	
1004	790	2004	395	Reissue filing fee	
1005	160	2005	80	Provisional filing fee	

SUBTOTAL (1) \$790.00

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims **36** -20** = **16** X **18.00** = **\$288.00**

Independent Claims **0** -3** = **0** X **0** = **0**

Multiple Dependent **0** X **0** = **0**

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1202	18	2202	9	Claims in excess of 20	
1201	88	2201	44	Independent claims in excess of 3	
1203	300	2203	150	Multiple dependent claim, if not paid	
1204	88	2204	44	** Reissue independent claims over original patent	
1205	18	2205	9	** Reissue claims in excess of 20 and over original patent	

SUBTOTAL (2) \$288.00

**or number previously paid, if greater, For Reissues, see above

SUBMITTED BY

Name (Print/Type)	Carlos R. Villamar	Registration No. (Attorney/Agent)	43,224	Telephone	(202) 585-8204
Signature		Date	October 4, 2004		

Complete (if applicable)

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450



FW

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,121
	Filing Date	October 4, 2004
	First Named Inventor	Mai NGUYEN, <i>et al.</i>
	Group Art Unit	2131
	Examiner Name	Not Yet Assigned
Total Number of Pages in This Submission	Attorney Docket Number	111325-291300

ENCLOSURES (check all that apply)		
<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input checked="" type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers (for an Application) <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input checked="" type="checkbox"/> Application Data Sheet <input checked="" type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) (please identify below):
Remarks		<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Marc S. Kaufman Registration No. 35,212 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	
Date	January 26, 2005

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
Date	Signature
_____	_____
	Typed or printed name



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	
Mai NGUYEN, <i>et al.</i>)	Examiner: Unassigned
Serial No. 10/956,121)	Group Art Unit: 2131
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR MANAGING)		
TRANSFER OF RIGHTS USING SHARED)		
STATE VARIABLES)		

U.S. Patent and Trademark Office
Customer Window, Mail Stop
Randolph Building
Alexandria, VA 22314

Sir:


INFORMATION DISCLOSURE STATEMENT

In accordance with the duty of disclosure as set forth in 37 C.F.R. §1.56, Applicants hereby submit the following information in conformance with 37 C.F.R. §§ 1.97 and 1.98. The references listed on the attached PTO-1449 forms have been made of record in parent application serial number 10/162,701 Filed on June 6, 2002, therefore no copies of the references cited are submitted herewith.

It is requested that the accompanying PTO-1449 be considered and made of record in the above-identified application. To assist the Examiner, the documents are listed on the attached form PTO-1449. It is respectfully requested that an Examiner initial a copy of this form be returned to the undersigned.

The Commissioner is hereby authorized to charge any fees connected with this filing which may be required now, or credit any overpayment to Deposit Account No. 19-2380 (111325-291300).

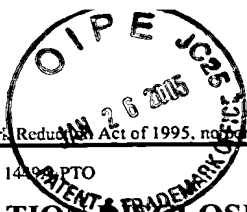
Respectfully submitted,
NIXON PEABODY, LLP

By: 

Marc S. Kaufman
Registration No. 35, 212

Date: **January 26, 2005**

Customer No. 22204
NIXON PEABODY LLP
401 9th Street, N.W., Suite 900
Washington, DC 20004-2128
Telephone: (202) 585-8000



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

Substitute for form 1449a PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	January 26, 2005
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	Not Yet Assigned
				Examiner Name	Not Yet Assigned
Sheet	1	of	10	Attorney Docket Number	111325-291300

U.S. PATENT DOCUMENTS						
Examiner Initials ²	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ³ (if known)				
		US-3,263,158		07/26/1966	D.W. Bargaen et al.	
		US-3,609,697		09/28/1971	Blevins et al.	
		US-3,790,700		02/05/1974	Callais et al.	
		US-3,798,605		03/19/1974	Feistel	
		US-4,159,468		06/26/1979	Barnes et al.	
		US-4,220,991		09/02/1980	Hamano et al.	
		US-4,278,837		07/14/1981	Best	
		US-4,323,921		04/06/1982	Guillou	
		US-4,442,486		04/10/1984	Mayer	
		US-4,529,870		07/16/1985	Chaum	
		US-4,558,176		12/10/1985	Arnold et al.	
		US-4,593,376		06/03/1986	Volk	
		US-4,614,861		09/30/1986	Pavlov et al.	
		US-4,644,493		02/17/1987	Chandra et al.	
		US-4,658,093		04/14/1987	Hellman	
		US-4,713,753		12/15/1987	Beobert et al.	
		US-4,796,220		01/03/1989	Wolfe	
		US-4,817,140		03/28/1989	Chandra et al.	
		US-4,827,508		05/02/1989	Shear	
		US-4,868,376		09/19/1989	Lessin et al.	

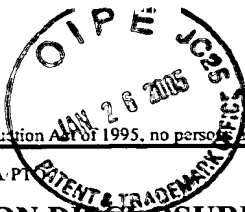
FOREIGN PATENT DOCUMENTS							
Examiner Initials ²	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴ Kind Code ⁵ (if known)				
		0 084 441	EP	07/27/1983	TABS LIMITED		
		0 180 460	EP	05/07/1986	SONY CORPORATION		
		0 332 707	EP	09/20/1989	HONDA GIKEN KOGYO KABUSHIKI KAISHA		
		0 651 554	EP	05/03/1995	EASTMAN KODAK CO.		
		0 668 695	EP	08/23/1995	VICTOR COMPANY OF JAPAN LIMITED		
		0 725 376	EP	08/07/1996	SONY CORP.		
		2 136 175	GB	09/12/1984	ATALLA CORP.		
		2 236 604	GB	04/10/1991	SUN MICROSYSTEMS INC		
		0 715 241	JP	06/05/1996	MITSUBISHI CORP.		
		04-369068	JP	12/21/1992	CHIYUUBU NIHON DENKI SOFUTOUEA KK		Abst

Examiner Signature	Date Considered
--------------------	-----------------

¹ EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

² Applicant's unique citation designation number (optional). ³ See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ⁴ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁵ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁶ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁷ Applicant is to place a check mark here if English language translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO: Commissioner for Patents, Washington, DC 20231.



Under the Paperwork Reduction Act of 1995, no person is required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A-PT INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Complete if Known		
				Application Number	10/956,121	
Sheet		2	of	10	Examiner Name	Not Yet Assigned
					Attorney Docket Number	111325-291300

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-4,891,838		01/02/1990	Faber	
		US-4,924,378		05/08/1990	Hershey et al.	
		US-4,932,054		06/05/1990	Chou et al.	
		US-4,937,863		06/26/1990	Robert et al.	
		US-4,949,187		08/14/1990	Cohen	
		US-4,953,209		08/28/1990	Ryder, Sr. et al.	
		US-4,961,142		10/02/1990	Elliott et al.	
		US-4,975,647		12/04/1990	Downer et al.	
		US-4,977,594		12/11/1990	Shear	
		US-4,999,806		03/12/1991	Chernow et al.	
		US-5,010,571		04/23/1991	Katznelson	
		US-5,014,234		05/07/1991	Edwards, Jr.	
		US-5,023,907		06/11/1991	Johnson et al.	
		US-5,047,928		09/10/1991	Wiedemer	
		US-5,050,213		09/17/1991	Shear	
		US-5,052,040		09/24/1991	Preston et al.	
		US-5,058,164		10/15/1991	Elmer et al.	
		US-5,103,476		04/07/1992	Waite et al.	
		US-5,113,519		05/12/1992	Johnson et al.	
		US-5,136,643		08/04/1992	Fischer	

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴ Kind Code ² (if known)				
		05-268415	JP	10/15/1993	RICOH CO LTD		Abst
		06-175794	JP	06/24/1994	FUJI XEROX CO LTD		Abst
		06-215010	JP	08/05/1994	SONY CORP.		Abst
		07-084852	JP	03/31/1995	HITACHI LTD.		Abst
		07-200317	JP	08/04/1995	TOSHIBA CORP.		Abst
		07-244639	JP	09/19/1995	FUJITSU LTD		Abst
		62-241061	JP	10/21/1987	NEC CORP.		Abst
		64-068835	JP	03/14/1989	RYOICHI MORI		Abst
		WO 01/63528	PCT	08/30/2001	IPDN COPR.		
		WO 92/20022	PCT	11/12/1992	DIGITAL EQUIPMENT CORP.		

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

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

Substitute for form 1449A PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	January 26, 2005
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	Not Yet Assigned
				Examiner Name	Not Yet Assigned
Sheet	3	of	10	Attorney Docket Number	111325-291300

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,138,712		08/11/1992	Corbin	
		US-5,146,499		09/08/1992	Geffrotin	
		US-5,148,481		09/15/1992	Abraham et al.	
		US-5,159,182		10/27/1992	Eisele	
		US-5,183,404		02/02/1993	Aldous et al.	
		US-5,191,193		03/02/1993	Le Roux	
		US-5,204,897		04/20/1993	Wyman	
		US-5,222,134		06/22/1993	Waite et al.	
		US-5,235,642		08/10/1993	Wobber et al.	
		US-5,247,575		09/21/1993	Sprague et al.	
		US-5,255,106		10/19/1993	Castro	
		US-5,260,999		11/09/1993	Wyman	
		US-5,263,157		11/16/1993	Janis	
		US-5,263,158		11/16/1993	Janis	
		US-5,276,444		01/04/1994	McNair	
		US-5,276,735		01/04/1994	Boebert et al.	
		US-5,291,596		03/01/1994	Mita	
		US-5,301,231		04/05/1994	Abraham et al.	
		US-5,311,591		05/10/1994	Fischer	
		US-5,319,705		06/07/1994	Halter et al.	

FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ⁵ (if known)				
		WO 93/01550	PCT	01/21/1993	INFOLOGIC SOFTWARE, INC		
		WO 94/01821	PCT	01/20/1994	SECURE COMPUTING CORP.		
		WO 96/24092	PCT	08/08/1996	BENSON, Greg		
		WO 97/48203	PCT	12/18/1997	INTEL CORP.		
		WO 98/11690	PCT	03/19/1998	GLOVER, John J.		
		WO 98/42098	PCT	09/24/1998	CRYPTOWORKS, INC.		
		WO 99/49615	PCT	09/30/1999	MICROTOME		
		WO 00/20950		04/13/2000			
Examiner Signature				Date Considered			

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

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

Substitute for form 1449A PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	January 26, 2005
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	2131
				Examiner Name	Not Yet Assigned
				Attorney Docket Number	111325-191300
Sheet	5	of	10		

U.S. PATENT DOCUMENTS						
Examiner Initials ²	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ³ (if known)				
		US-5,504,818		04/02/1996	Okano	
		US-5,504,837		04/02/1996	Griffeth et al.	
		US-5,509,070		04/16/1996	Schull	
		US-5,530,235		06/25/1996	Stefik et al.	
		US-5,532,920		07/02/1996	Hartrick et al.	
		US-5,534,975		07/09/1996	Stefik et al.	
		US-5,539,735		07/23/1996	Moskowitz	
		US-5,563,946		10/08/1996	Cooper et al.	
		US-5,568,552		10/22/1996	Davis	
		US-5,621,797		04/15/1997	Rosen	
		US-5,629,980		05/13/1997	Stefik et al.	
		US-5,633,932		05/27/1997	Davis et al.	
		US-5,634,012		05/27/1997	Stefik et al.	
		US-5,638,443		06/10/1997	Stefik et al.	
		US-5,649,013		07/15/1997	Stuckey et al.	
		US-5,655,077		08/05/1997	Jones et al.	
		US-5,708,717		01/13/1998	Alasia	
		US-5,734,823		03/31/1998	Saigh et al.	
		US-5,734,891		03/31/1998	Saigh	
		US-5,737,413		04/07/1998	Akiyama et al.	

FOREIGN PATENT DOCUMENTS								
Examiner Initials ²	Cite No. ¹	Foreign Patent Document			Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³	Number ⁴	Kind Code ⁵ (if known)				

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

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

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	January 26, 2005
				First Named Inventor	Mai NGUYEN, <i>et al.</i>
				Art Unit	2131
				Examiner Name	Not Yet Assigned
Sheet	8	of	10	Attorney Docket Number	111325-291300

OTHER PRIOR ART – NON PATENT LITERATURE DOCUMENTS			
Examiner Initials [*]	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		“National Semiconductor and EPR Partner for Information Metering/Data Security Cards” March 4, 1994, Press Release from Electronic Publishing Resources, Inc.	
		Weber, R., “Digital Rights Management Technology” October 1995	
		Flasche, U. et al., “Decentralized Processing of Documents”, pp. 119-131, 1986, Comput. & Graphics, Vol. 10, No. 2	
		Mori, R. et al., “Superdistribution: The Concept and the Architecture”, pp. 1133-1146, 1990. The Transactions of the IEICE, Vo. E 73, No. 7, Tokyo, JP	
		Weber, R., “Metering Technologies for Digital Intellectual Property”, pp. 1-29, Oct. 1994, A Report to the International Federation of Reproduction Rights Organizations	
		Clark, P.C. et al., “Bits: A Smartcard protected Operating System”, pp. 66-70 and 94, November 1994, Communications of the ACM, Vol. 37, No. 11	
		Ross, P.E., “Data Guard”, pp. 101, June 6, 1994, Forbes	
		Saigh, W.K., “Knowledge is Sacred”, 1992, Video Pocket/Page Reader Systems, Ltd.	
		Kahn, R.E., “Deposit, Registration and Recordation in an Electronic Copyright Management System”, pp. 1-19, August 1992, Corporation for National Research Initiatives, Virginia	
		Hilts, P. et al., “Books While U Wait”, pp. 48-50, January 3, 1994, Publishers Weekly	
		Stratner, A., “Cash Register on a Chip may Revolutionize Software Pricing and Distribution; Wave Systems Corp.”, pp. 1-3, April 1994, Computer Shopper, Vol. 14, No. 4, ISSN 0886-0556	

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

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

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT (use as many sheets as necessary)				<i>Complete if Known</i>		
				Application Number	10/956,121	
Sheet 9 of 10				Filing Date	January 26, 2005	
				First Named Inventor	Mai NGUYEN, et al.	
				Art Unit	2131	
				Examiner Name	Not Yet Assigned	
				Attorney Docket Number	111325-291300	

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials [*]	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		O'Conner, M., "New Distribution Option for Electronic Publishers; iOpener Data Encryption and Metering System for CD-ROM use; Column", pp. 1-6, March 1994, CD-ROM Professional, Vol. 7, No. 2, ISSN: 1409-0833	
		Willett, S., "Metered PCs: Is Your System Watching You? Wave System beta tests new technology", pp. 84, May 2, 1994, InfoWorld	
		Linn, R., "Copyright and Information Services in the Context of the National Research and Education Network", pp. 9-20, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Perrit, Jr., H., "Permission Headers and Contract Law", pp. 27-48, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Upthegrove, L., "Intellectual Property Header Descriptors: A Dynamic Approach", pp. 63-66, January 1994, IMA Intellectual Property Proceedings, Vol. 1, Issue 1	
		Sirbu, M., "Internet Billing Service Design and prototype Implementation", pp. 67-80, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Simmell, S. et al., "Metering and Licensing of Resources: Kala's General Purpose Approach", pp. 81-110, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Kahn, R., "Deposit, Registration and Recordation in an Electronic Copyright Management System", pp. 111-120, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Tygar, J. et al., "Dyad: A System for Using Physically Secure Coprocessors", pp. 121-152, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Griswold, G., "A Method for Protecting Copyright on Networks", pp. 169-178, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	

Examiner Signature		Date Considered	
-----------------------	--	--------------------	--

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

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

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956
				Filing Date	January 26, 2005
				First Named Inventor	Mai NGUYEN, et al.
				Art Unit	2131
				Examiner Name	Not Yet Assigned
Sheet	10	of	10	Attorney Docket Number	111325-291300

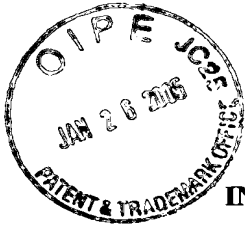
OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		Nelson, T., "A Publishing and Royalty Model for Networked Documents", pp. 257-259, January 1994, IMA Intellectual Property Project Proceedings, Vol. 1, Issue 1	
		Robinson, E., "Redefining Mobile Computing", pp. 238-240, 247-248 and 252, July 1993, PC Computing	
		Abadi, M. et al., "Authentication and Delegation with Smart-cards", PP. 1-24, 1990, Research Report DEC Systems Research Center	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 219-253, 1996, Internet Dreams: Archetypes, Myths, and Metaphors, IDSN 0-262-19373-6	
		Mark Stefik, "Letting Loose the Light: Igniting Commerce in Electronic Publication", pp. 2-35, February 8, 1995, Internet Dreams: Archetypes, Myths and Metaphors	
		Henry H. Perritt, Jr., "Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment", April 2-3, 1993, Knowbots, Permissions Headers & Contract Law	

Examiner Signature	Date Considered	
--------------------	-----------------	--

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

Burden Hour Statement: This form is estimated to take 2.0 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of)	Confirmation No.: 8924
Mai NGUYEN, <i>et al.</i>)	Group Art Unit: 2131
Application No. 10/956,121)	Examiner: Unassigned
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR)	
MANAGING TRANSFER OF RIGHTS)	
USING SHARED STATE VARIABLES)	

U.S. Patent and Trademark Office
Customer Window, Mail Stop
Randolph Building
Alexandria, VA 22314

REQUEST FOR CORRECTED OFFICIAL FILING RECEIPT

Sir:


Please note that upon reviewing the Official Filing Receipt received in the above-identified application, we noted that the first inventor was omitted. Please add as the first inventor:

--Mai NGUYEN--

Submitted herewith is a copy of the original Official Filing Receipt indicating this addition marked in red. Also, enclosed is a copy of application data sheet indicating that this error was due to an oversight at the USPTO.

In view of the above, it is requested that a Corrected Filing Receipt be issued.

Respectfully submitted,
NIXON PEABODY, LLP



Marc S. Kaufman
Registration No. 35,212

NIXON PEABODY LLP
Customer No. 22204
401 Ninth Street, N.W., Suite 900
Washington, DC 20004
Telephone: (202) 585-8000
Fax:: (202) 585-8080



UNITED STATES PATENT AND TRADEMARK OFFICE

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

APPL NO.	FILING OR 371 (c) DATE	ART UNIT	FIL FEE REC'D	ATTY. DOCKET NO	DRAWINGS	TOT CLMS	IND CLMS
10/956,121	10/04/2004	2131	1078	111325-291300	14	36	3

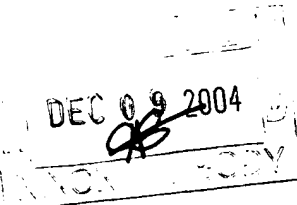
CONFIRMATION NO. 8924

FILING RECEIPT



OC000000014674573

22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC 20004-2128



Date Mailed: 12/07/2004

Receipt is acknowledged of this regular Patent Application. It will be considered in its order and you will be notified as to the results of the examination. Be sure to provide the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION when inquiring about this application. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please write to the Office of Initial Patent Examination's Filing Receipt Corrections, facsimile number 703-746-9195. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections (if appropriate).**

Applicant(s) *Xin Wang, Thanh Ta, Guillermo Lao, Eddie J. Chen*
 Xin Wang, Torrance, CA;
 Thanh Ta, Huntington Beach, CA;
 Guillermo Lao, Torrance, CA;
 Eddie J. Chen, Rancho Palos Verdes, CA;

Power of Attorney: The patent practitioners associated with Customer Number 22204.

Domestic Priority data as claimed by applicant

This application is a CIP of 10/162,701 06/06/2002
 which claims benefit of 60/331,624 11/20/2001
 and claims benefit of 60/331,623 11/20/2001
 and claims benefit of 60/331,621 11/20/2001
 and claims benefit of 60/296,113 06/07/2001
 and claims benefit of 60/296,117 06/07/2001
 and claims benefit of 60/296,118 06/07/2001

Foreign Applications

If Required, Foreign Filing License Granted: 12/03/2004

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US10/956,121**

Projected Publication Date: 03/17/2005

Non-Publication Request: No

Early Publication Request: No

Title

System and method for managing transfer of rights using shared state variables

Preliminary Class

713

**LICENSE FOR FOREIGN FILING UNDER
Title 35, United States Code, Section 184
Title 37, Code of Federal Regulations, 5.11 & 5.15**

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

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

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Office of Export Administration, Department of Commerce (15 CFR 370.10 (j)); the Office of Foreign Assets Control, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

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



APPLICATION DATA SHEET

Electronic Version 0.0.11

Stylesheet Version: 1.0

Attorney Docket Number: 111325-291300

Publication Filing Type: new-utility

Application Type: utility

Title of Invention: SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS
USING SHARED STATE VARIABLES

Legal Representative:

Attorney or Agent: Marc S. Kaufman

Registration Number: 35212

Attorney or Agent: Carlos R. Villamar

Registration Number: 43224

Customer Number Correspondence Address: 22204

22204

Continuity Data:

This application is a continuation in part of 10/162,701 2002-06-06 which claims benefit from U.S. Provisional Application Serial Nos. 60/331,624; 60/331,623 and 60/331,621 filed on November 20, 2001 and U.S. Provisional Application Serial Nos. 60/296,113; 60/296,117 and 60/296,118 filed on June 7, 2001

INVENTOR(s):

Primary Citizenship: United States

Given Name: Mai

Family Name: Nguyen

Residence City: Buena Park

Residence State: CA

Residence Country: US

Address: 5611 Cambridge Avenue

Buena Park CA, 96021 US

Primary Citizenship: United States

Given Name: Xin
Family Name: Wang
Residence City: Torrance
Residence State: CA
Residence Country: US
Address: 3720 Emerald Street, #V2
Torrance CA, 90503 US

Primary Citizenship: United States
Given Name: Thanh
Family Name: Ta
Residence City: Huntington Beach
Residence State: CA
Residence Country: US
Address: 18694 Stratton Lane
Huntington Beach CA, 92648 US

Primary Citizenship: United States
Given Name: Guillermo
Family Name: Lao
Residence City: Torrance
Residence State: CA
Residence Country: US
Address: 5531 Lorna Street
Torrance CA, 90503 US

Primary Citizenship: United Sttes
Given Name: Eddie
Middle Name: J.
Family Name: Chen
Residence City: Rancho Palos Verdes
Residence State: CA
Residence Country: US
Address: 6796 Vallon Drive
Rancho Palos Verdes CA, 90275 US

FILE COPY

UNITED STATES PATENT AND TRADEMARK OFFICE

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

APPLICATION NUMBER	FILING OR 371(c) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
10/956,121	10/04/2004	Xin Wang	111325-291300

CONFIRMATION NO. 8924

OC000000015158564

 22204
 NIXON PEABODY, LLP
 401 9TH STREET, NW
 SUITE 900
 WASHINGTON, DC 20004-2128

Date Mailed: 02/10/2005

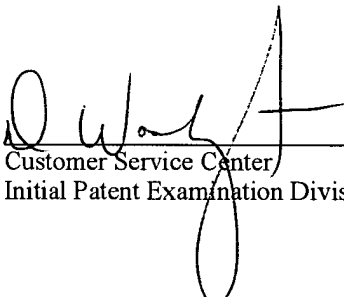
RESPONSE TO REQUEST FOR CORRECTED FILING RECEIPT**Claims, Fees, and Inventors**

In response to your request for a corrected Filing Receipt, the Office is unable to comply with the request because:

- The total number of claims appearing on the Filing Receipt does not include multiple dependent claims. The total fee appearing on the Filing Receipt includes the cost of multiple dependent claims that were present at the time the application was filed.
- The filing fee is correct. It may include additional claims fees and/or the surcharge under 37 CFR 1.16 (e) for filing an oath/declaration or basic filing fee after the application filing date; or it may not reflect fees refunded to the applicant that were paid by mistake.
- The number of claims reflected on the filing receipt is correct. Upon review of the claims, it was found that there was a miscalculation by the applicant. This may be due to improperly presented multiple dependent claims, typographical error, misnumbering of the claims, or other oversight. An amendment may be necessary to correct the problem.
- The filing fee reflected on the filing receipt is correct. Applicant may have miscalculated the fees due.
- Applicant calculated fees as other than small entity; however, applicant asserted small entity status in the application. Therefore, fees were applied as small entity and the remainder was refunded to the applicant.
- The difference between the fees paid and the fees due was refunded to the applicant and will not be shown on the filing receipt.
- The inventor information may be truncated if the family name consists of more than 50 characters (letters and spaces combined) and if the given name consists of more than 50 characters (letters and

spaces combined).

- The inventor's residence allows for up to 40 characters (letters and spaces combined).
- The inventor's residence will only include the city and state for U.S. residences or city and country for residences outside the U.S. (See MPEP 605.02).
- A petition to correct the inventorship is needed to make this change. See 37 CFR 1.48. For non-provisional applications, the petition should be directed to the Director of the examining group assigned to your application.
- Changes made after submission of an executed declaration to the inventor information other than correction of typographical errors must be submitted in the form of a substitute declaration. Change of inventorship requires a petition under 37 CFR 1.48.
- The number of drawings shown on the filing receipt reflects the number of drawing sheets submitted and is not necessarily equal to the number of figures submitted.
- The correspondence address was captured as directed by applicant on filing. If you wish correspondence to be directed otherwise, please submit a request for a change of address.
- The docket number allows a maximum of 25 characters.
- The person signing on behalf of the deceased inventor is reflected on the Filing Receipt as the legal representative.
- The filing date of a parent application cannot be changed by this request. A petition to correct the filing date in the parent application is required.



Customer Service Center
Initial Patent Examination Division (703) 308-1202

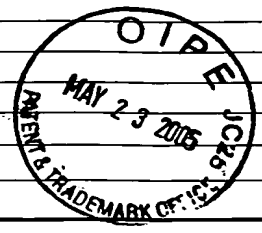
PART 3 - OFFICE COPY

FEE TRANSMITTAL FOR FY 2005

Patent fees are subject to annual revision.

Applicant claims small entity status. See 37 CFR 1.27

<i>Complete if Known</i>	
Application Number	10/956,121
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Unassigned
Art Unit	2131
Attorney Docket No.	111325-291300



TOTAL AMOUNT OF PAYMENT \$40.00

METHOD OF PAYMENT (check all that apply)

Check Credit Card Money Order Other None

Deposit Account:
 Deposit Account Number: 19-2380
 Deposit Account Name: Nixon Peabody LLP

The Commissioner is authorized to: (check all that apply)

Charge fee(s) indicated below Credit any overpayments
 Charge any additional fee(s)
 Charge fee(s) indicated below, except for the filing fee to the above-identified deposit account.

FEE CALCULATION (continued)

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1001	300	2001	150	Utility filing fee	
1002	200	2002	100	Design filing fee	
1003	200	2003	100	Plant filing fee	
1004	300	2004	150	Reissue filing fee	
1005	200	2005	100	Provisional filing fee	
SUBTOTAL (1)					(\$) 0

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	
1051	130	2051	65	Surcharge - late filing fee or oath	
1052	50	2052	25	Surcharge - late provisional filing fee or cover sheet	
1053	130	1053	130	Non-English specification	
1812	2,520	1812	2,520	For filing a request for <i>ex parte</i> reexamination	
1804	920*	1804	920*	Requesting publication of SIR prior to Examiner action	
1805	1,840*	1805	1,840*	Requesting publication of SIR after Examiner action	
1251	120	2251	60	Extension for reply within first month	
1252	450	2252	225	Extension for reply within second month	
1253	1,020	2253	510	Extension for reply within third month	
1254	1,590	2254	795	Extension for reply within fourth month	
1255	2,160	2255	1,080	Extension for reply within fifth month	
1401	500	2401	250	Notice of Appeal	
1402	500	2402	250	Filing a brief in support of an appeal	
1403	1,000	2403	500	Request for oral hearing	
1451	1,510	1451	1,510	Petition to institute a public use proceeding	
1452	500	2452	250	Petition to revive - unavoidable	
1453	1,500	2453	750	Petition to revive - unintentional	
1501	1,400	2501	700	Utility issue fee (or reissue)	
1502	800	2502	400	Design issue fee	
1503	1,100	2503	550	Plant issue fee	
1460	130	1460	130	Petitions to the Commissioner	
1807	50	1807	50	Processing fee under 37 CFR 1.17(q)	
1806	180	1806	180	Submission of Information Disclosure Stmt	
8021	40	8021	40	Recording each patent assignment per property (times number of properties)	40.00
1809	790	2809	395	Filing a submission after final rejection (37 CFR 1.129(a))	
1810	790	2810	395	For each additional invention to be examined (37 CFR 1.129(b))	
1801	790	2801	395	Request for Continued Examination (RCE)	
1802	900	1802	900	Request for expedited examination of a design application	

Other fee (specify) _____

2. EXTRA CLAIM FEES FOR UTILITY AND REISSUE

Total Claims	Extra Claims	Fee from below	Fee Paid
<input type="text"/> -20** = <input type="text"/> X <input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Independent Claims <input type="text"/> -3** = <input type="text"/> X <input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
Multiple Dependent <input type="text"/> X <input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Large Entity Fee Code	Large Entity Fee (\$)	Small Entity Fee Code	Small Entity Fee (\$)	Fee Description	Fee Paid
1202	50	2202	25	Claims in excess of 20	
1201	200	2201	100	Independent claims in excess of 3	
1203	360	2203	180	Multiple dependent claim, if not paid	
1204	200	2204	100	** Reissue independent claims over original patent	
1205	50	2205	25	** Reissue claims in excess of 20 and over original patent	
SUBTOTAL (2)					(\$) 0

**or number previously paid, if greater; For Reissues, see above

*Reduced by Basic Filing Fee Paid **SUBTOTAL (3) \$40.00**

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]

I hereby certify that this correspondence is being:

deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450

transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.

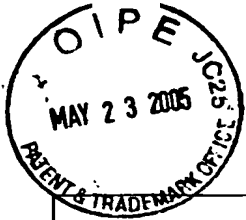
Date

Signature

Typed or printed name

SUBMITTED BY		<i>Complete (if applicable)</i>	
Name (Print/Type)	Marc S. Kaufman	Registration No. (Attorney/Agent)	35,212
Signature		Telephone	(202) 585-8000
		Date	May 23, 2005

SEND TO: Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450



DECLARATION (37 CFR 1.63) FOR UTILITY OR DESIGN APPLICATION USING AN APPLICATION DATA SHEET (37 CFR 1.76) AND POWER OF ATTORNEY

Attorney Docket No. 111325-291300

Title of Invention SYSTEM AND METHOD FOR MANAGING TRANSFER OF RIGHTS USING SHARED STATE VARIABLES

As the below named inventor(s), I/we declare that:

This declaration is directed to:

- The attached application, or
[X] Application No. 10/956,121 filed on October 4, 2004,
as amended on (if applicable);

I/We believe that I/we am/are the original and first inventor(s) of the subject matter which is claimed and for which a patent is sought;

I/We have reviewed and understand the contents of the above-identified application, including the claims, as amended by any amendment specifically referred to above;

I/We acknowledge the duty to disclose to the United States Patent and Trademark Office all information known to me/us to be material to patentability as defined in 37 CFR 1.56, including for continuation-in-part applications, material information which became available between the filing date of the prior application and the national or PCT International filing date of the continuation-in-part application.

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

I/We hereby appoint:

Practitioners at Customer Number 22204 as my/our attorney(s) or agent(s) to prosecute the application identified above, and to transact all business in the United States Patent and Trademark Office connected therewith.

FULL NAME OF INVENTOR(S)

Inventor one: Mai NGUYEN

Signature: Mai Nguyen Citizen of: United States

Inventor two: Xin WANG

Signature: Xin Wang Citizen of: United States

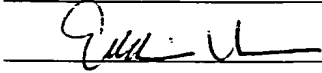
Inventor three: Thanh TA

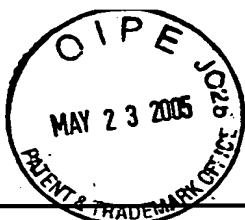
Signature: Thanh TA Citizen of: United States

Inventor four: Guillermo LAO

Signature: Guillermo LAO Citizen of: United States

Additional inventors are being named on page two attached hereto.

Inventor five:	<u>Eddie J. CHEN</u>	
Signature:		Citizen of: <u>United States</u>
Inventor six:	_____	
Signature:	_____	Citizen of: _____
Inventor seven:	_____	
Signature:	_____	Citizen of: _____
Inventor eight:	_____	
Signature:	_____	Citizen of: _____
Inventor nine:	_____	
Signature:	_____	Citizen of: _____
Inventor ten:	_____	
Signature:	_____	Citizen of: _____
Inventor eleven:	_____	
Signature:	_____	Citizen of: _____
Inventor twelve:	_____	
Signature:	_____	Citizen of: _____
Inventor thirteen:	_____	
Signature:	_____	Citizen of: _____
Inventor fourteen:	_____	
Signature:	_____	Citizen of: _____
Inventor fifteen:	_____	
Signature:	_____	Citizen of: _____



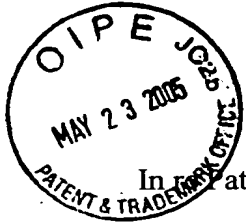
IFU

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, et al.
		Group Art Unit	2131
		Examiner Name	Not Yet Assigned
Total Number of Pages in This Submission		Attorney Docket Number	111325-291300
		Confirmation Number	8924

ENCLOSURES (check all that apply)		
<input checked="" type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input checked="" type="checkbox"/> Assignment Papers (for an Application) <input type="checkbox"/> Drawing(s) <input checked="" type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input checked="" type="checkbox"/> Petition to Request Correction of Inventorship <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input checked="" type="checkbox"/> Other Enclosure(s) (please identify below): 1. Statement by Mai Nguyen In Support of the Letter Re Correction of Inventorship Under 37 C.F.R. 1.48
Remarks		<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Marc S. Kaufman, Reg. No. 35,212 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	
Date	May 23, 2005

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
Date	Signature
_____	_____
	Typed or printed name



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Patent Application of:)	ATTENTION: Director of
)	Group Art 2131
Mai NGUYEN, <i>et al.</i>)	Examiner: Unassigned
Serial No. 10/956,121)	Group Art Unit: 2131
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR)	
MANAGING TRANSFER OF RIGHTS USING)	
SHARED STATE VARIABLES)	

U.S. Patent and Trademark Office
Customer Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

REQUEST FOR CORRECTION OF INVENTORSHIP
UNDER 37 C.F.R. 1.48(a)

Dear Sir:

In accordance with the provisions of 37 C.F.R. §1.48, Applicant hereby petitions to correct the original listing of inventors that was improperly set forth in the copy of the executed declaration filed with the instant application. Specifically, the inventorship in the above-identified application is being corrected to add the following inventor:

Mai NGUYEN
5611 Cambridge Avenue
Buena Park CA, 96021 US

Pursuant to the requirements of 37 C.F.R. 1.48(a), please find attached the Declarations of each of the above-named, newly added inventor, and a Statement wherein the new inventor states that the error in not being initially named as an inventor in the above-identified application arose without any deceptive intention on the part of either the initially-designated inventor, or the new inventor.

The Commissioner is hereby authorized to charge any fees connected with this filing which may be required now, or credit any overpayment to Deposit Account No. 19-2380 (111325-291300).

Prompt attention to this matter is requested.

Respectfully submitted,

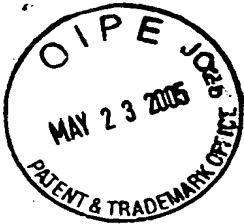
NIXON PEABODY, LLP



Marc Kaufman
Registration No. 35,212

Date: May 23, 2005

Customer No. 22204
NIXON PEABODY LLP
401 9th Street, N.W., Suite 900
Washington D.C. 20004
Telephone: (202) 585-8164
FAX: (202) 585-8080



PATENT
Attorney Docket No. 111325-291300

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	
Mai NGUYEN, <i>et al.</i>)	Examiner: Unassigned
Serial No. 10/956,121)	Group Art Unit: 2131
Filed: October 4, 2004)	
For: SYSTEM AND METHOD FOR MANAGING)		
TRANSFER OF RIGHTS USING SHARED)		
STATE VARIABLES)		

STATEMENT BY MAI NGUYEN
IN SUPPORT OF THE LETTER RE CORRECTION OF INVENTORSHIP
UNDER 37 C.F.R. 1.48

U.S. Patent and Trademark Office
Customer Window, Mail Stop
Randolph Building
Alexandria, VA 22314

Dear Sir:

Pursuant to the requirements of 37 C.F.R. 1.48(a)(2), it is hereby stated that the error in failing to list myself as inventor of the above identified application occurred without deceptive intention on my part.

Respectfully submitted,

Date: 4 / 6 / 2005

Mai Nguyen
Mai NGUYEN

5611 Cambridge Avenue
Buena Park, CA 96021 US

Security Requirements of Digital Rights Management

Mayur Kamat
Texas A&M University

Introduction

- Information runs the Web
- 100 billion \$ market for digital content on the Web – Meta Group
- Information protection : Imperative for a viable business proposition
- DRM - Digital enabling right holder's revenue system by secure content marketing & rights enforcement

Digital Rights

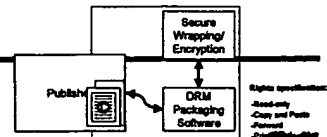
- Increasingly, the Internet is being used a source of digital content like text, audio, video, images, etc
- Ease of copying of digital content makes Internet suitable for piracy

Digital Rights

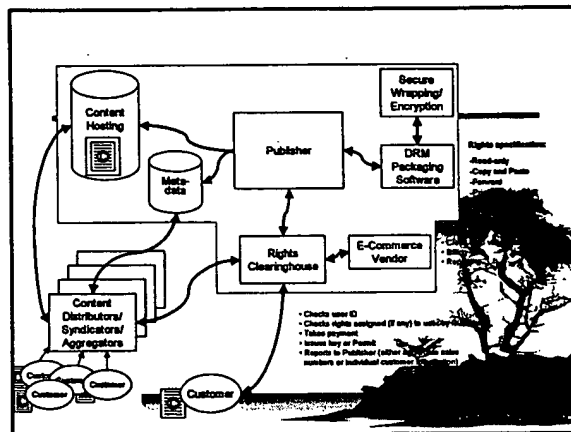
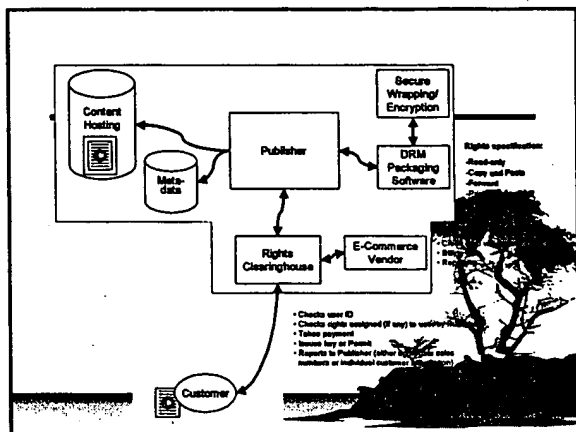
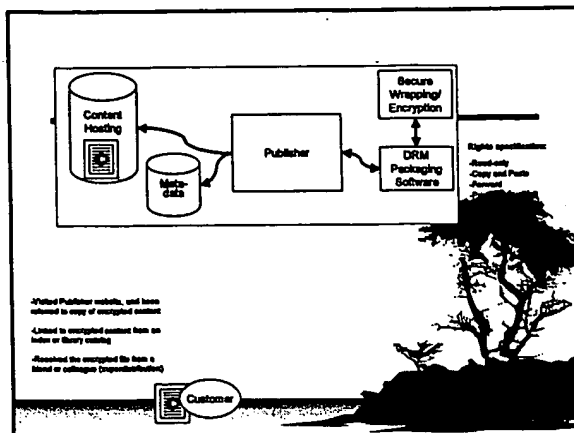
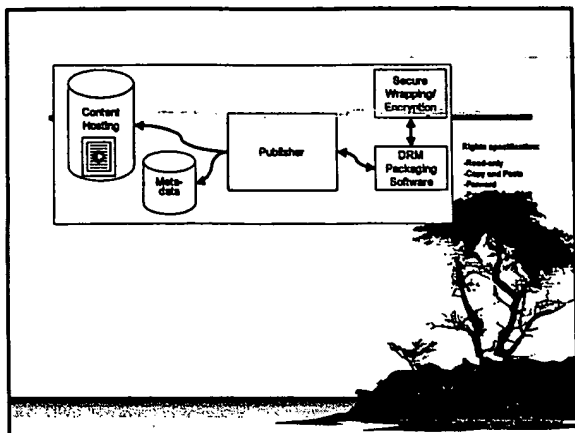
- Rights consist of:
 - Permissions - what you are allowed to do
 - Constraints - restrictions on the permissions
 - Obligations - what you have to do/propagate
 - Rights Holders - who is entitled to what
- DRM : Digital Management of Content Rights

Advantages

- Enables Digital Commerce
- Digital Content Protection
- Secure Content distribution & marketing
- Ensures content authenticity



BEST AVAILABLE COPY



Security Requirements

- Inherent security requirements
- Cater to *hostile* user over *hostile* network
- Why not apply existing security solutions?

Trusted Software Component

- Enforce interests of content owner
- User has unlimited time and resources to bypass content protection mechanism
- Software component, hence must be able to preserve its integrity in hostile environment

Trusted Software Component

□ Functions of TSC:

- Perform integrity checking
- Decrypt content
- Enforce rights
- Provides assurance to content owners

□ Requirements of TSC

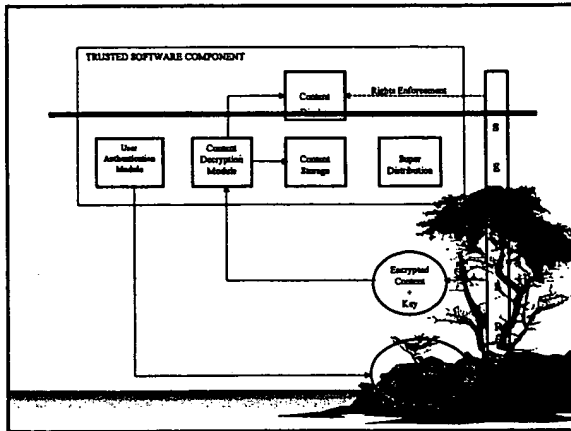
- Reverse-engineering-proof
- System Portability

TSC – How to build one?

- Use of tamper-resistant software
- Include 'entropy' along with source code
- Entropy changes, TSC compromised, application closes down
- Works like checksum error correction
- Creates programs that are difficult to analyze

Basic Elements of TRS

- Changing order of instructions
- Inserting Spoof Code
- Source level code encryption
- Using complex TRS generator – commercially available



Superdistribution

- Simple concept – do not restrict replication, restrict usage
- Turn customers into distributors
- Reach out to target customers without much efforts
- Prevent piracy
- Needs – universal ID, secure implementation

Conclusion

- Rights enforcement - Means for securing content business
- DRM has solutions for information marketing
- Traditional security measures cannot be directly implemented for DRM

Conclusion

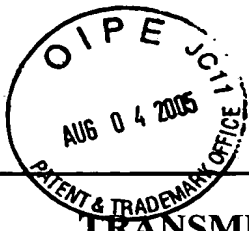
- Concentration of development efforts on the client-side software (TSC)
- Use of TRS for coding the TSC
- *Superdistribution* – New mantra for content marketing



THANK YOU



BEST AVAILABLE COPY



1 Fee

TRANSMITTAL FORM <i>(to be used for all correspondence after initial filing)</i>	Application Number	10/956,121	
	Filing Date	October 4, 2004	
	First Named Inventor	Mai NGUYEN, et al.	
	Group Art Unit	2131	
	Examiner Name	Not Yet Assigned	
Total Number of Pages in This Submission		Attorney Docket Number	111325-291300

ENCLOSURES (check all that apply)		
<input type="checkbox"/> Fee Transmittal Form <input type="checkbox"/> Fee Attached <input type="checkbox"/> Amendment / Reply <input type="checkbox"/> After Final <input type="checkbox"/> Affidavits/declaration(s) <input type="checkbox"/> Extension of Time Request <input type="checkbox"/> Express Abandonment Request <input checked="" type="checkbox"/> Information Disclosure Statement <input type="checkbox"/> Certified Copy of Priority Document(s) <input type="checkbox"/> Response to Missing Parts/ Incomplete Application <input type="checkbox"/> Response to Missing Parts under 37 CFR 1.52 or 1.53	<input type="checkbox"/> Assignment Papers (for an Application) <input type="checkbox"/> Drawing(s) <input type="checkbox"/> Declaration and Power of Attorney <input type="checkbox"/> Licensing-related Papers <input type="checkbox"/> Petition <input type="checkbox"/> Petition to Convert to a Provisional Application <input type="checkbox"/> Power of Attorney, Revocation Change of Correspondence Address <input type="checkbox"/> Terminal Disclaimer <input type="checkbox"/> Request for Refund <input type="checkbox"/> CD, Number of CD(s) _____	<input type="checkbox"/> After Allowance Communication to Group <input type="checkbox"/> Appeal Communication to Board of Appeals and Interferences <input type="checkbox"/> Appeal Communication to Group (Appeal Notice, Brief, Reply Brief) <input type="checkbox"/> Proprietary Information <input type="checkbox"/> Status Letter <input type="checkbox"/> Application Data Sheet <input type="checkbox"/> Request for Corrected Filing Receipt with Enclosures <input type="checkbox"/> A self-addressed prepaid postcard for acknowledging receipt <input type="checkbox"/> Other Enclosure(s) (please identify below):
Remarks		<input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any additional fees required or credit any overpayments to Deposit Account No. 19-2380 for the above identified docket number.

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT	
Firm or Individual name	Carlos R. Villamar Registration No. 43,224 Nixon Peabody LLP 401 9 th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	/Carlos R. Villamar, Reg.# 43,224/ Carlos R. Villamar
Date	August 4, 2005

CERTIFICATE OF MAILING OR TRANSMISSION [37 CFR 1.8(a)]	
I hereby certify that this correspondence is being:	
<input type="checkbox"/> deposited with the United States Postal Service on the date shown below with sufficient postage as first class mail in an envelope addressed to: Mail Stop _____, Commissioner for Patents, P. O. Box 1450, Alexandria, VA 22313-1450	
<input type="checkbox"/> transmitted by facsimile on the date shown below to the United States Patent and Trademark Office at (703) _____.	
Date	Signature
	_____ Typed or printed name



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:)	
Mai NGUYEN, <i>et al.</i>)	Examiner: Unassigned
)	
Application No.: 10/956,121)	Group Art Unit: 2131
)	
Filed: October 4, 2004)	
)	
For: SYSTEM AND METHOD FOR)	Confirmation No.: 8924
MANAGING TRANSFER OF RIGHTS USING)	
SHARED STATE VARIABLES)	

U.S. Patent and Trademark Office
Customer Service Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

Sir:

INFORMATION DISCLOSURE STATEMENT UNDER 37 C.F.R. § 1.97 (b)

Pursuant to 37 C.F.R. §§ 1.56 and 1.97(b), Applicants bring to the attention of the Examiner the documents listed on the attached PTO-1449. This Information Disclosure Statement is being filed before the first Office Action on the merits for the above reference application. The listed documents were cited in a communication from the International Search Authority. The International Search Report was mailed on March 2, 2005. Copies of the listed documents are attached.

It is requested that the accompanying PTO-1449 be considered and made of record in the above-identified application. To assist the Examiner, the documents are listed on the attached form PTO-1449. It is respectfully requested that an Examiner initialed copy of this form be returned to the undersigned.

The Commissioner is hereby authorized to charge any fees connected with this filing which may be required now, or credit any overpayment to Deposit Account No. 19-2380. (111325-291300).

Respectfully submitted,
NIXON PEABODY, LLP

By: /Carlos R. Villamar, Reg.# 43,224/
Carlos R. Villamar
Registration No.: 43,224

Date: August 4, 2005

NIXON PEABODY LLP
Customer No.: 22204
401 9th Street, N.W., Suite 900
Washington, DC 20004-2128
Telephone: (202) 585-8000
FAX: (202) 585-8080



Substitute for Form 1449A/P/TG		<i>Complete if Known</i>	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Mai NGUYEN, <i>et al.</i>
		Art Unit	2131
		Examiner Name	Not Yet Assigned
Sheet	1	of	1
		Attorney Docket Number	111325-291300

U.S. PATENT DOCUMENTS						
Examiner Initials*	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
		US-5,715,403		February 3, 1998	Stefik	

FOREIGN PATENT DOCUMENTS							
Examiner Initials*	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
		Country Code ³ Number ⁴	Kind Code ⁵ (if known)				

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
		Bill ROSENBLATT, et al., ContentGuard White Pages; "Integrating Content Management with Digital Rights Management - - Imperatives and Opportunities for Digital Content Lifecycles" GiantSteps Media Technology Strategies; May 15, 2005; pages 1-20; www.contentguard.com/whitepapers/CM-DRMwhitepaper.pdf ;	
		M. KAMAT; Texas A&M University; "Security Requirements for Digital Rights Management"; In The Proceedings of ISECON 2002, v 19 (San Antonio): §353b. ISSN: 1542-7382; pages 1-4; http://isedj.org/isecon/2002/353b/ISECON.2002.kamat.ppt	
		International Search Report; mailed March 2, 2005 (International Application No. PCT/US04/32588)	

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.



GiantSteps
Media Technology Strategies

200 West 57th Street, Suite 305
New York NY 10019
212 956 1045
fax: 212 258 3286
www.giantstepsmts.com

Integrating Content Management with Digital Rights Management

Imperatives and Opportunities for
Digital Content Lifecycles

By **Bill Rosenblatt and Gail Dykstra**

May 14, 2003

© 2003 GiantSteps Media Technology Strategies and Dykstra Research.
All trademarks are the property of their respective owners.

Introduction	2
Executive Summary	2
Overview of Content Management Systems and Processes	2
Overview of Digital Rights Management.....	4
Business Imperatives for Integrating Rights Management	6
Control Access During Workflow.....	6
Outsourcing	7
Downstream Use.....	7
Protection throughout Content Lifecycles	8
Modification of Rights Over Time	8
Regulatory and Business Standards.....	9
Technology Integration Opportunities	12
Content Ingestion and Metadata Creation	12
Access Control and Workflow.....	13
Distribution	14
Rights Language: The Key to Integration	17
Conclusion	19
About the Authors.....	20
About ContentGuard, Inc.	20



Introduction

Executive Summary

Many different types of organizations, including media companies, large corporations, government agencies, and others, have been adopting content management systems (CMSs) to help them organize digital content and create content-based products for their customers, employees, and partners. CMSs are intended to be control centers for entire content lifecycles, including content creation, management, production, and distribution, but the increasing complexities and interdependencies of these processes result in CMSs falling short of their ideal responsibilities.

One of the most important elements of complexity in content processes is content rights. The processes of tracking rights, controlling, and managing access to content based on rights information are increasingly necessary nowadays due to various business imperatives. Adding *persistent protection* to content is the most effective way to control and track access. Vendors of content management and related content-handling systems should integrate their solutions with persistent content protection by including rights and licensing information in the metadata that their systems track and by ensuring that their products are interoperable using standards-based persistent protection technologies. The result will be integrated content-handling systems that meet their customers' current and future needs.

In this paper, after brief introductions of content management and digital rights management terms, we explore many of the business and legal imperatives that have led to content processes that are more complex from a rights perspective. Then we discuss some of the ways in which vendors of content-handling systems should integrate rights information handling into their products in order to offer more complete solutions to customers' content management and distribution problems, at lower costs and with faster, lower-risk deployments.

We conclude by explaining how adoption of a standard Rights Expression Language (REL), such as the RELs being defined by MPEG, the Open EBook Forum, and OASIS, goes a long way towards ensuring that integration of content-processing systems through rights information is seamless, predictable, and cost-effective for all types of content producing organizations.

Overview of Content Management Systems and Processes

The term "content management" originated in the mid-1990s, and it has several different meanings in today's marketplace. At its most generic, a content management system is one that stores digital content for search, browsing, access, and retrieval by users in a workgroup or enterprise. The most prevalent types of content management systems are:

- o Digital Asset Management (DAM): systems that manage rich media assets, often including digital audio and video clips, for retrieval and repurposing in media production environments. These systems are sometimes also called Media Asset Management (MAM).
- o Web Content Management (WCM): tools that provide page template design, editorial workflow, and publishing environments specifically for Web sites and other forms of Internet content delivery.

- Enterprise Content Management (ECM): systems that facilitate management of corporate documents and other types of information for use internally as well as externally with a company's business partners, customers, regulators, and the general public.

In this paper, we will use the term Content Management System (CMS) to encompass all of the above, although we will occasionally distinguish among those three types. All of those types of systems – plus those few that straddle the boundaries among them – have common technology elements as well as common processes associated with their use. Some of the common technology elements are:

- **Database management systems** for managing metadata (information describing content) and sometimes the content itself.
- **Content storage systems**, including disk drives, storage area networks (SANs), and nearline/offline storage, particularly for storage-intensive assets such as high-resolution still images and digital video.
- **Content indexing and search** technologies, such as inverted text indexes, to promote searching and browsing of content.
- **Metadata creation** technologies, including text categorization, entity extraction, and image understanding.
- **Workflow capabilities**, which include check-in and check-out, version control, and approval routing.

Although the following is not meant to be an exhaustive list of processes that CMSs support, here are the most important ones:

- **Metadata creation:** Some types of metadata (e.g., date and time of creation, image resolution) can be automatically extracted from file formats. Other types can be inferred from the content by automated tools (e.g., categorization engines that analyze text and generate keywords). Other types of metadata, such as information about asset creators or detailed descriptions, must be entered manually. As we will see, rights metadata is another important type of metadata that can be created automatically if rights information is captured upstream from the CMS.
- **Asset storage:** A CMS can store content in a native format, an output-neutral format (e.g., XML), or a format specific to an output medium (e.g., HTML for web pages). The term **ingestion** is often used to comprise metadata creation and asset storage.
- **Workflow:** Many CMSs provide for the identification of roles (e.g., author, editor, producer) and their association with specific privileges on an asset, which could include reading, editing, or the ability to change the asset's metadata. Users can check content out for editing and check it back in again, and they can often use the CMS to send (route) content to other users, whether in an ad hoc manner or according to fixed, predefined routing schemes.
- **Search and browse:** CMSs have interfaces for users to enter query terms to search for assets whose metadata fit those terms. Many also have browsing

interfaces, where a user can scan a collection of asset descriptions (e.g., text abstracts, image thumbnails, short audio clips) to find assets of interest.

- **Distribution:** the final process that most types of CMS support is making assets available through some channel(s) outside of the domain of the CMS. This could mean publishing HTML pages to a Web site, sending files to a business partner over FTP or a syndication protocol, or persistently protecting assets with a DRM packager.

Overview of Digital Rights Management

Digital rights management (DRM) is a popular term for a field that (like content management) also came into being in the mid-1990s¹, when content providers, technology firms, and policymakers began to confront the effect of ubiquitous computer networks on the distribution of copyrighted material in digital form. There are two basic definitions of DRM: a narrow one and a broader one.

The narrower definition of DRM focuses on persistent protection of digital content. This refers to technology for protecting files via encryption and allowing access to them only after the entity desiring access (a user or a device) has had its identity authenticated and its rights to that specific type of access verified. Protection in such DRM systems is persistent because it remains in force wherever the content goes; in contrast, a file that sits on a server behind the server's access control mechanism loses its protection once it is moved from the server.

Persistent protection solutions consist of these primary technology components²:

- **Packagers** assemble content and metadata into secure files that are variously called packages, containers, envelopes, etc.³
- **Controllers** reside on client devices (PCs, music players, ebook readers, etc.). They authenticate the identities of the devices and/or users that request access to content, verify the nature of the access requested, decrypt the content, and provide the access. Controllers may also initiate financial transactions where necessary.
- Some persistent protection solutions, particularly newer ones, also include **license servers**. These create and distribute encrypted *licenses* (sometimes called tickets, permits, or vouchers) that describe rights to content, the identities of the users or devices to whom the rights are granted, and the conditions (e.g., payment) under which they are granted. DRM solutions that do not include separate license servers install rights descriptions directly into each content file at packaging time.

¹ Some observers point to the *Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment* conference in January 1994 as the birth of DRM as a discipline. The first commercial DRM solutions became available soon thereafter.

² The terminology here follows that of Rosenblatt et al., *Digital Rights Management: Business and Technology* (John Wiley & Sons, 2001).

³ Early DRM vendors trademarked names for their secure file formats, such as "Cryptolope" from IBM and "DigiBox" from InterTrust.

A broader definition of DRM encompasses everything that can be done to define, manage, and track rights to digital content. In addition to persistent protection, this definition includes these other elements:

- **Business rights (a/k/a contract rights):** an item of content can have rights associated with it by contract, such as an author's rights to a magazine article or a musician's rights to a song recording. Such rights are often very complex and have financial terms attached to them that depend on the content's use (e.g., royalties).
- **Access tracking:** DRM solutions in the broader sense can be capable of tracking access to and operations on content. Information about access is often inherently valuable to content providers, even if they do not charge for access to content.
- **Rights licensing:** content providers can define specific rights to content and make them available by contract. It is often not possible to track rights licensing by technological means: for example, a book publisher may offer language translation rights to a novel, and in general there's no technological way to ensure that the licensee's translation is either faithful or distributed according to the same terms as the original book.

Business Imperatives for Integrating Rights Management

In this section, we show how new business imperatives increase the desirability of having agile rights management functionality in enterprise content systems. As organizations turn to more sophisticated production processes and seek out revenue-generation opportunities, they require persistent content protection integrated with content management to ensure proper business practices and implement new business models.

Intellectual property is increasingly, if not exclusively, in digital form. While the nature of their products and their users differ, media companies, corporations, and other entities share similar business needs for ensuring that rights are tracked at ingestion; that access is controlled during production processes; and that protection for the content extends throughout product lifecycles. We concentrate on the shared business concerns rather than focus on uniqueness of individual digital media formats, products, and processes.

The keystone for building digital products is the recognition, respect, and tracking of the relationships between the various layers of rights, licenses, permissions and agreements that accrete to content as it moves through its lifecycle from sources to intermediaries to publishers to consumers. Often the layers of rights are so complex that companies either do not bother to process them correctly or process them through lots of expensive manual overhead.

Content management systems are widely adopted because of their capacity to handle complex, multi-layered relationships and processes, along with their ability to leverage large amounts of metadata. Until recently, the complex nature of rights-related business relationships and layered rights data stymied the inclusion of DRM technologies within content management systems. Unless the enterprise or the content owner can efficiently and effectively trust the distribution of its valued content, its CMS does not provide the full range of functions. With embedded and multi-faceted rights management technologies, CMS systems will be used to their full potential.

Ascribing, memorializing, and communicating rights should be a core competency of any full-featured content management system.

Ideally, CMSs should govern the entire content processing chain; they should demonstrate the ability to handle any combination of authenticating persons, devices, allowed uses, individual and group roles, and varying levels of permission.

Control Access During Workflow

Controlling allowed uses of digital content is a critical function of DRM technology. By pre-determining and controlling the exact use(s) for content, DRM technology extends and enhances the traditional role-based access more commonly found in content management systems.

Example: Content-rich products, such as music, video and software games, are often pirated during production processes by people working from within the company that owns the content or its production service suppliers. Elaborate password systems are time-consuming to maintain, frequently thwarted, and do not provide the level of trusted protection required by businesses with intellectual property that has long-term

revenue potential. DRM technologies provide the assurance of secured content both behind as well as beyond the corporate firewall. Not only can the content be protected during the production process, its copyright, licensing, reproduction and conditions adhere to the content throughout its use-cycle.

Example: A draft manufacturing guideline is circulated among an international standards committee and participating qualified companies. Using DRM technology, this becomes a closed circulation. The draft guidelines are in a tamper-proof format, with print-only user-rights, limited to a pre-determined timeframe, after which the draft is withdrawn and replaced by the final set of guidelines. The owner of the content, in this instance the standards committee, can withdraw, alter, or grant permissions related to the content at any time.

Outsourcing

Outsourcing of content production processes increases requirement for control of authority and authentication. Companies are even outsourcing the "family jewels"—critical customer-facing and revenue-producing applications.

Offshore processing and data-conversion service bureaus have long been a staple of trade, technical, professional and database publishers. Software and entertainment products are routinely outsourced to contract production and manufacturing services. A less traditional form of outsourcing is the use of vendor-contractor to perform core business functions.

While many firms are familiar with outsourcing data processing, IT, or web services, there is a growing trend to rely on outsourced personnel for the roles companies traditionally reserved for employees. Some companies are replacing entire departments with contracted vendor services, while others rely on strategic placement of contract or outsourced personnel to prove a "need for speed" or specialized development expertise to accelerate product and service development cycles.

The bottom line is that many of the people working on digital content products and processes do not have long-term relationships with or loyalty to the company. Security and communication become large issues and require a level of embedded knowledge within core business processes. Decisions cannot rely on 'handed-down' assumptions, knowledge of past practices, or inaccessible files.

Content management systems must accommodate increased requirements for control of authority and authentication across business boundaries.

Solid business decisions are based on "knowing about the rights," not "assuming." This is especially true when intellectual property rights are at the core of an investment decision or structuring a business model. Rights management technology ensures that information expressed in a standard format to minimize ambiguity, provide an efficient and accurate way to update operational routines, and assure appropriate levels of accountability.

Downstream Use

Rights-managed content creates new value propositions and value networks. Companies need to deliver controlled access downstream so that content can be licensed, deployed

and repurposed by business partners in accordance with the terms of agreements. For this to occur efficiently, rights information about content must be stored as part of ingestion processes.

Example: Music publishers license DRM-enabled content to online transactional or subscription services. The DRM-enabled content allows both distributors and consumers to choose from multiple fee/free business models. For example, the content could be included in both the free-play list for one-time use on multiple devices, or it could be licensed on a fee-for play use by media companies, publishers, corporate, government or institutional users. Further, with DRM-enabled content, owners may choose to permit licensees the ability to re-distribute or enter into re-publication agreements.

Content management systems should facilitate downstream product development that respects the rights of content owners.

Protection throughout Content Lifecycles

Piracy, whether of software, music, film, images, or text, costs billions of dollars each year. Besides draining corporate revenues, piracy squanders valuable company time and resources by requiring costly efforts to detect and deter theft⁴. Further, widespread piracy creates an atmosphere of distrust that can become counterproductive to developing new business models for digital content; it results in content-based products that are less user-friendly than they might otherwise be.

There are other costs associated with unauthorized uses of content as well. For example, some investment banks employ DRM for M&A documents that must be kept secret in order to maximize the values of those deals, preserve various types of business relationships, and avoid unwanted publicity. The same is true of certain types of corporate governance documents in large companies.

Fluid business models rely on an assurance that copyright, and use-rights, are protected and extended beyond content production and distribution systems. DRM-enabled protection continues throughout the distribution of the content, auditing its use and accounting for its fees and licenses.

Modification of Rights Over Time

Digital content can be transformed, reused, repurposed and renegotiated. Companies look for ways to mold their content as business needs dictate and rights, licenses, and relationships allow. Many business cases looking at return on investment (ROI) for CMS deployment are based on the proposition of "create once, reuse many times." Core to this CMS function is the system's ability to accommodate changes by updating the parameters of rights and usage as needed to accommodate new distribution models. The nature of the content and its layer of rights and relationships dictate frequency of updates.

⁴ See, for example, <http://www.mpaa.org/anti-piracy/> from the Motion Picture Association of America, or <http://www.ifpi.org/site-content/antipiracy/piracy2002.html> from the International Federation of the Phonographic Industry.

Post-hoc re-do of rights data costs money and has the potential to influence customer confidence in the integrity and accuracy of the rights and metadata; indeed it can be a disincentive for customers who insist on high standards of guaranteed accuracy and flexibility from content owners. Furthermore, the lack of ability to change access rights to content can be a serious business liability.

Example: The U.S. Supreme Court decision in *New York Times v. Tasini* (2001) compelled content industry vendors to remove or modify core research records in database archives, because creators of content in those archives were not being properly compensated. Compliance costs for vendors included additional staffing to re-code or remove records, systems development expenses, along with increased demand on customer service and marketing departments.

Example: Sensitive documents are often sent around corporations, and to business partners, via email or web posted content. Even with the increased popularity of PDF format for web posting and setting "Security" levels for email documents, recipients find ways to download files (e.g., "Save As"), thus gaining the ability to alter or distribute the file. Under normal circumstances, it is impossible to change access rights to a file once it has been "detached" from a central repository (CMS or file server)⁵.

Change happens, especially within the world of digital content. Corporate reorganizations, mergers, and acquisitions change content licenses and determine who within the organization can access, change, or repurpose content. Multinationals and multi-product corporations have multiple product lines and business models that support internal competing organizations and product strategies. Efficiencies are gained through central content processing functions (ingestion, storage, workflow, search and distribution) that ensure that rights, licenses, and permissions remain attached to the content.

Content management systems should facilitate the strengths of digital rights management to foster collaboration and adaptable business models.

Collaborative business-value chains are built on trust. Rights management technology facilitates collaboration, creating the 'trusted environment' needed for collaboration by persistently protecting critical intellectual property beyond the boundaries of business processes and corporate organizations.⁶

Example: A boutique international consulting company leading large government and industry projects uses DRM technology to seal its project documents and control and track its critical intellectual property. With the assurance its intellectual property is protected beyond firewalls, the boutique firm enters into a collaboration agreement with another consulting company that is, in other circumstances, the boutique's competition.

Regulatory and Business Standards

Integrity, authentication, security, privacy and accountability are 'watchwords' for new legislative and regulatory standards. Privacy legislation demands stringent assurance of

⁵ However, some vendors of DRM solutions for corporate applications support the ability to revoke rights to a file even after it has been sent to other users by email or other means.

⁶ CIOs have identified "lack of trust" as the #1 factor inhibiting inter-company collaboration. See, for example, Paul, Lauren Gibbons, "Suspicious Minds," *CIO Magazine*, January 15, 2003.

security.⁷ Conversely, security legislation requires assurances of accuracy and authenticity. Public confidence, investors, and stockholders depend on secure and accountable sharing of financial and governance data

Example: Audited financial statements must preclude tampering while providing more timely, accurate and detailed accounting. Financial reporting and securities research require transparency and personal accountability of corporate offices and boards.⁸

Example: HIPPA regulations mandate new levels for privacy and authentication for document management in healthcare institutions and the medical community⁹.

Example: Warranties and liability requirements demand strict assurances that the latest, most comprehensive, and appropriate instructions, product information and warning of potential hazards are in the hands of the users.

Integrated DRM-CMS solutions can offer corporations, public sector institutions and regulated industries enterprise-wide assurance that content and document operations comply with current regulatory regimes, accountability, privacy, and security legislation. Tracking submissions to government bodies is of particular importance to businesses operating in a regulatory environment. Regulatory requirements are subject to change. Compliance can be mandated within a short timeframe with significant consequences for not being able to meet new, and often more stringent, regulatory or administrative standards for business operations.

Companies doing business on a global basis, or those expanding into new jurisdictions, must meet new regulatory requirements. This may call for an entirely different, and more complex, set of jurisdictional rights to be part of the content property. This is a particular concern for companies doing business in the European Union where privacy and database legislation call for significantly different content rights.

With scalable and integrated CMS-DRM technology, organizations can more rapidly respond to change.

Content management systems must ensure enterprise-wide compliance with regulatory and legislative requirements, including controlling and tracking use.

Many of the business requirements for DRM-empowered Content Management Systems can be expressed as gains in productivity. These include:

- o Elimination of bottlenecks in manual and paper-file dependent systems.
- o Decreasing "hands-on" personnel costs in data entry and updating records on rights and permissions.

⁷ Privacy concerns affect consumer confidence and therefore can have a negative effect on the market for digital content. As an example, news reports about the security breach that exposed 8 million credit card account numbers add fuel to consumer concerns about privacy. Governments often respond by legislating new layers of regulation on privacy, e-commerce and credit reporting. (See, for example, Jonathan Krim, "8 million credit accounts exposed," Washington Post, February 19, 2003, p. E01)

⁸ *Sarbanes-Oxley Act 2002*, SEC and stock exchange reforms.

⁹ Key provisions of the *Health Insurance Portability and Accountability Act of 1996* went into effect on April 14, 2003.

- o Maximizing internal skills through greater specialization and flexibility in staffing choices.

Content-driven businesses can enjoy productivity improvements from tightly integrating digital rights, user-action permissions, and auditable tracking technology within core CMS technology.

The integration of DRM controls increases the ROI for adoption and deployment of CMS solutions for content industries by accelerating product development cycles and eliminating lengthy delays because of missing rights and licenses. The ability to rely on post-CMS control of users' rights permits a wide array of product specialization to meet customer requirements and affords added flexibility in meeting market demands. Content security, reduction of legal liability, and increased customer confidence are additional benefits from integrated DRM and CMS technologies.



Technology Integration Opportunities

Many of the business imperatives described above in this paper lead to ways in which vendors of CMSs and other content-handling systems can improve their value through integration with rights management functions. Interoperation of CMSs with rights management requires two primary steps:

1. Store standards-based metadata that describes rights with content and other metadata in the CMS.
2. Provide hooks in the CMS that enable it to interoperate with software components that interpret rights metadata, provide persistent protection, manage contract rights and rights licensing processes, and so on.

In this section, we look at typical content processes that are handled by CMSs and focus on how integrated rights management adds value to them.

Content Ingestion and Metadata Creation

The metadata creation process is the nexus for integration between rights management systems and CMSs that satisfies business concerns such as those mentioned above. As with all other types of metadata, it is most desirable to avoid having to rely on manual input for creating rights metadata: In addition to adding undesirable overhead to business processes, relying on manual input introduces opportunities for errors and inconsistencies in metadata.

The metadata creation process is the most crucial point of integration between rights management systems and CMSs.

The simplest way to automate the creation of rights metadata at ingestion time is to program the CMS to use default rights metadata settings according to company policy – for example, to assume, unless otherwise specified, that the company holds copyrights on all assets. A more advanced variation on this idea is to set up the CMS to infer rights metadata according to rules that take into account the type of content, the type of content creation/editing tool from which the asset is being ingested into the CMS, the user doing the ingesting, or the point in a workflow routing. In cases where no automation is possible, the CMS vendor would integrate a template-based rights editor into the ingestion process, so that a user can fill in the appropriate rights on a case-by-case basis.

Example: a magazine publisher, which stores copyright info in its CMS, creates all text content in-house but obtains all images from freelancers or other external sources. In this case, if the user is a text editor who is ingesting text items through a text creation tool such as Quark CopyDesk, then the CMS should infer that copyright on those items belongs to the publisher and set the rights metadata accordingly. For a photo editor who is ingesting images through Adobe Photoshop, the CMS should prompt the editor for information about the external source of a photo.

A company can achieve even more advanced ways of automating the creation of rights metadata in a CMS if it uses systems for tracking business rights, such as contracts with content creators and other sources of content. An example of this is shown in Figure 1.

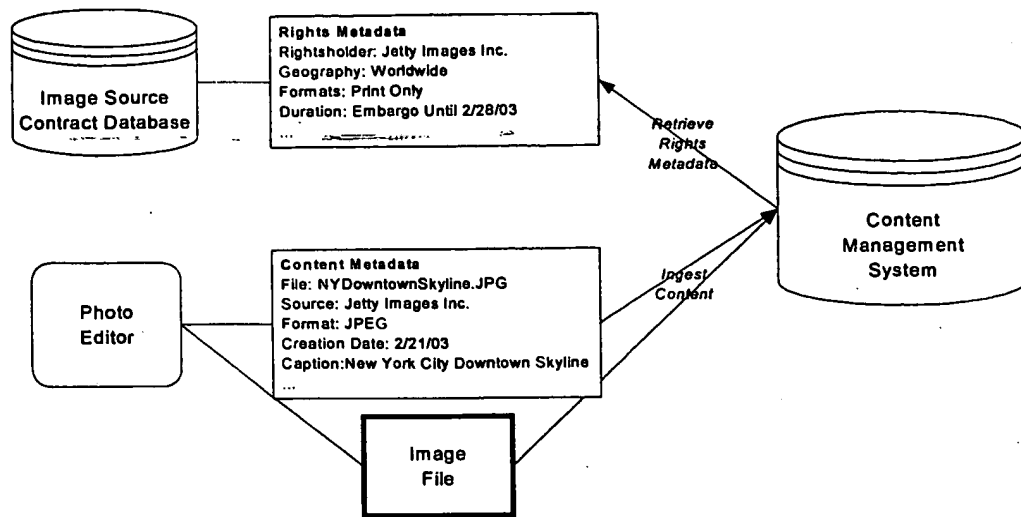


Figure 1: Integrating retrieval of rights metadata with ingestion of a digital image into a CMS.

In the scenario of Figure 1, the magazine publisher has a system for keeping track of freelance photographers or stock image agencies; many magazine publishers have such systems in the form of small databases on PCs. Systems for tracking freelancers sometimes also track information from the publisher's contract with each freelancer, covering such elements as the terms under which the publisher can redistribute the images it licenses. Terms can include restrictions by time (e.g., duration or embargo date), geography (e.g., U.S. only), and medium (e.g., print only, not electronic).

It is beneficial to integrate such rights databases with CMSs so that, as Figure 1 shows, rights information associated with the content sources can go into the CMS as rights metadata at ingestion time.

Access Control and Workflow

The above example had to do with a scenario involving DAM and editorial and production workflow at a media company. ECM systems used within large corporations depend more on the identities and roles of users, both internal to the company and at the company's external business partners, to determine rights. That is because the "consumers" of information stored in corporate ECM systems are employees or business partners of the corporation, whose identities are known and authenticated.

In ECM systems, rights metadata can be supersets of the following types of information typically found in corporate systems:

- o File access permissions, such as read, write, and delete.
- o Resource access control lists of the type found in advanced operating systems and document management systems.
- o User and group (role) identifiers, whether local to a single system or network identities, authenticated by passwords, biometrics, or other means.

The means by which a user establishes identity to a PC, server, or network is another important foundation for integrating rights information with content management.

ECM systems can use rights metadata in integrating with extranet portals that automatically provide selected information to business partners or the general public according to the "real time enterprise" model. Such systems can use identity and other rights metadata to determine what content to make available to which users and under what conditions. When integrated with persistent protection, those access conditions can hold for authenticated users even when they copy content away from the portal (e.g., onto the hard drives of PCs). Other types of metadata, such as keywords generated by a categorization tool, can help the portal system place each content item in the appropriate place on the Web site. All this can be done automatically, without user intervention.

Integration of content management with user and role identity is just as important in certain media industry applications as it is in corporate applications. For example, consider check-in and check-out functions that are common in production workflow and DAM systems in use at media companies. Once a user has checked content out of a workflow or DAM system, there is no telling what could be done with it. In the media industry, one of the "dirty little secrets" is that a lot of professional piracy occurs before products are released – that is, piracy is done (or at least facilitated) either by personnel inside a media company or by its business partners, such as post-production houses or mastering labs.

To help combat this problem, content creation/editing tool vendors can provide "trusted tools" that interoperate with persistent protection schemes. Tools can incorporate DRM controller (see p. 4) functions that use rights metadata to determine allowable operations on content, decrypt it, and provide that level of access. For example, only a sufficiently privileged user would be able to use a "Save As" function within a content editing tool. The tool would read rights metadata that was stored in the CMS from whence the asset came and packaged with the content (or contained in a separate license). As a backup to such trusted tools, the CMS could track and report on all content usage, so that any suspicious activity can be identified.

Distribution

Various CMS vendors have made claims that their products function equally well for managing content internally to an organization as for distributing content to customers and business partners, but in reality, content management and distribution remain largely disparate steps in content lifecycles. WCM systems, and many ECM systems, often function as publishing platforms for Web sites rather than as internal content management platforms, while DAM systems rarely touch distribution processes. As a result, companies must often integrate separate systems for managing and publishing content.

Rights metadata should be a key element in the integration of content management and distribution systems.

In the classic B-to-C DRM scenario (see p. 4), a DRM packaging tool takes content files and assorted metadata, and it creates packages that are decrypted on the client side by controller hardware or software. DRM packaging applications typically have user interfaces for loading content and specifying rights to that content. A better solution would

be to store rights information directly in a CMS and have the DRM packager simply read it from there through database queries. Simple rights metadata could be stored in a CMS directly. More complex rights information, especially that which has to do with business rights or rights licensing terms (see p.5), would more typically be stored in a separate repository, and the CMS would merely store a unique identifier that links to the appropriate entry in that repository.

A more sophisticated integration between content management and DRM-based distribution is possible at media companies, which often maintain "product catalog" systems that contain product metadata. Product metadata overlaps with content metadata, but it is distinct, because a given item of content can appear in more than one different product. Different products can be intended for different types of customers (subscribers, one-time purchasers, free trial users, etc.) under different usage terms (unlimited, 30 days only, etc.), even though they may all include the same content.

Although few product catalog systems at media companies include this level of detail today, they will need to in the future as media companies put out greater and greater varieties of products based on their content. A further (and admittedly more extreme) need is to define and track products targeted to individual consumers, which implies a requirement to integrate content management and distribution systems with CRM (customer relationship management) and other types of customer databases, in order to define content rights in terms of individual identities instead of user types.

Example: an online music distributor has several different types of offers for its catalog of music tracks, including a monthly subscription to the entire catalog, a 7-day free trial of the monthly subscription, and paid downloads of individual tracks. A product catalog system should feed a DRM packaging application information about rights to music files that customer's request.

As Figure 2 shows, rights metadata in both product catalog and DAM systems can feed directly into DRM packagers to achieve seamless integration with distribution without requiring manual overhead.

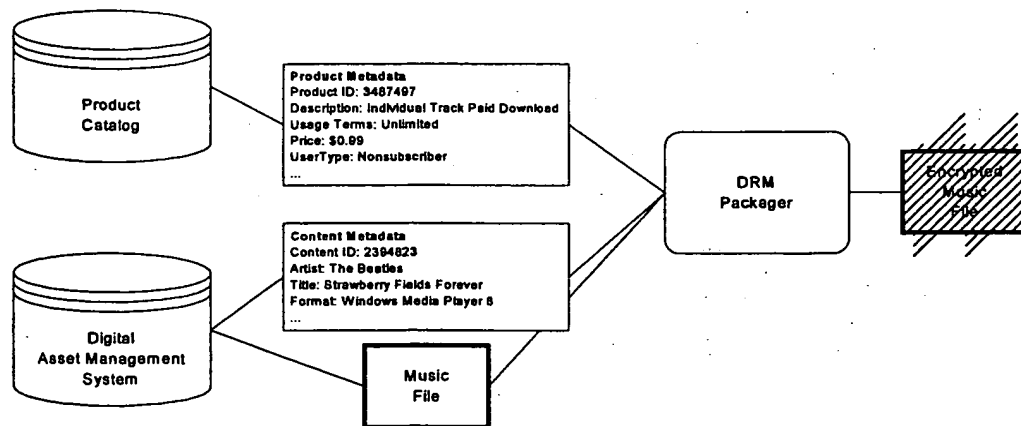


Figure 2: Integrating product and content metadata in a DRM packaging operation.

Note that rights-controlled distribution is not limited to persistent protection-based DRM systems. Many media companies feed their content to distribution partners under terms

that are covered by contract and therefore need not be enforced through persistent protection.

The simplest way to set up multiple content feeds is via file transfer protocol (FTP). A given content provider can have many different FTP feeds, each of which includes a different subset of the company's content; the ultimate example of this would be a news wire service, which has many different service levels for its subscribers. In this case, information about distribution partners can be linked with rights metadata from product catalog-type systems, which describe different levels of content offerings, to automate the process of putting the appropriate content in various FTP directories for distribution partners to pick up. The ICE protocol¹⁰ provides ways of automating this process and describing rights and licensing terms, though without providing a persistent protection mechanism.

Example: In the magazine publishing example above, rights restrictions on images that derive from contracts with outside content sources result in rights metadata, stored in the CMS, which in turn governs distribution process so that each customer or distribution partner only sees the content to which they are entitled.

As Figure 3 shows, the magazine publisher from Figure 1 might have a Web publishing system that takes content automatically from the CMS and uses it to maintain the magazine's Web site. The Web publishing system would not use any images with rights metadata set to exclude online distribution.

¹⁰ The Information and Content Exchange protocol from IDEAlliance; see <http://www.icestandard.org>.

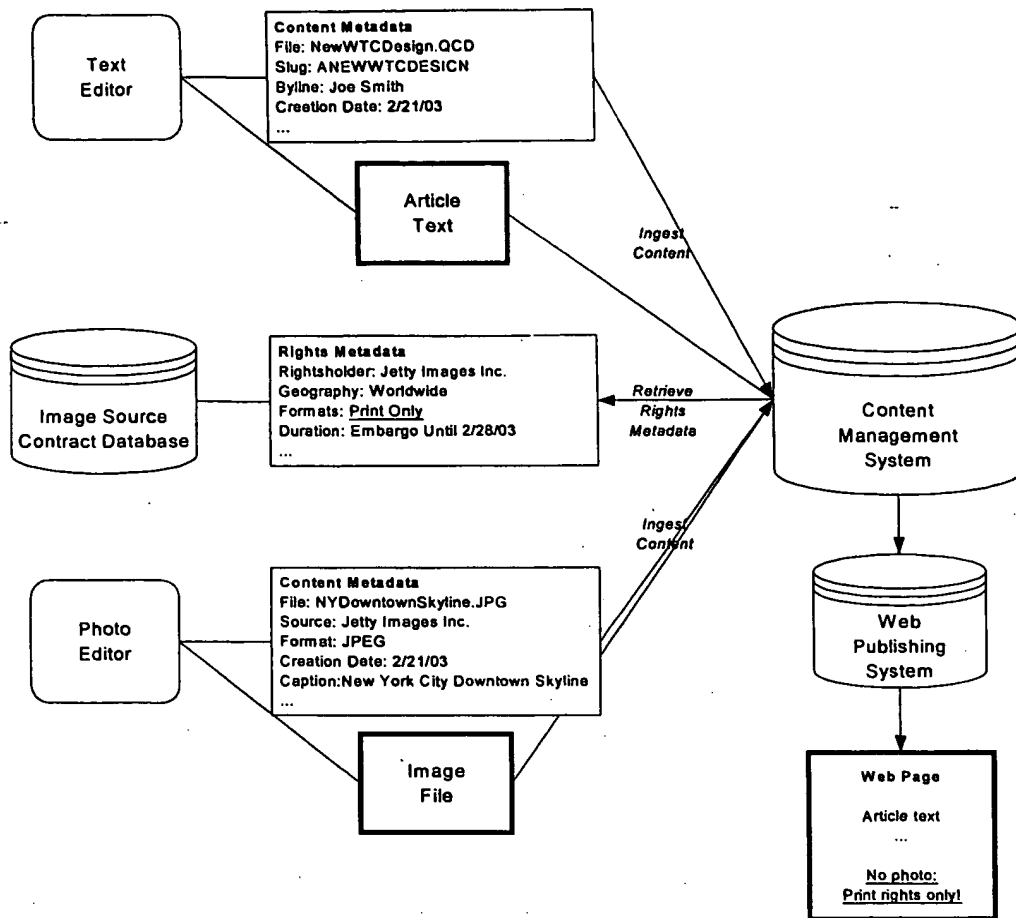


Figure 3: Integrating content and rights metadata through publishing process to automatically ensure that rights are respected.

Rights Language: The Key to Integration

In the above examples, we have seen several different types of systems that all depend on the same types of rights metadata to achieve the types of automated process integration mentioned:

- Content creation and editing tools
- Content management systems – DAM and ECM
- Web publishing systems, including corporate portal systems
- Product catalog systems
- CRM and customer tracking systems
- Content distribution systems

As we noted on p. 12, integrating all of these types of systems with respect to rights-based processes would be much easier and less costly if every one of these systems had two things:

1. A common understanding of content rights and related information: that is, the same way of specifying, storing, and communicating rights information.
2. Standard ways of interoperating with software components that can interpret rights information and act on it in consistent ways – including persistent protection of content; authenticated access to protected content; tracking of content access; and facilitation of financial transactions or other forms of consideration that enable content access according to license terms.

The way to ensure that such integration can take place is to specify content rights and related information in a standard Rights Expression Language (REL). One such REL, XrML from ContentGuard, Inc, has been used as the basis for several standards bodies' REL definitions, including the Moving Picture Experts Group (MPEG), the Open eBook Forum (OeBF), and the Organization for the Advancement of Structured Information Standards (OASIS). XrML derives from research done in the mid-1990s at Xerox PARC by Dr. Mark Stefik into empirical types of content rights, information necessary to associate with content rights, and ways of expressing all such information with precision and non-ambiguity¹¹.

Use of a standard Rights Expression Language provides many benefits to content owners. It ensures that the semantics of rights information remains consistent across systems without having to rely on "lowest common denominator" mappings among multiple types of rights information, thereby lowering both the cost of systems integration and the risk of legal trouble through misinterpretation of rights information.

For CMSs and various other types of content processing tools, use of an REL also makes these components more valuable by making them easier to integrate into highly automated end-to-end content lifecycle solutions. Amid all of today's claims of integrated digital media solutions, very few truly end-to-end solutions are available without requiring millions of dollars of risky custom development, much of which is spent on patching together isolated systems. An REL provides a good part of the interoperability "glue" that makes integration faster and cheaper, while also helping content owners protect their technology investments by ensuring component-level compatibility as the capabilities of CMSs and other systems grow over time.

¹¹ See, for example, Stefik's paper "Letting Loose the Light: Igniting Commerce in Electronic Publication," in his book, *Internet Dreams: Archetypes, Myths, and Metaphors* (MIT Press, 1996).

Conclusion

We have described the increasing complexity of content processes in various types of business environments, ranging from media companies to large corporations to government institutions. We have shown how persistent content protection and management of rights information are increasingly crucial to ensuring that business processes comply with contractual and regulatory demands, facilitate the implementation of new content-based business models, and protect valued corporate digital content both within the enterprise and with business partners.

We have also discussed various ways in which vendors of CMSs and other content-processing systems should integrate rights information, persistent protection schemes, and other rights processing components into their products. We noted that incorporating support for a standard Rights Expression Language goes a long way towards making such integration less costly, time-consuming, and risky by giving all components a common understanding of rights semantics as well as a common syntax for expressing them.

Ever since network-based distribution of digital content became a reality, content owners have been searching – mostly in vain – for cost-effective content management and distribution solutions that are truly integrated, enable them to pursue new business models and keep up with the latest technology, and ensure that content rights are respected for both legal and economic reasons. Standard Rights Expression Languages will help make this search finally come to a successful end.

About the Authors

Bill Rosenblatt is president of GiantSteps Media Technology Strategies, a management consultancy focused on the content industries (www.giantstepsmts.com). Bill has 20 years of experience in technology architecture, business development, and marketing; publishing; new media; and online education. His expertise spans digital media technologies such as content management, digital rights management, streaming media, and publishing systems. Bill is the author of several books, including *Digital Rights Management: Business and Technology* (John Wiley & Sons, 2001), and he is the publisher of the newsletter DRM Watch (www.drmwatch.com) and the producer of DRM conferences for Seybold Seminars.

Contact:

GiantSteps Media Technology Strategies
200 West 57th St., Suite 305
New York, NY 10019
(212) 956-1045
billr@giantstepsmts.com
<http://www.giantstepsmts.com/>

Gail Dykstra is president of Dykstra Research, a consultancy providing licensing services and product development in digital rights management to publishers and software companies. She creates content licensing and business development strategies for information-related products and companies. Dykstra Research helps companies protect their content rights, acquire new relationships, and license the content they need. It helps vendors of digital rights management technology understand customer requirements within corporate and public sector information services. Gail is the author of articles on digital rights management and public access in Information Today (www.infotoday.com), a frequent speaker at conferences, and organizer of seminars on digital rights.

Contact:

Dykstra Research
10550 NE 29th Street, Apt. E
Bellevue, WA 98003
(425) 827-3380
gail.dykstra@dykstraresearch.com

**White paper commissioned by
ContentGuard, Inc.**



ContentGuard, Inc. is driving the standard for interoperability in Digital Rights. The company's broad foundation portfolio of DRM system patents, and its Rights Expression Language, XrML (eXtensible rights Markup Language) were originally developed at the Xerox Palo Alto Research Center (PARC). ContentGuard is driving the adoption of XrML as the industry standard for access and usage rights. XrML has been selected as the basis for the Moving Picture Expert's Group (MPEG) and the Open eBook Forum (OeBF) Rights Expression Language, and has been contributed to the Organization for the Advancement of Structured Information Systems (OASIS) Rights Language Technical Committee. Launched in April 2000, ContentGuard conducts its operations in Bethesda, MD, and El Segundo, CA. The company is owned by Xerox Corporation (NYSE:XRX), with Microsoft Corporation (NASDAQ: MSFT) holding a minority position.

For more information, please visit www.contentguard.com.

PATENT COOPERATION TREATY

PCT

INTERNATIONAL SEARCH REPORT

(PCT Article 18 and Rules 43 and 44)

Applicant's or agent's file reference 111325-291401	FOR FURTHER ACTION <small>see Form PCT/ISA/220 as well as, where applicable, item 5 below.</small>	
International application No. PCT/US04/32588	International filing date (<i>day/month/year</i>) 04 October 2004 (04.10.2004)	(Earliest) Priority Date (<i>day/month/year</i>)
Applicant CONTENTGUARD HOLDINGS, INC.		

This international search report has been prepared by this International Searching Authority and is transmitted to the applicant according to Article 18. A copy is being transmitted to the International Bureau.

This international search report consists of a total of 3 sheets.

It is also accompanied by a copy of each prior art document cited in this report.

1. Basis of the Report

a. With regard to the language, the international search was carried out on the basis of the international application in the language in which it was filed, unless otherwise indicated under this item.

The international search was carried out on the basis of a translation of the international application furnished to this Authority (Rule 23.1(b)).

b. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, see Box No. I.

2. Certain claims were found unsearchable (See Box No. II)

3. Unity of invention is lacking (See Box No. III)

4. With regard to the title,

the text is approved as submitted by the applicant.

the text has been established by this Authority to read as follows:

5. With regard to the abstract,

the text is approved as submitted by the applicant.

the text has been established, according to Rule 38.2(b), by this Authority as it appears in Box No. IV. The applicant may, within one month from the date of mailing of this international search report, submit comments to this Authority.

6. With regard to the drawings,

a. the figure of the drawings to be published with the abstract is Figure No. 1

as suggested by the applicant.

as selected by this Authority, because the applicant failed to suggest a figure.

as selected by this Authority, because this figure better characterizes the invention.

b. none of the figures is to be published with the abstract.

Form PCT/ISA/210 (first sheet) (January 2004)

BEST AVAILABLE COPY

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US04/32588

<p>A. CLASSIFICATION OF SUBJECT MATTER IPC(7) : G06F 1/14, 13/372, 12/14 US CL : 710/200; 340/5.2,5.74; 380/4,5,201,203; 711/100,101,154,163; 713/200; 705/57,59 According to International Patent Classification (IPC) or to both national classification and IPC</p>														
<p>B. FIELDS SEARCHED</p> <p>Minimum documentation searched (classification system followed by classification symbols) U.S. : Please See Continuation Sheet</p> <p>Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched</p> <p>Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) Please See Continuation Sheet</p>														
<p>C. DOCUMENTS CONSIDERED TO BE RELEVANT</p> <table border="1"> <thead> <tr> <th>Category *</th> <th>Citation of document, with indication, where appropriate, of the relevant passages</th> <th>Relevant to claim No.</th> </tr> </thead> <tbody> <tr> <td>X</td> <td>US 5,715,403 (STEFIK) 3 February 1998 (03.02.1998), column 6, line 30 -column 9, line 16; column 17, line 65 - column 18, line 65.</td> <td>1-36</td> </tr> <tr> <td>A</td> <td>ContentGuard White Pages. Integrating Content Management with Digital Rights Management -- Imperatives and Opportunities for Digital Content Lifecycles. http://www.contentguard.com/whitepapers/CM-DRMwhitepaper.pdf. 15 May 2003 (15.05.2003).</td> <td>1-36</td> </tr> <tr> <td>O</td> <td>Kamat, M. Security Requirements for Digital Rights Management. In The Proceedings of ISECON 2002, v 19 (San Antonio): §353b. ISSN: 1542-7382. http://isedj.org/isecon/2002/353b/ISECON.2002.Kamat.ppt</td> <td>1-36</td> </tr> </tbody> </table>			Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.	X	US 5,715,403 (STEFIK) 3 February 1998 (03.02.1998), column 6, line 30 -column 9, line 16; column 17, line 65 - column 18, line 65.	1-36	A	ContentGuard White Pages. Integrating Content Management with Digital Rights Management -- Imperatives and Opportunities for Digital Content Lifecycles. http://www.contentguard.com/whitepapers/CM-DRMwhitepaper.pdf . 15 May 2003 (15.05.2003).	1-36	O	Kamat, M. Security Requirements for Digital Rights Management. In The Proceedings of ISECON 2002, v 19 (San Antonio): §353b. ISSN: 1542-7382. http://isedj.org/isecon/2002/353b/ISECON.2002.Kamat.ppt	1-36
Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.												
X	US 5,715,403 (STEFIK) 3 February 1998 (03.02.1998), column 6, line 30 -column 9, line 16; column 17, line 65 - column 18, line 65.	1-36												
A	ContentGuard White Pages. Integrating Content Management with Digital Rights Management -- Imperatives and Opportunities for Digital Content Lifecycles. http://www.contentguard.com/whitepapers/CM-DRMwhitepaper.pdf . 15 May 2003 (15.05.2003).	1-36												
O	Kamat, M. Security Requirements for Digital Rights Management. In The Proceedings of ISECON 2002, v 19 (San Antonio): §353b. ISSN: 1542-7382. http://isedj.org/isecon/2002/353b/ISECON.2002.Kamat.ppt	1-36												
<p><input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.</p>														
<p>* Special categories of cited documents:</p> <table border="0"> <tr> <td>"A" document defining the general state of the art which is not considered to be of particular relevance</td> <td>"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</td> </tr> <tr> <td>"E" earlier application or patent published on or after the international filing date</td> <td>"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</td> </tr> <tr> <td>"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)</td> <td>"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</td> </tr> <tr> <td>"O" document referring to an oral disclosure, use, exhibition or other means</td> <td>"&" document member of the same patent family</td> </tr> <tr> <td>"P" document published prior to the international filing date but later than the priority date claimed</td> <td></td> </tr> </table>			"A" document defining the general state of the art which is not considered to be of particular relevance	"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	"E" earlier application or patent published on or after the international filing date	"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	"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)	"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	"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family	"P" document published prior to the international filing date but later than the priority date claimed			
"A" document defining the general state of the art which is not considered to be of particular relevance	"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													
"E" earlier application or patent published on or after the international filing date	"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													
"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)	"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													
"O" document referring to an oral disclosure, use, exhibition or other means	"&" document member of the same patent family													
"P" document published prior to the international filing date but later than the priority date claimed														
<p>Date of the actual completion of the international search 07 February 2005 (07.02.2005)</p>		<p>Date of mailing of the international search report 02 MAR 2005</p>												
<p>Name and mailing address of the ISA/US Mail Stop PCT, Attn: ISA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703) 305-3230</p>		<p>Authorized officer Andrew Caldwell Telephone No. 703-305-3900</p>												

BEST AVAILABLE COPY

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US04/32588

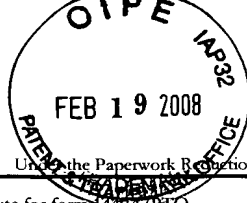
BEST AVAILABLE COPY

Continuation of B. FIELDS SEARCHED Item 1:

US: 710/200; 340/5.2,5.74; 380/4,5,201,203; 711/100,101,154,163; 713/200; 705/57,59
IPC: G06F 1/14, 13/372, 12/14; H04L

Continuation of B. FIELDS SEARCHED Item 3:

[EAST]: Stefik, rights, management, meta, digital media, state variable, transfer, content usage, state, DRM
[INSPEC]: DRM, rights, management, meta, digital media, transfer, content usage



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

Substitute for form PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Xin Wang et al.
		Art Unit	3621
		Examiner Name	West, Thomas C.
Sheet 1 of 9		Attorney Docket Number	111325/291300

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number - Kind Code ² (if known)				
	1	US 20010009026 A1		07-19-2001	Terao et al.	
	2	US 20010011276 A1		08-02-2001	Durst Jr. et al.	
	3	US 20010014206 A1		08-16-2001	Artigalas et al.	
	4	US 20010037467 A1		11-01-2001	O'Toole Jr. et al.	
	5	US 20010039659 A1		11-08-2001	Simmons et al.	
	6	US 20020001387 A1		01-03-2002	Dillon	
	7	US 20020035618 A1		03-21-2002	Mendez et al.	
	8	US 20020044658 A1		04-18-2002	Wasilewski et al.	
	9	US 20020056118 A1		05-09-2002	Hunter et al.	
	10	US 20020069282 A1		06-06-2002	Reisman	
	11	US 20020099948 A1		07-25-2002	Kocher et al.	
	12	US 20020127423 A1		09-12-2002	Kayanakis	
	13	US 20030097567 A1		05-22-2003	Terao et al.	
	14	US 20040052370 A1		03-18-2004	Katznelson	
	15	US 20040172552 A1		09-02-2004	Boyles et al.	
	16	US 4,159,468		06-26-1979	Barnes et al.	
	17	US 4,200,700		04-29-1980	Mäder	
	18	US 4,361,851		11-30-1982	Asip et al.	
	19	US 4,423,287		12-27-1983	Zeidler	
	20	US 4,429,385		01-31-1984	Cichelli et al.	
	21	US 4,621,321		11-04-1986	Boebert et al.	
	22	US 4,736,422		04-05-1988	Mason	
	23	US 4,740,890		04-26-1988	William	
	24	US 4,796,220		01-03-1989	Wolfe	
	25	US 4,816,655		03-28-1989	Musyck et al.	
	26	US 4,888,638		12-19-1989	Bohn	
	27	US 4,937,863		06-26-1990	Robert et al.	
	28	US 4,953,209		08-28-1990	Ryder et al.	
	29	US 4,977,594		12-11-1990	Shear	
	30	US 5,014,234		05-07-1991	Edwards	
	31	US 5,129,083		07-07-1992	Cutler et al.	
	32	US 5,138,712		08-11-1992	Corbin	
	33	US 5,174,641		12-29-1992	Lim	
	34	US 5,204,897		04-20-1993	Wyman	
	35	US 5,247,575		09-21-1993	Sprague et al.	
	36	US 5,260,999		11-09-1993	Wyman	
	37	US 5,276,444		01-04-1994	McNair	
	38	US 5,291,596		03-01-1994	Mita	
	39	US 5,293,422		03-08-1994	Loiacono	

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

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

Substitute for form 1449A/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Xin Wang et al.
		Art Unit	3621
		Examiner Name	West, Thomas C.
Sheet	2	of	9
		Attorney Docket Number	111325/291300

U.S. PATENT DOCUMENTS					
Examiner Initials*	Cite No. ¹	U.S. Patent Document Number - Kind Code ² (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
	40	US 5,335,275	08-02-1994	Millar et al.	
	41	US 5,337,357	08-09-1994	Chou et al.	
	42	US 5,386,369	01-31-1995	Christiano	
	43	US 5,453,601	09-26-1995	Rosen	
	44	US 5,485,577	01-16-1996	Eyer et al.	
	45	US 5,504,816	04-02-1996	Hamilton et al.	
	46	US 5,530,235	06-25-1996	Stefik et al.	
	47	US 5,535,276	07-09-1996	Ganesan	
	48	US 5,557,678	09-17-1996	Ganesan	
	49	US 5,629,980	05-13-1997	Stefik et al.	
	50	US 5,636,346	06-03-1997	Saxe	
	51	US 5,638,443	06-10-1997	Stefik et al.	
	52	US 5,708,709	01-13-1998	Rose	
	53	US 5,715,403	02-03-1998	Stefik	
	54	US 5,745,879	04-28-1998	Wyman	
	55	US 5,764,807	06-09-1998	Pearlman et al.	
	56	US 5,765,152	06-09-1998	Erickson	
	57	US 5,787,172	07-28-1998	Arnold	
	58	US 5,790,677	08-04-1998	Fox et al.	
	59	US 5,812,664	09-22-1998	Bernobich et al.	
	60	US 5,825,876	10-20-1998	Peterson	
	61	US 5,825,879	10-20-1998	Davis	
	62	US 5,838,792	11-17-1998	Ganesan	
	63	US 5,848,154	12-08-1998	Nishio et al.	
	64	US 5,848,378	12-08-1998	Shelton et al.	
	65	US 5,850,433	12-15-1998	Van Oorschot et al.	
	66	US 5,915,019	06-22-1999	Ginter et al.	
	67	US 5,917,912	06-29-1999	Ginter et al.	
	68	US 5,933,498	08-03-1999	Schneck et al.	
	69	US 5,940,504	08-17-1999	Griswold	
	70	US 5,982,891	11-09-1999	Ginter et al.	
	71	US 5,987,134	11-16-1999	Shin et al.	
	72	US 5,999,624	12-07-1999	Hopkins	
	73	US 6,006,332	12-21-1999	Rabne et al.	
	74	US 6,020,882	02-01-2000	Kinghorn et al.	
	75	US 6,047,067	04-04-2000	Rosen	
	76	US 6,073,234	06-06-2000	Kigo et al.	
	77	US 6,091,777	07-18-2000	Guetz et al.	
	78	US 6,112,239	08-29-2000	Kenner et al.	

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

10886656.1

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

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Complete if Known		
				Application Number	10/956,121	
Sheet 3 of 9				Filing Date	October 4, 2004	
				First Named Inventor	Xin Wang et al.	
				Art Unit	3621	
				Examiner Name	West, Thomas C.	
				Attorney Docket Number	111325/291300	

U.S. PATENT DOCUMENTS						
Examiner Initials ¹	Cite No. ¹	U.S. Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear
		Number -- Kind Code ² (if known)				
	79	US 6,135,646		10-24-2000	Kahn et al.	
	80	US 6,141,754		10-31-2000	Choy	
	81	US 6,157,719		12-05-2000	Wasilewski et al.	
	82	US 6,169,976 B1		01-02-2001	Colosso	
	83	US 6,185,683 B1		02-06-2001	Ginter et al.	
	84	US 6,189,037 B1		02-13-2001	Adams et al.	
	85	US 6,189,146 B1		02-13-2001	Misra et al.	
	86	US 6,209,092 B1		03-27-2001	Linnartz	
	87	US 6,216,112 B1		04-10-2001	Fuller et al.	
	88	US 6,219,652 B1		04-17-2001	Carter et al.	
	89	US 6,236,971 B1		05-22-2001	Stefik et al.	
	90	US 6,307,939 B1		10-23-2001	Vigarie	
	91	US 6,353,888 B1		03-05-2002	Kakehi et al.	
	92	US 6,397,333 B1		05-28-2002	Söhne et al.	
	93	US 6,401,211 B1		06-04-2002	Brezak Jr. et al.	
	94	US 6,405,369 B1		06-11-2002	Tsuria	
	95	US 6,424,717 B1		07-23-2002	Pinder et al.	
	96	US 6,424,947 B1		07-23-2002	Tsuria et al.	
	97	US 6,487,659 B1		11-26-2002	Kigo et al.	
	98	US 6,516,052 B2		02-04-2003	Voudouris	
	99	US 6,516,413 B1		02-04-2003	Aratani et al.	
	100	US 6,523,745 B1		02-25-2003	Tamori	
	101	US 6,796,555 B1		09-28-2004	Blahut	

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

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

¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at 222.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language translation is attached.

10886656.1

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

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	October 4, 2004
				First Named Inventor	Xin Wang et al.
				Art Unit	3621
				Examiner Name	West, Thomas C.
Sheet	4	of	9	Attorney Docket Number	111325/291300

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ²
		Country Code ³	Number ⁴				
	102	WO	83/04461 A1	12-22-1983	Western Electric Company, Inc.		
	103	WO	92/20022 A1	11-12-1992	Digital Equipment Corporation		
	104	WO	93/01550 A1	01-21-1993	Infologic Software, Inc.		
	105	WO	93/11480 A1	06-10-1993	Intergraph Corporation		
	106	WO	94/03003 A1	02-03-1994	Crest Industries, Inc.		
	107	WO	96/24092 A2	08-08-1996	Benson		
	108	WO	96/27155 A2	09-06-1996	Electronic Publishing Resources, Inc.		
	109	WO	97/25800 A1	07-17-1997	Mytec Technologies Inc.		
	110	WO	97/37492 A1	10-09-1997	Macrovision Corporation		
	111	WO	97/41661 A2	11-06-1997	Motorola Inc.		
	112	WO	97/43761 A2	11-20-1997	Intertrust Technologies Corp.		
	113	WO	98/09209 A1	03-05-1998	Intertrust Technologies Corp.		
	114	WO	98/10561 A1	03-12-1998	Telefonaktiebolaget LM Ericsson		
	115	WO	98/11690 A1	03-19-1998	Glover		
	116	WO	98/19431 A1	05-07-1998	Qualcomm Incorporated		
	117	WO	98/43426 A1	10-01-1998	Canal+Societe Anonyme		
	118	WO	98/45768 A1	10-15-1998	Northern Telecom Limited		
	119	WO	99/24928 A2	05-20-1999	Intertrust Technologies Corp.		
	120	WO	99/34553 A1	07-08-1999	V-One Corporation		
	121	WO	99/35782 A1	07-15-1999	Cryptography Research, Inc.		
	122	WO	99/48296 A1	09-23-1999	Intertrust Technologies Corporation		
	123	WO	99/60461 A1	11-25-1999	International Business Machines Corporation		
	124	WO	99/60750 A2	11-25-1999	Nokia Networks Oy		
	125	WO	00/04727 A2	01-27-2000	Koninklijke Philips Electronics N.V.		
	126	WO	00/05898 A2	02-03-2000	Optivision, Inc.		
	127	WO	00/59152 A2	10-05-2000	Microsoft Corporation		
	128	WO	00/72118 A1	11-30-2000	Compaq Computers Inc.		
	129	WO	00/73922 A2	12-07-2000	Entera, Inc.		
	130	WO	01/37209 A1	05-25-2001	Teralogic, Inc.		
	131	EP	0 067 556 B1	12-22-1982	Data General Corporation		
	132	EP	0 257 585 A2	03-02-1988	NEC Corporation		

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

10886656.1

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

Substitute for form 1449A/PTO				Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Application Number	10/956,121
				Filing Date	October 4, 2004
				First Named Inventor	Xin Wang et al.
				Art Unit	3621
				Examiner Name	West, Thomas C.
Sheet	5	of	9	Attorney Docket Number	111325/291300

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ²
		Country Code ³	Number ⁴				
	133	EP	0 332 304 A2	09-13-1989	Digital Equipment Corporation		
	134	EP	0 393 806 A2	10-24-1990	TRW Inc.		
	135	EP	0 450 841 A2	10-09-1991	GTE Laboratories Incorporated		
	136	EP	0 529 261 A2	03-03-1993	International Business Machines Corporation		
	137	EP	0 613 073 A1	08-31-1994	International Computers Limited		
	138	EP	0 678 836 A1	10-25-1995	Tandem Computers Incorporated		
	139	EP	0 679 977 A1	11-02-1995	International Business Machines Incorporated		
	140	EP	0 715 243 A1	06-05-1996	Xerox Corporation		
	141	EP	0 715 244 A1	06-05-1996	Xerox Corporation		
	142	EP	0 715 245 A1	06-05-1996	Xerox Corporation		
	143	EP	0 731 404 A1	09-11-1996	International Business Machines Corporation		
	144	EP	0 763 936 A2	03-19-1997	LG Electronics Inc.		
	145	EP	0 818 748 A2	01-14-1998	Murakoshi, Hiromasa		
	146	EP	0 840 194 A2	05-06-1998	Matsushita Electric Industrial Co., Ltd.		
	147	EP	0 892 521 A2	01-20-1999	Hewlett-Packard Company		
	148	GB	1483282	08-17-1977	Compagnie Internationale Pour L'Informatique C11-Honeywell-Bull		
	149	GB	2236604 A	04-10-1991	Sun Microsystems Inc.		
	150	GB	2309364 A	07-23-1997	Northern Telecom Limited		
	151	GB	2316503 A	02-25-1998	ICL Personal Systems Oy		
	152	BR	9810967 A (Abstract only)	10-30-2001	Scientific Atlanta Inc.		
	153	EP	0 934 765 A1	08-11-1999	Canal+Societe Anonyme		
	154	EP	0 946 022 A2	09-29-1999	Nippon Telegraph and Telephone Corporation		
	155	EP	0 964 572 A1	12-15-1999	Canal+Societe Anonyme		
	156	EP	1 103 922 A2 (Abstract only)	05-30-2001	CIT Alcatel		
	157	GB	2022969 A	12-19-1979	Data Recall Limited		
	158	GB	2354102 A	03-14-2001	Barron McCann Limited		
	159	JP	11031130 A2 (Abstract only)	02-02-1999	Fuji Xerox Co. Ltd.		

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

10886656.1

Substitute for form 1449A/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Xin Wang et al.
		Art Unit	3621
		Examiner Name	West, Thomas C.
Sheet	6	of	9
		Attorney Docket Number	111325/291300

FOREIGN PATENT DOCUMENTS							
Examiner Initials ¹	Cite No. ¹	Foreign Patent Document		Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ²
		Country Code ³	Number ⁴				
	160	JP	11032037 A2 (Abstract only)	02-02-1999	Fuji Xerox Co. Ltd.		
	161	JP	11205306 A2 (Abstract only)	07-30-1999	Fuji Xerox Co. Ltd.		
	162	JP	11215121 A2 (Abstract only)	08-06-1999	Fuji Xerox Co. Ltd.		
	163	JP	2000215165 A2 (Abstract only)	08-04-2000	Nippon Telegraph and Telephone		
	164	JP	2005218143 A2 (Abstract only)	08-11-2005	Scientific Atlanta Inc.		
	165	JP	2005253109 A2 (Abstract only)	09-15-2005	Scientific Atlanta Inc.		
	166	JP	2006180562 A2 (Abstract only)	07-06-2006	Intarsia Software LLC; Mitsubishi Corp.		
	167	JP	5168039 A2 (Abstract only)	07-02-1993	Sony Corp.		
	168	WO	96/13814 A1	05-09-1996	Vazvan		
	169	WO	00/46994 A1	08-10-2000	Canal+Societe Anonyme		
	170	WO	00/62260 A1 (Abstract only)	10-19-2000	Swisscom Mobile AG		
	171	WO	01/03044 A1	01-11-2001	Transcast International, Inc.		
	172	WO	04/103843 (Abstract only)	12/02/2004	S2F Flexico		
	173	WO	04/34223 A2	04-22-2004	Legal IGaming, Inc.		

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

10886656.1

Substitute for form 1449A/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Xin Wang et al.
		Art Unit	3621
		Examiner Name	West, Thomas C.
Sheet	7	of	9
		Attorney Docket Number	111325/291300

OTHER PRIOR ART – NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	12
	174	BLAZE et al, "Divertible Protocols and Atomic Proxy Cryptography" 1998 Advances in Cryptography – Euro Crypt International Conference on the Theory and Application of Crypto Techniques, Springer Verlag, DE	
	175	BLAZE et al, "Atomic Proxy Cryptography" DRAFT (Online) (November 2, 1997) XP002239619 Retrieved from the Internet	
	176	NO AUTHOR, "Capability- and Object-Based Systems Concepts," Capability-Based Computer Systems, pp. 1-19 (no date)	
	177	COX, "Superdistribution" Wired Magazine (September 1994) XP002233405 URL: http://www.wired.com/wired/archive/2.09/superdis_pr.html&gt	
	178	DUNLOP et al, Telecommunications Engineering, pp. 346-352 (1984)	
	179	ELGAMAL, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Transactions on Information Theory IT-31(4):469-472 (July 1985)	
	180	GHEORGHIU et al, "Authorization for Metacomputing Applications" (no date)	
	181	IANNELLA, ed., Open Digital Rights Language (ODRL), pp. 1-31 (November 21, 2000)	
	182	KAHLE, wais.concepts.txt, Wide Area Information Server Concepts, Thinking Machines Version 4, Draft, pp. 1-18 (November 3, 1989)	
	183	KAHN, "Deposit, Registration and Recordation in an Electronic Copyright Management System," Technical Report, Corporation for National Research Initiatives, Reston, Virginia (August 1992) URL: http://www.cni.org/docs/ima_ip-workshop/kahn.html	
	184	KAHN et al, "The Digital Library Project, Volume 1: The World of Knowbots (DRAFT), An Open Architecture for a Digital Library System and a Plan for its Development," Corporation for National Research Initiatives, pp. 1-48 (March 1988)	
	185	KOHL et al, Network Working Group Request for Comments: 1510, pp. 1-112 (September 1993)	
	186	LEE et al, CDMA Systems Engineering Handbook (1998) [excerpts but not all pages numbered]	
	187	MAMBO et al, "Protection of Data and Delegated Keys in Digital Distribution," Information Security and Privacy. Second Australian Conference, ACISP '97 Proceedings, pp. 271-282 (Sydney, NSW, Australia, 7-9 July 1997, 1997 Berlin, Germany, Springer-Verlag, Germany), XP008016393 ISBN: 3-540-63232-8	
	188	MAMBO et al, "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts," IEICE Trans. Fundamentals VOL. E80-A, NO. 1:54-63 (January 1997) XP00742245 ISSN: 0916-8508	
	189	Microsoft Word, Users Guide, Version 6.0, pp. 487-89, 549-55, 560-64, 572-75, 599-613, 616-31 (1993)	
	190	OJANPERÄ and PRASAD, eds., Wideband CDMA for Third Generation Mobile Communications (1998) [excerpts but not all pages numbered]	
	191	PERRITT, "Knowbots, Permissions Headers and Contract Law," Paper for the Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, pp. 1-22 (April 2-3, 1993 with revisions of April 30, 1993)	

Examiner Signature		Date Considered	
-----------------------	--	--------------------	--

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

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

Substitute for form 1449A/PTO		Complete if Known	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>		Application Number	10/956,121
		Filing Date	October 4, 2004
		First Named Inventor	Xin Wang et al.
		Art Unit	3621
		Examiner Name	West, Thomas C.
		Attorney Docket Number	111325/291300
Sheet	8	of	9

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	192	RAGGETT, (Hewlett Packard), "HTML+(Hypertext markup language)," pp. 1-31 (12 July 1993) URL: http://citeseer.ist.psu.edu/correc/340709	
	193	SAMUELSON et al, "Intellectual Property Rights for Digital Library and Hypertext Publishing Systems: An Analysis of Xanadu," Hypertext '91 Proceedings, pp. 39-50 (December 1991)	
	194	NO AUTHOR, "Softlock Services Introduces... Softlock Services" Press Release (January 28, 1994)	
	195	NO AUTHOR, "Appendix III - Compatibility with HTML," NO TITLE, pp. 30-31 (no date)	
	196	NO EDITOR, NO TITLE, Dictionary pages, pp. 469-72, 593-94 (no date)	
	197	BENOIT, Digital Television MPEG-1, MPEG-2 and Principles of the DVB System, pp. 75-80, 116-121 (no date)	
	198	BENOIT, Digital Television MPEG-1, MPEG-2 and Principles of the DVB System, 2 nd edition, pp. 74-80 (no date)	
	199	AH Digital Audio and Video Series, "DTV Receivers and Measurements," Understanding Digital Terrestrial Broadcasting, pp. 159-64 (no date)	
	200	O'DRISCOLL, The Essential Guide to Digital Set-Top Boxes and Interactive TV, pp. 6-24 (no date)	
	201	IUS MENTIS, "The ElGamal Public Key System," pp. 1-2 (October 1, 2005) online at http://www.iusmentis.com/technology/encryption/elgamal/	
	202	SCHNEIER, "Crypto Bibliography," Index of Crypto Papers Available Online, pp. 1-2 (online) (no date)	
	203	NO AUTHOR, NO TITLE, pp. 344-55 (no date)	
	204	NO AUTHOR, "Part Four Networks," NO TITLE, pp. 639-714 (no date)	
	205	Microsoft Word User's Guide, pp. 773-74, 315-16, 487-89, 561-64, 744, 624-33 (1993)	
	206	NO AUTHOR, "What is the ElGamal Cryptosystem," p. 1 (November 27, 2006) online at http://www.x5.net/faqs/crypto/q29.html	
	207	JOHNSON et al., "A Secure Distributed Capability Based System," ACM, pp. 392-402 (1985)	
	208	Wikipedia, "El Gamal Encryption," pp.1-3 (last modified November 2, 2006) online at http://en.wikipedia.org/wiki/ElGamal_encryption	
	209	BLAZE, "Atomic Proxy Cryptography," p. 1 Abstract (October 20, 1998)	
	210	BLAZE, "Matt Blaze's Technical Papers," pp. 1-6 (last updated August 6, 2006)]	
	211	Online Search Results for "inverted file", "inverted index" from www.techweb.com , www.cryer.co.uk , computing-dictionary.thefreedictionary.com , www.nist.gov , en.wikipedia.org , www.cni.org , www.tiscali.co.uk (July 15-16, 2006)	
	212	Corporation for National Research Initiatives, "Digital Object Architecture Project", http://www.nnri.reston.va.us/doa.html (updated 28 Nov 2006)	

Examiner Signature	Date Considered
--------------------	-----------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

10886656.1

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

Substitute for form 1449A/PTO INFORMATION DISCLOSURE STATEMENT BY APPLICANT <i>(use as many sheets as necessary)</i>				Complete if Known		
				Application Number	10/956,121	
Sheet 9 of 9				Filing Date	October 4, 2004	
				First Named Inventor	Xin Wang et al.	
				Art Unit	3621	
				Examiner Name	West, Thomas C.	
				Attorney Docket Number	111325/291300	

OTHER PRIOR ART - NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²
	213	STEFIK, Summary and Analysis of A13 (Kahn, Robert E and Vinton G Cerf, "The Digital Library Project, Volume 1: The World of Knowbots (DRAFT), An Open Architecture for a Digital Library System and a Plan for its Development," Corporation for National Research Initiatives (March 1988)), pp. 1-25 (May 30, 2007)	

Examiner Signature	Date Considered
-----------------------	--------------------

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

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

10886656.1



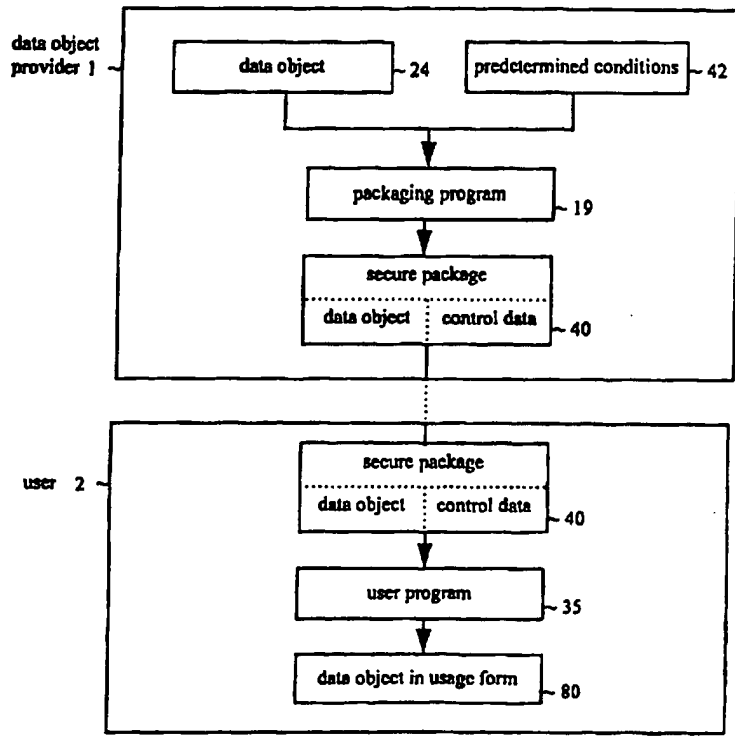
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁶ : G06F 1/00, 12/14</p>	<p>A2</p>	<p>(11) International Publication Number: WO 96/24092 (43) International Publication Date: 8 August 1996 (08.08.96)</p>
<p>(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).</p>	<p>(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).</p> <p>Published <i>Without international search report and to be republished upon receipt of that report.</i></p>	

(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
BG	Burkina Faso	IE	Ireland	NZ	New Zealand
BH	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	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			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
5 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
10 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
15 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
20 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
25 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
30 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
35 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
5 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
10 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
15 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
20 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
25 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
30 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
35 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

5 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
10 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.

15 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
20 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
25 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
30 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
35 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.

5 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
10 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
15 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
20 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
25 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
30 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
35 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.

5 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.

15 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.

25 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.

30 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
decrypting 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 con-
5 trol 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 particu-
10 larly, 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
15 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
20 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
25 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.

30 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.

35 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.

5 Fig. 7 is a data flow diagram of loading an object to the data object provider's data processor.

10 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.

15 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.

20 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.

25 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.

30 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

35 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
5 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
10 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
15 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
20 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
25 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
30 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
35 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.

5 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.

10 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
15 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:

20 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
25 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,
30 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
35 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.

5 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.

10 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.

15 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
20 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
25 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
30 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
35 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.

5 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.

10 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:

15 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
20 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 en-
25 ryption 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.

30 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
35 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
5 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
10 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
15 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.
20

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
25 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.

30 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.

35 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
5 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
10 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
15 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
20 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
25 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
30 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
35 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.

5 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
10 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
15 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,
20 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
25 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.
30 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.

35 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 constitutes 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.

20 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
5 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.

10 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.

15 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
20 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
25 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
30 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
35 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
5 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
10 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
15 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,
20 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
25 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
30 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-
35 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.

10 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.

25 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
5 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.
10 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.

15 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
20 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.

25 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
30 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.

35 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 transferred 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.

5 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.

10 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.

15 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
20 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
25 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
30 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
35 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
5 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 interested in using it. He then loads the data package 40 containing the image to
10 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
15 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
20 "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
25 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
30 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
35 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
5 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
10 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
15 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 pro-
20 gram 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
25 module, which forwards the approval to the usage control module, which enables one usage. If the user requests further usages of the data object, the authorization process is repeated. This procedure results in a permanent data object security.

30 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
35 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
5 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.

10 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 indicated in the first usage element the
15 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
20 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:

25 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.
30 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
35 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

5 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
10 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
15 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
20 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
25 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.

30 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
35 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:

5 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
10 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 imple-
15 ment 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. How-
20 ever, only those data objects with control data permitting further distribution or reselling are enabled in that way.

Rebrokering

25 An author of a data object may want to allow his original broker to distribute his data object to other brokers whom will also distribute his image. He then includes a control element which enables rebrokering in the control data before distributing the data object with its associated control data to the original broker. Upon
30 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
35 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.

5 Suppose some company wants 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
10 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
15 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
20 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 virtual money that is issued and
25 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
30 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.
35 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
5 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
10 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.

15 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.

20 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.

25

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:
- 5 - 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;
 - 10 - 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
 - 20 - 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.
- 30 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.
- 35 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:

- 5 - 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;
- 10 - 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
- 15 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

20 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 authori-

25 zation 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

30 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

35 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
5 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.

10 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
15 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 compo-
20 site 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.

25 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
30 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
35 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;

10 - 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;

15 - 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;

20 - 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.

30 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:

35 - 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;

- 5 - 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
- 10 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
- 15 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.

- 20 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
- 25 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.
- 30

20. A method as set forth in any one of claims 16-19, wherein the step of checking whether the requested
- 35 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
5 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
10 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
15 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
20 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
25 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,
35 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.

5 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.

10 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;

20 - 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

25 - 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.

30 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:

35 - 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
5 elements being encrypted;

- decrypting the usage control elements of the user sets of control data;

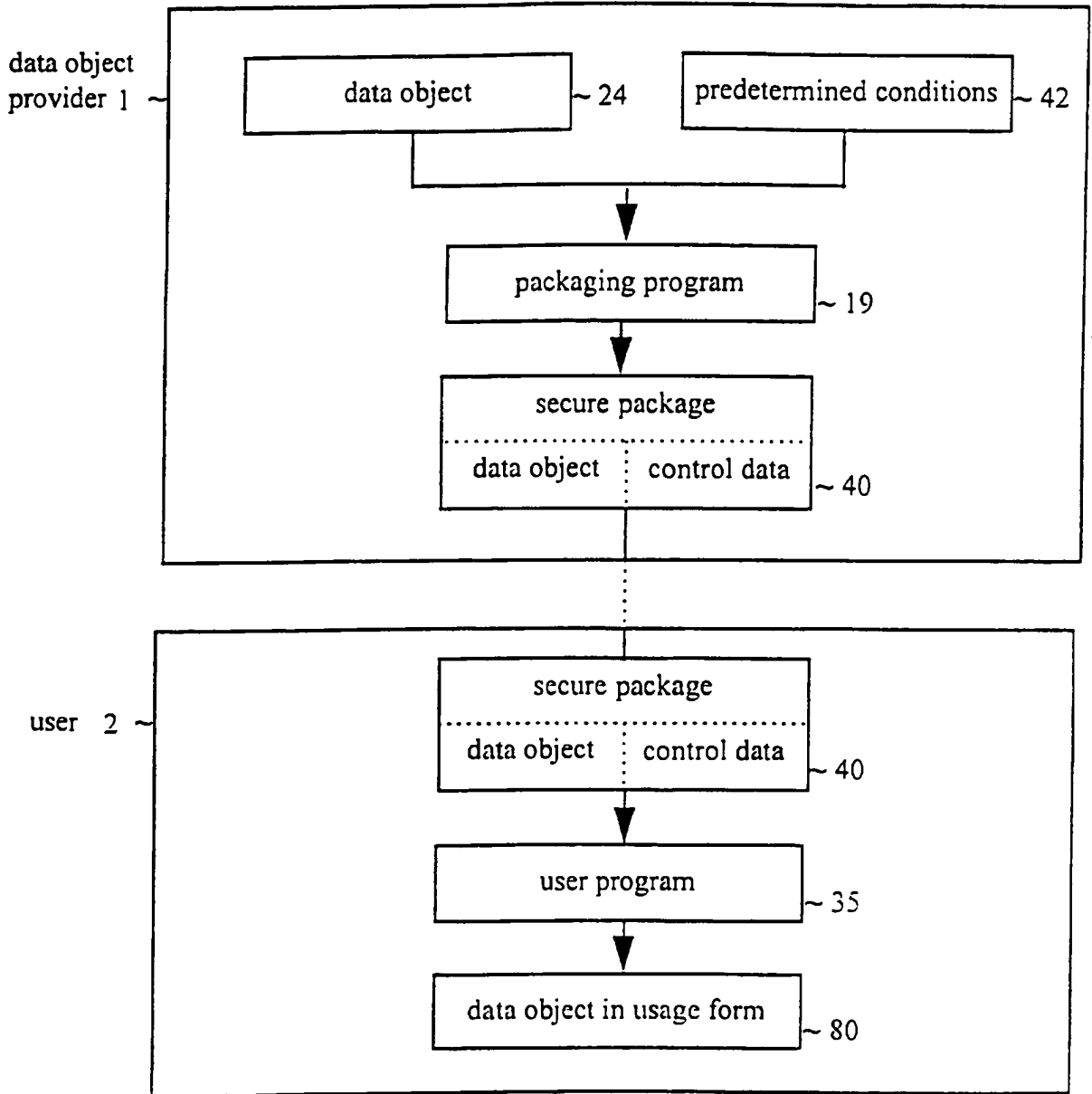
- examining the usage control elements of said at least two data packages to find a match;

10 - 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
15 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.

20

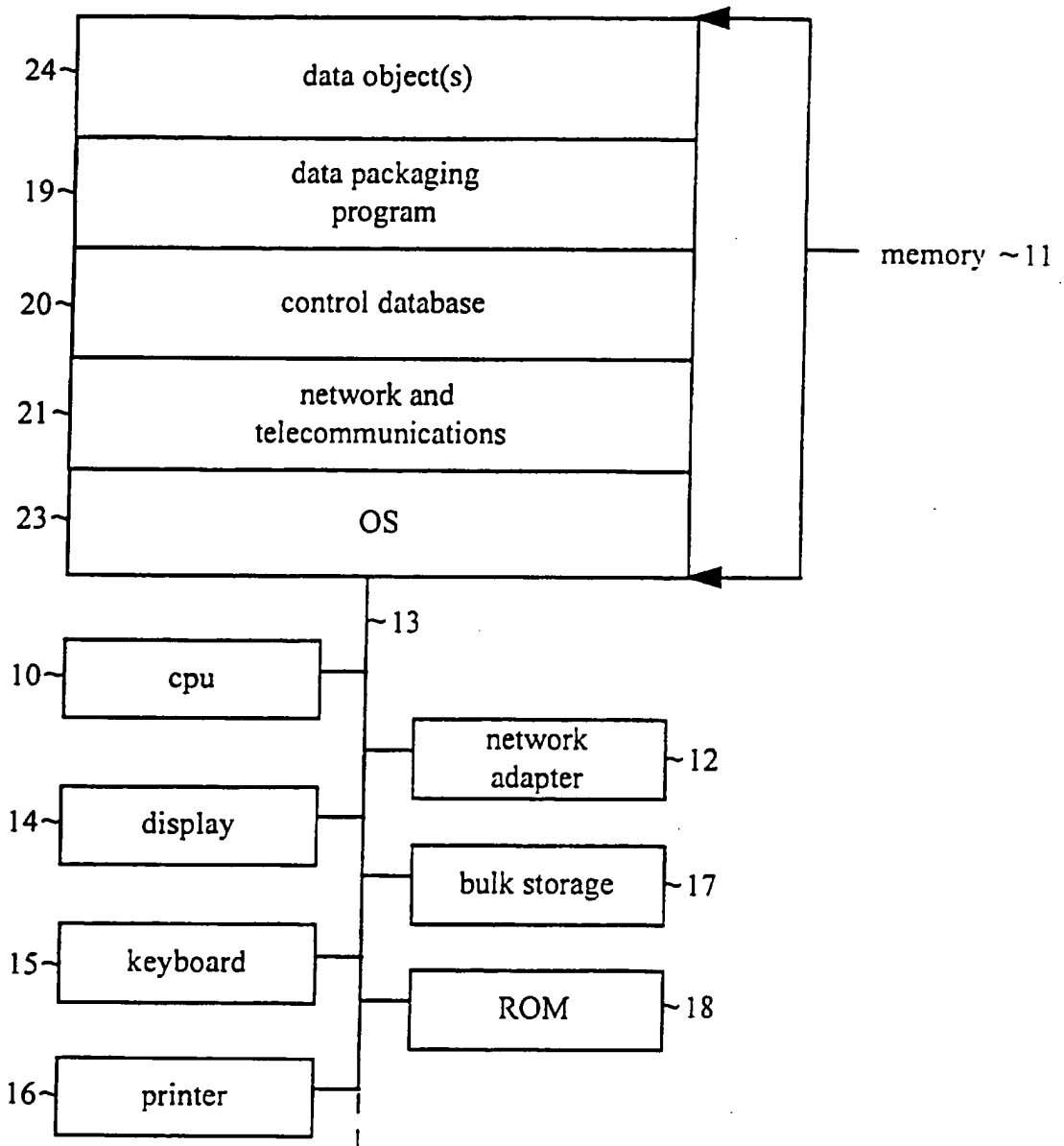
Fig 1



SUBSTITUTE SHEET

2/15

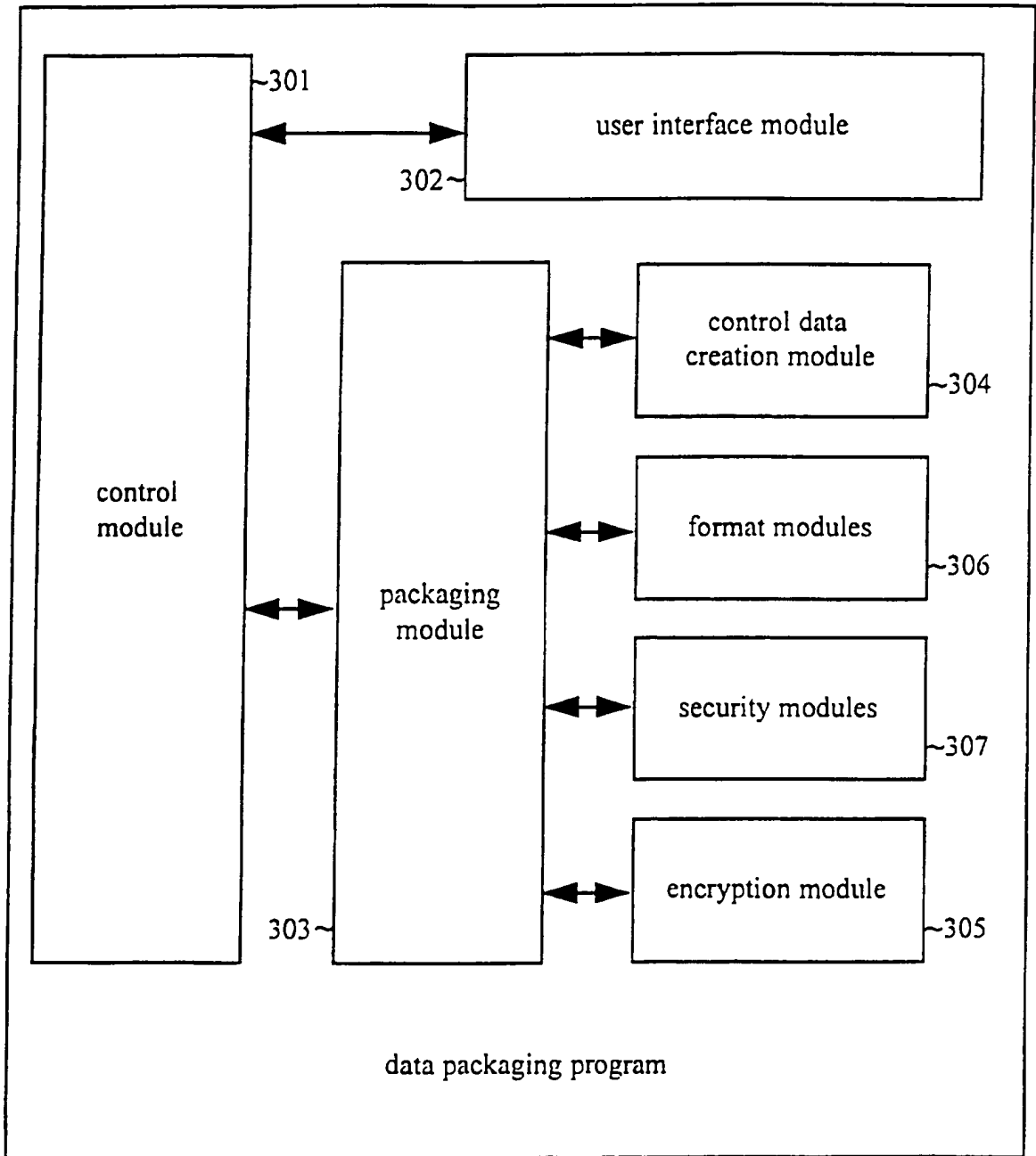
Fig 2



SUBSTITUTE SHEET

3/15

Fig 3

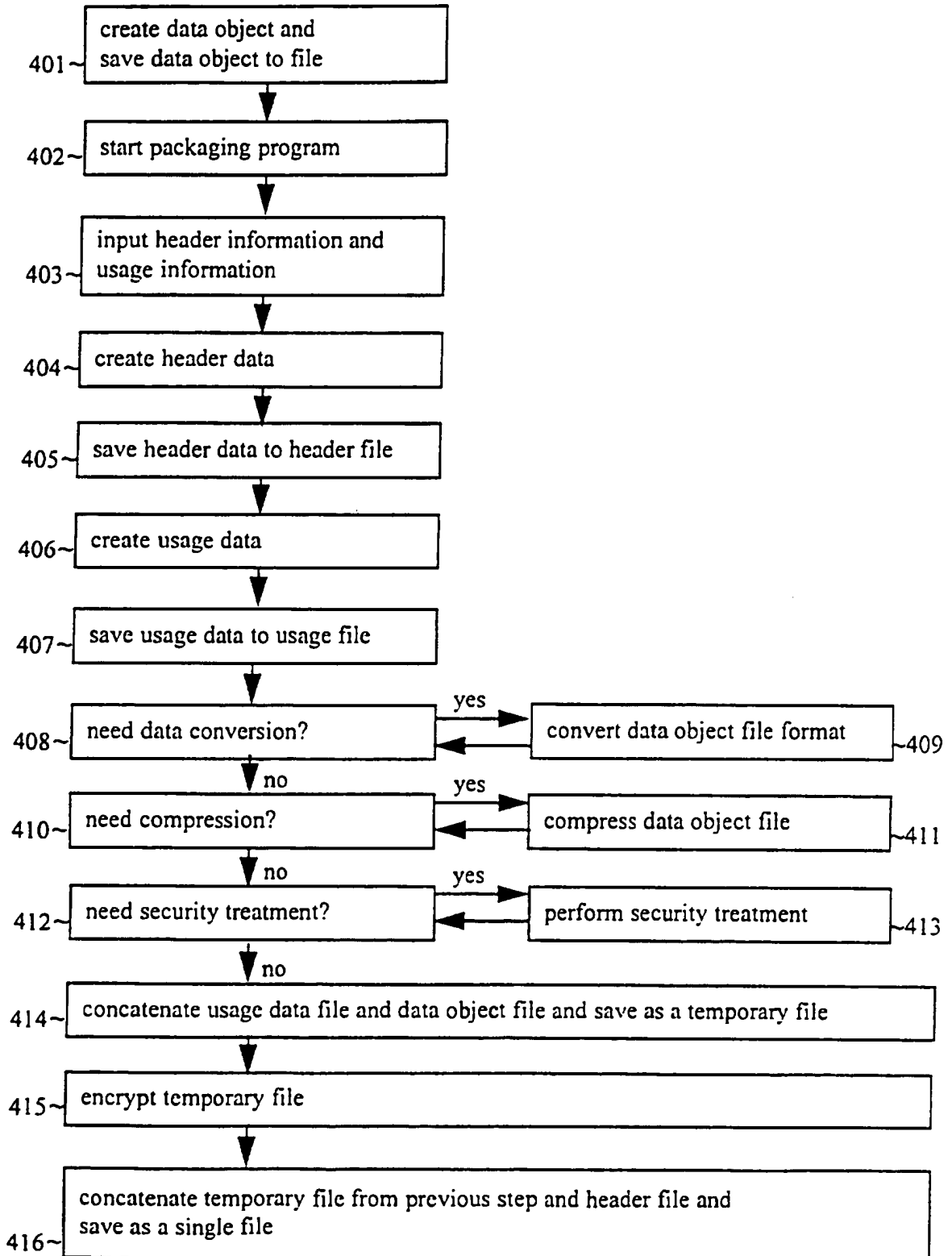


19

SUBSTITUTE SHEET

4/15

Fig 4



5/15

Fig 5

file identifier	123456789
title	image
format code	a
security code	b

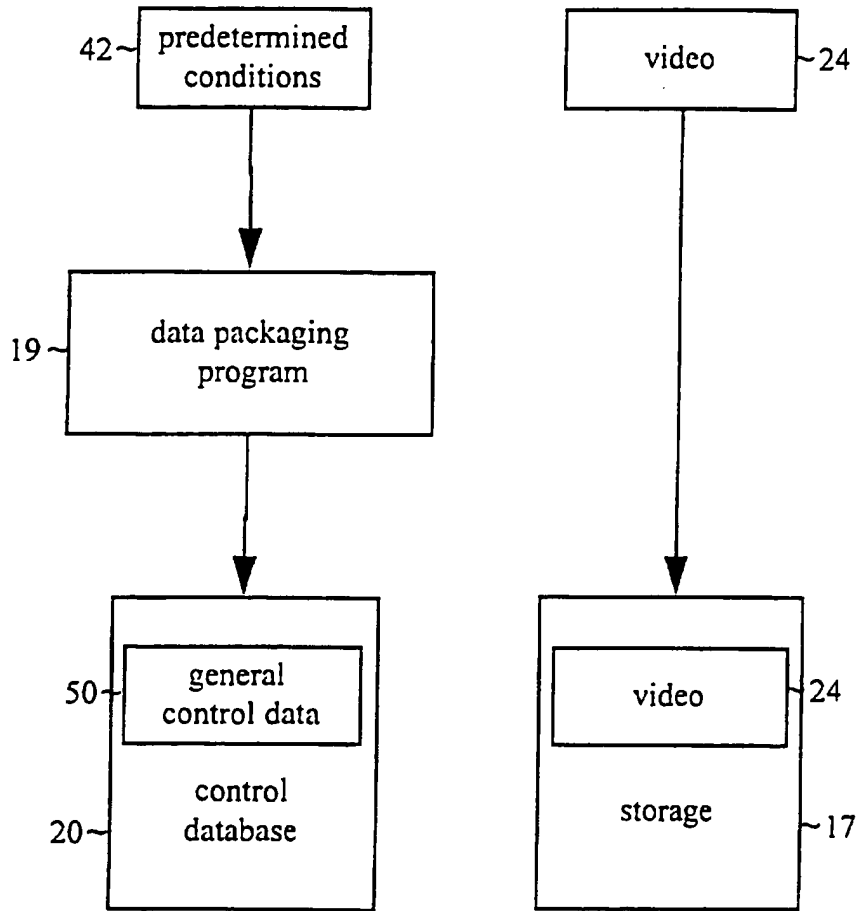
Fig 6

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

SUBSTITUTE SHEET

6/15

Fig 7



7/15
Fig 8a

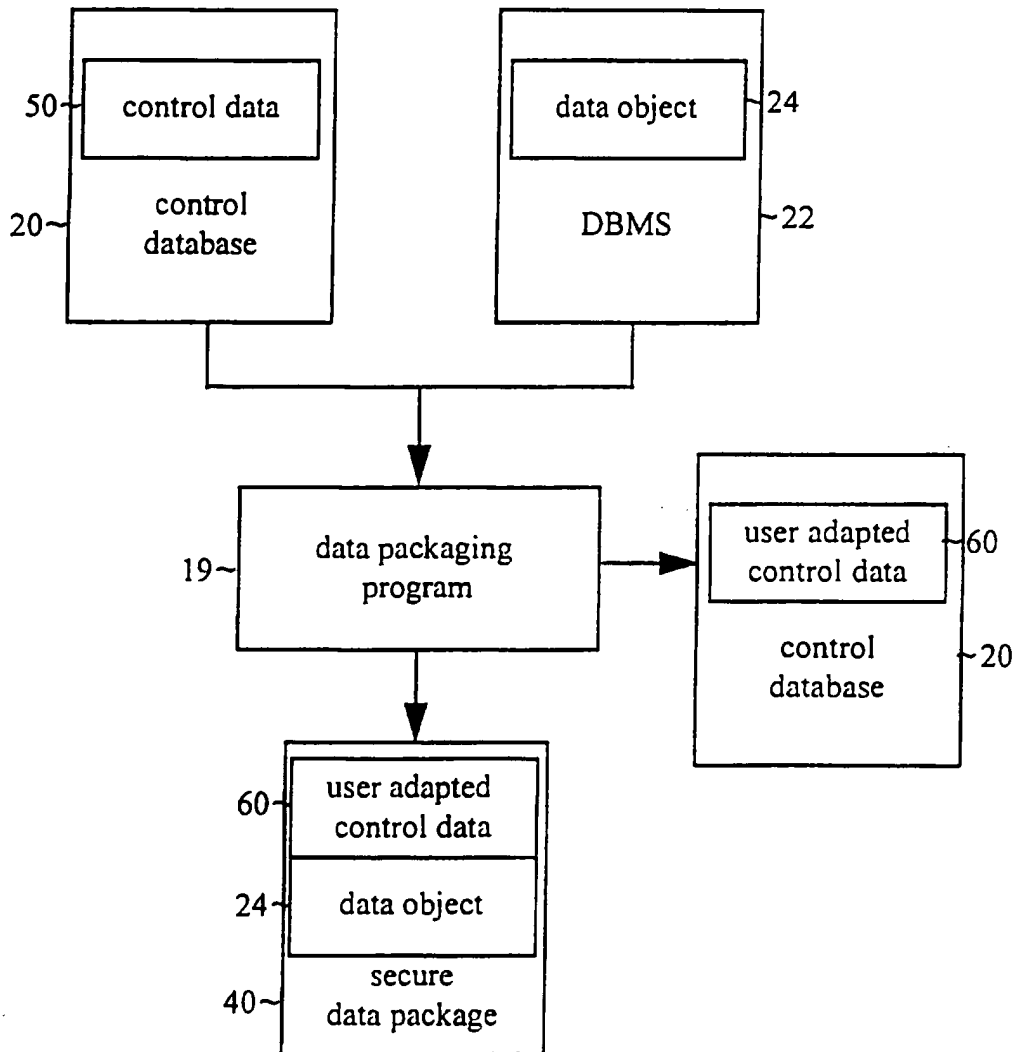
header	object identifier	123456789
	format code	0010
	security code	0010
	number of usage elements	2
	size of usage data	17
	size of data object	273
	1st usage element id	001
	1st usage element size	6
	1st usage element data	1
	2nd usage element id	002
	2nd usage element size	3
	2nd usage element data	

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

8/15

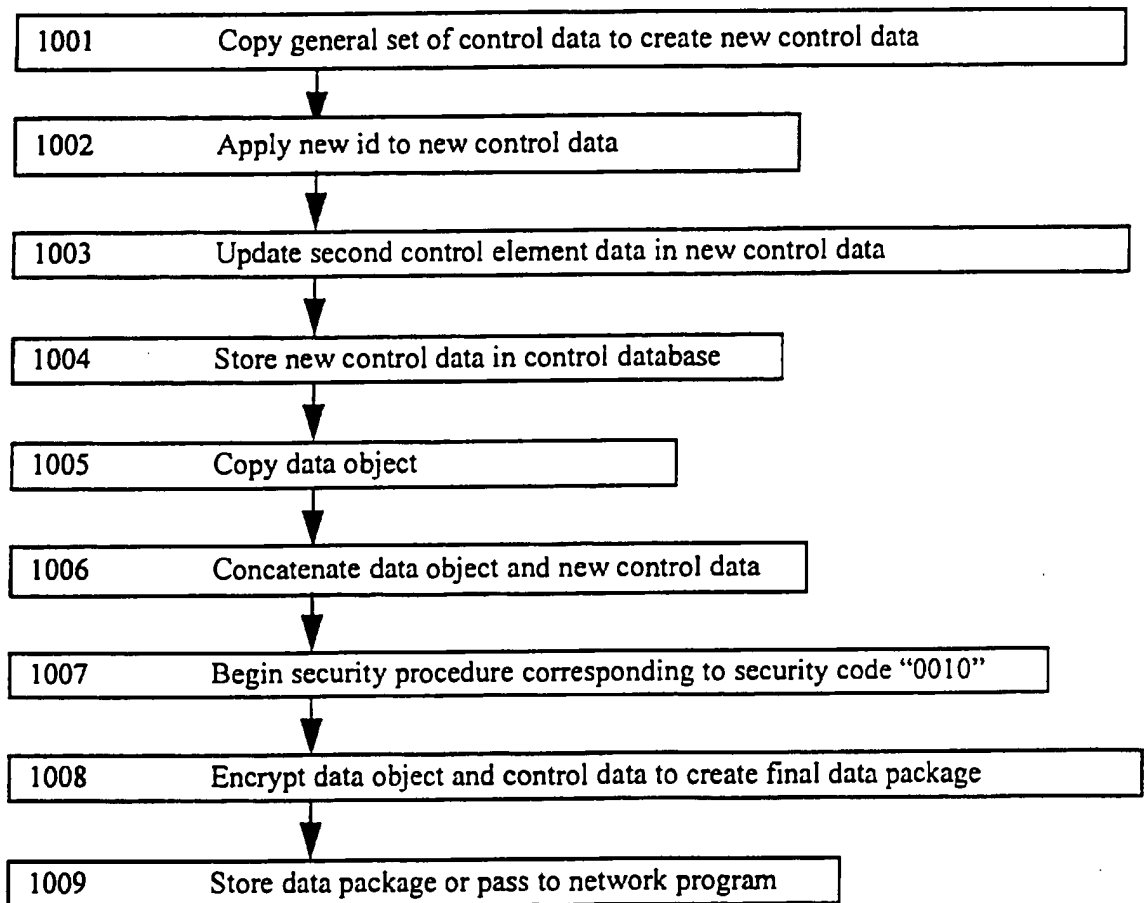
Fig 9



SUBSTITUTE SHEET

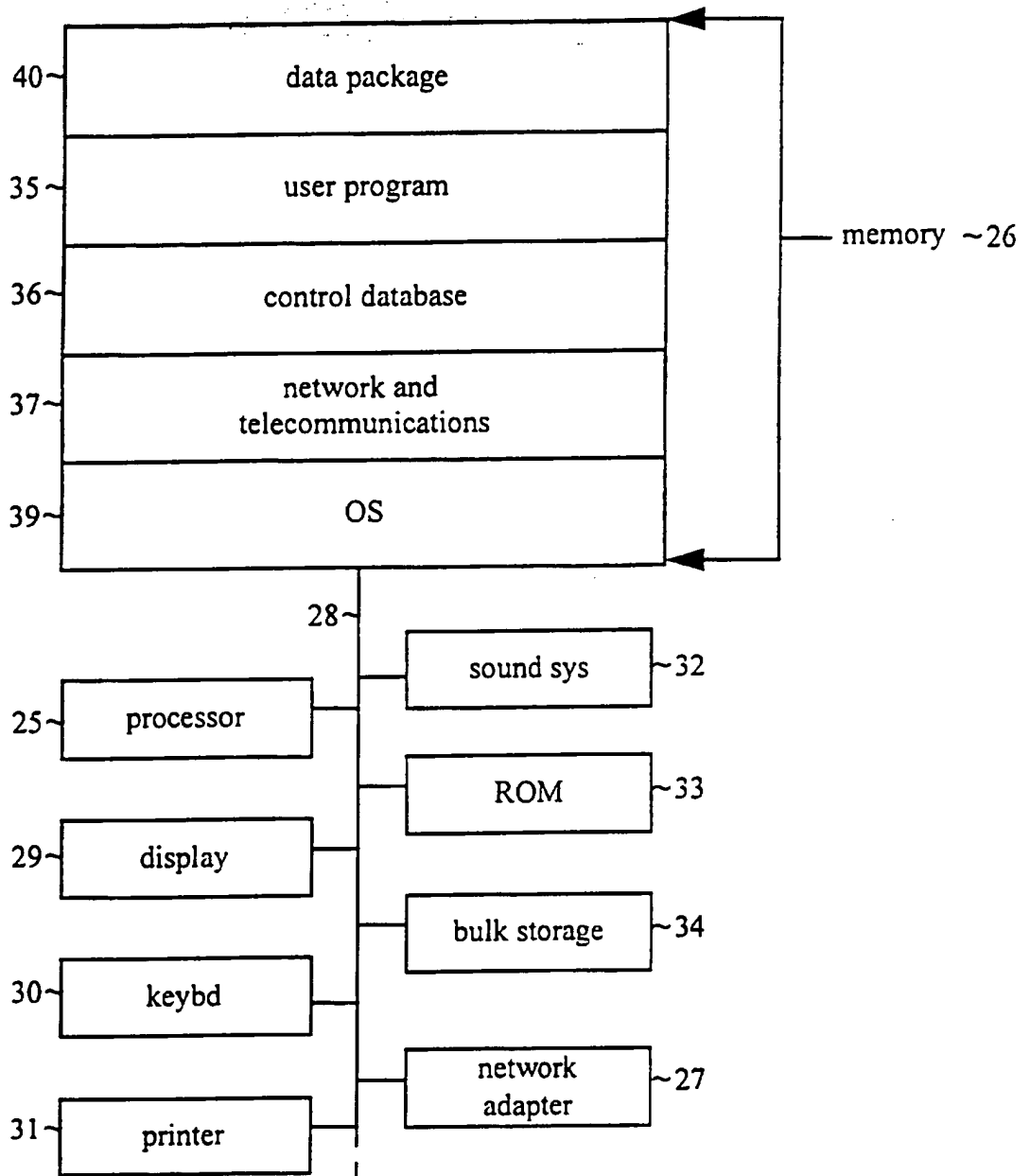
9/15

Fig 10

**SUBSTITUTE SHEET**

11/15

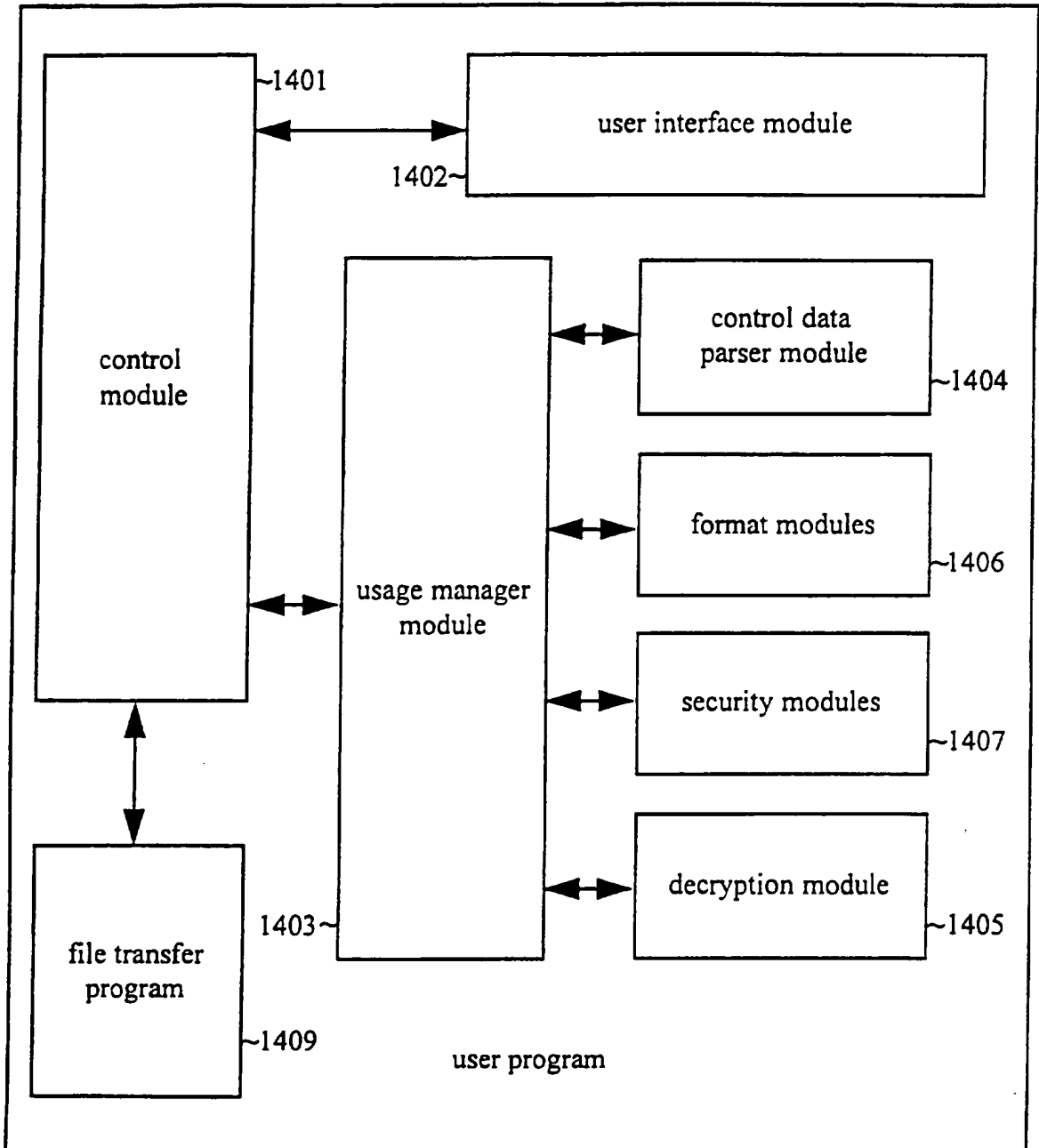
Fig 13



SUBSTITUTE SHEET

12/15

Fig 14

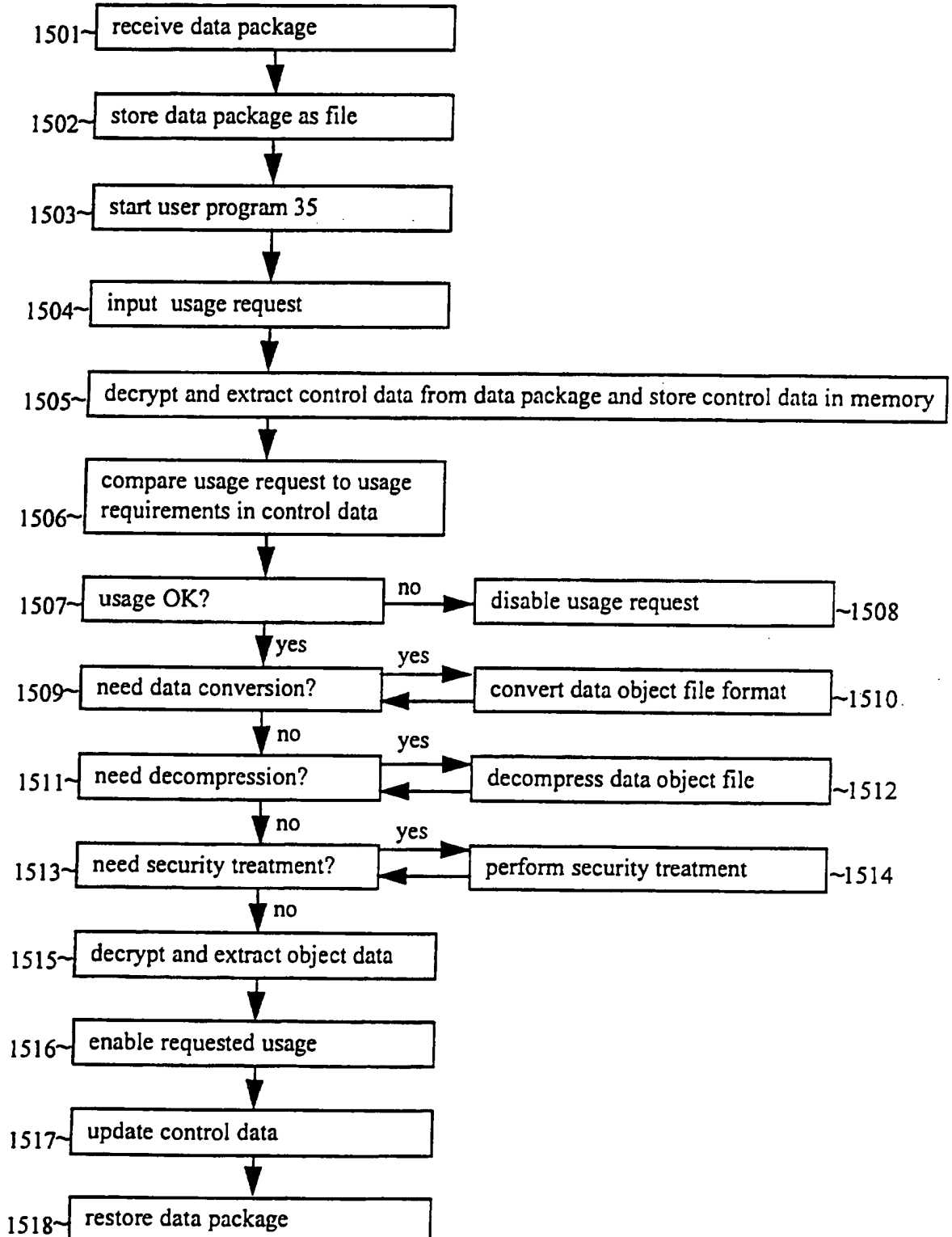


35

SUBSTITUTE SHEET

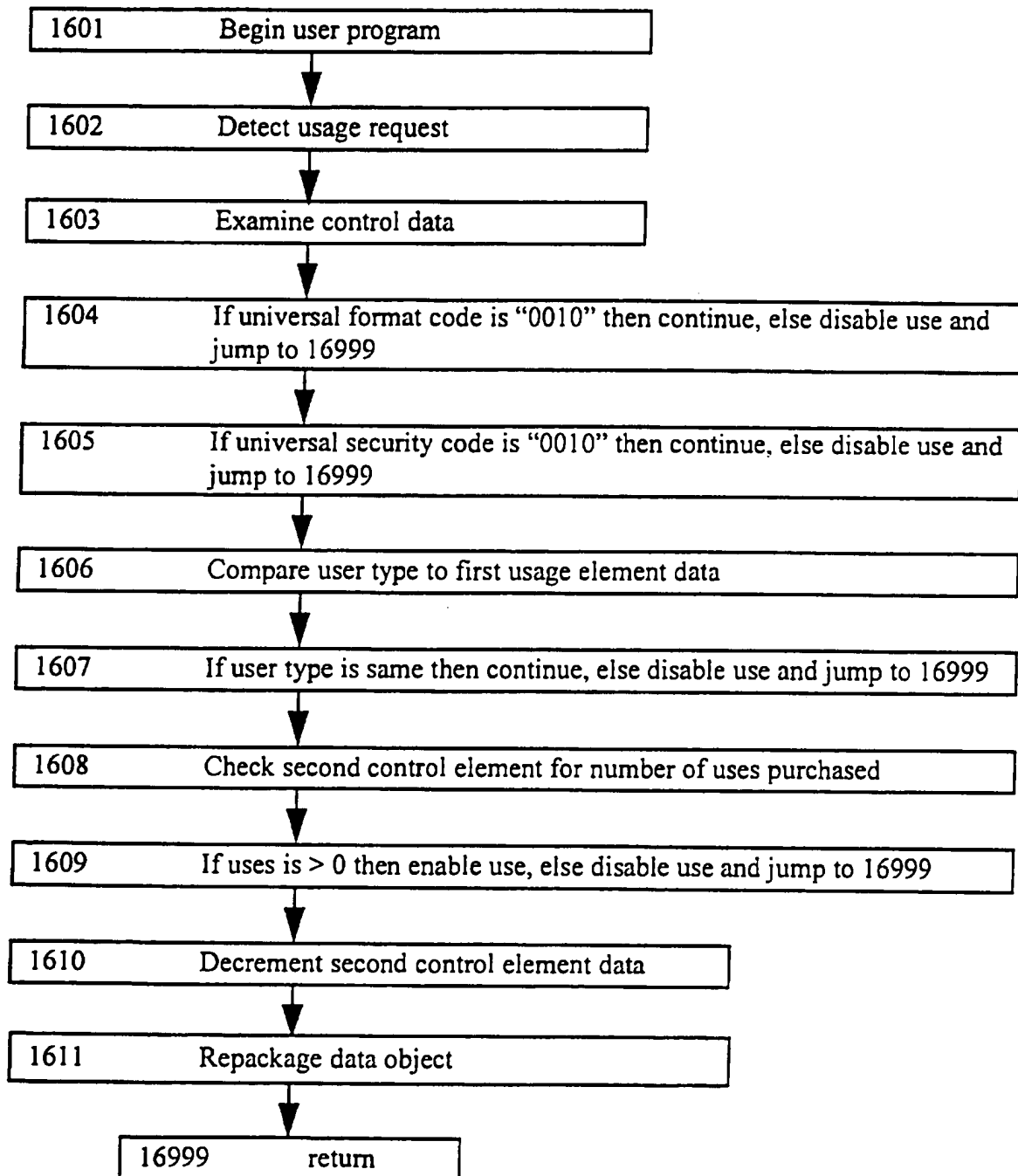
13/15

Fig 15



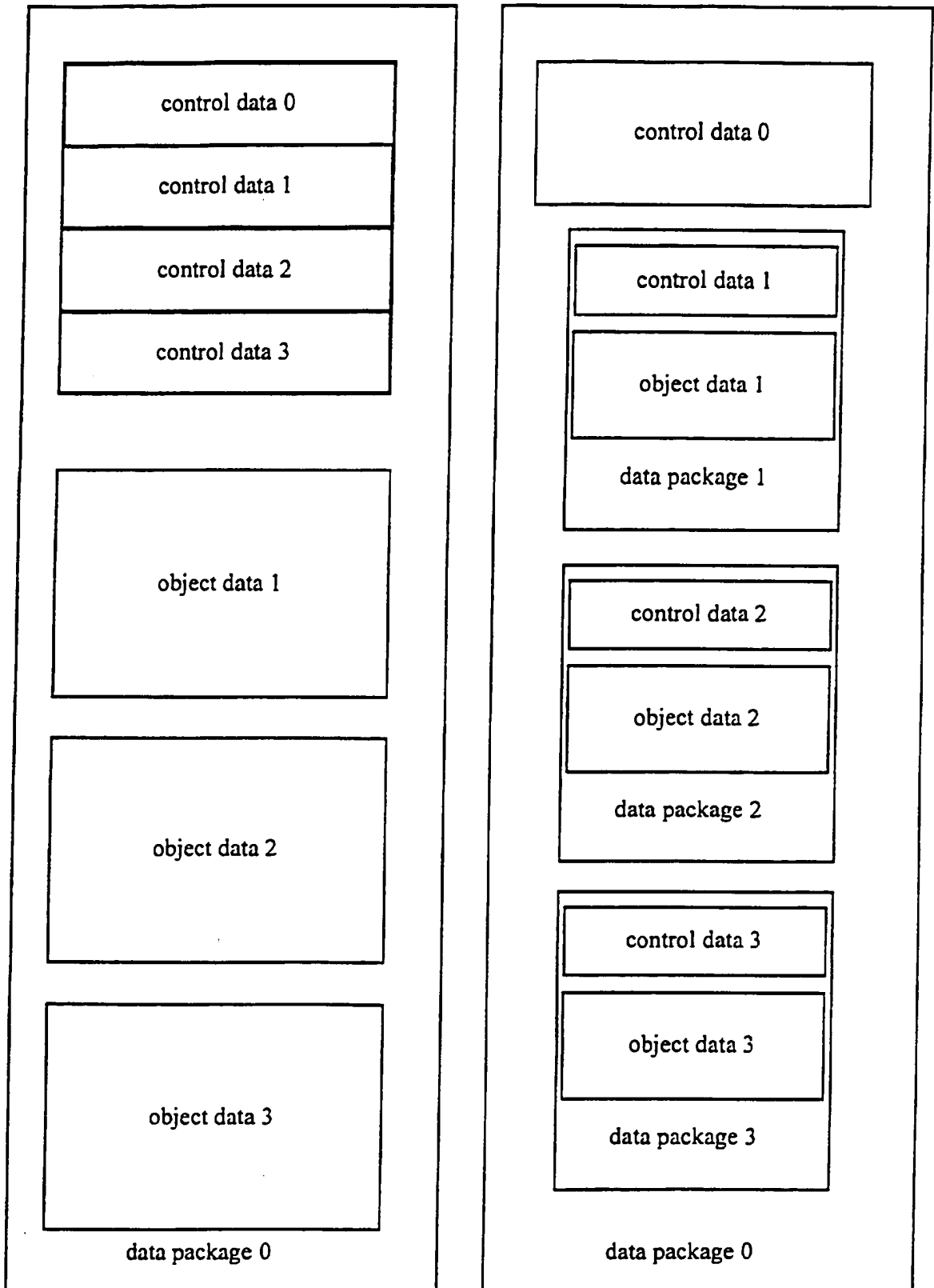
14/15

Fig 16

**SUBSTITUTE SHEET**

15/15

Fig 17

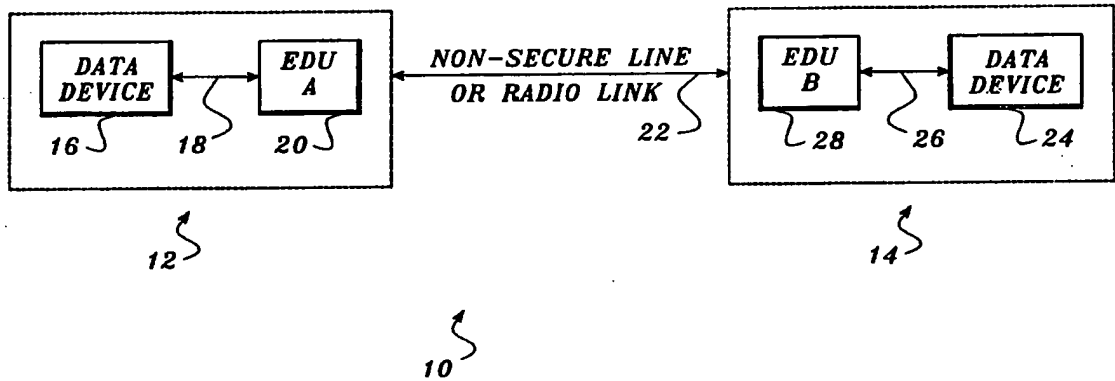




INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁵ : H04L 9/00</p>	<p>A1</p>	<p>(11) International Publication Number: WO 94/03003 (43) International Publication Date: 3 February 1994 (03.02.94)</p>
<p>(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).</p>		<p>(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 <i>With international search report.</i></p>

(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	KR	Republic of Korea	SD	Sudan
CG	Congo	KZ	Kazakhstan	SE	Sweden
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovak Republic
CM	Cameroon	LU	Luxembourg	SN	Senegal
CN	China	LV	Latvia	TD	Chad
CS	Czechoslovakia	MC	Monaco	TC	Togo
CZ	Czech Republic	MG	Madagascar	UA	Ukraine
DE	Germany	ML	Mali	US	United States of America
DK	Denmark	MN	Mongolia	UZ	Uzbekistan
ES	Spain			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
5 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
10 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
15 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
20 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.

10 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

15 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.

30 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.

5 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
10 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
15 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
20 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
25 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"
30 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
35 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
5 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
10 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
15 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
20 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
25 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
30 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
35 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
5 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
10 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
15 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
20 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
25 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
35 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.

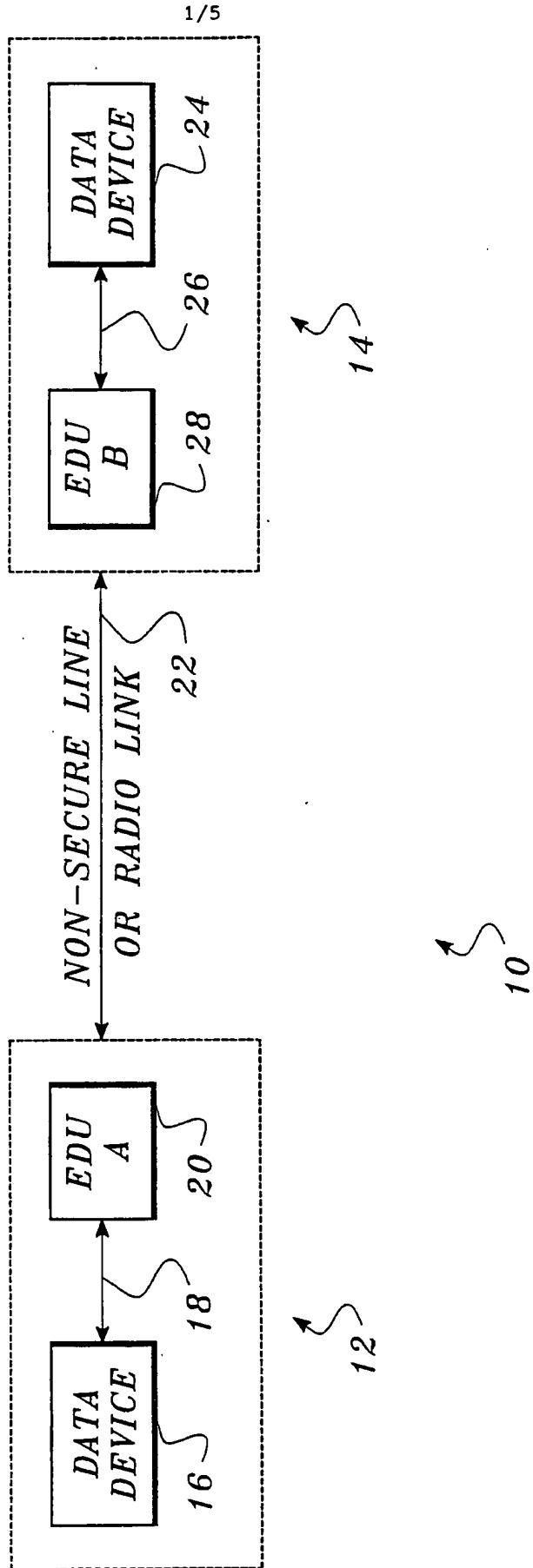


FIG. 1.

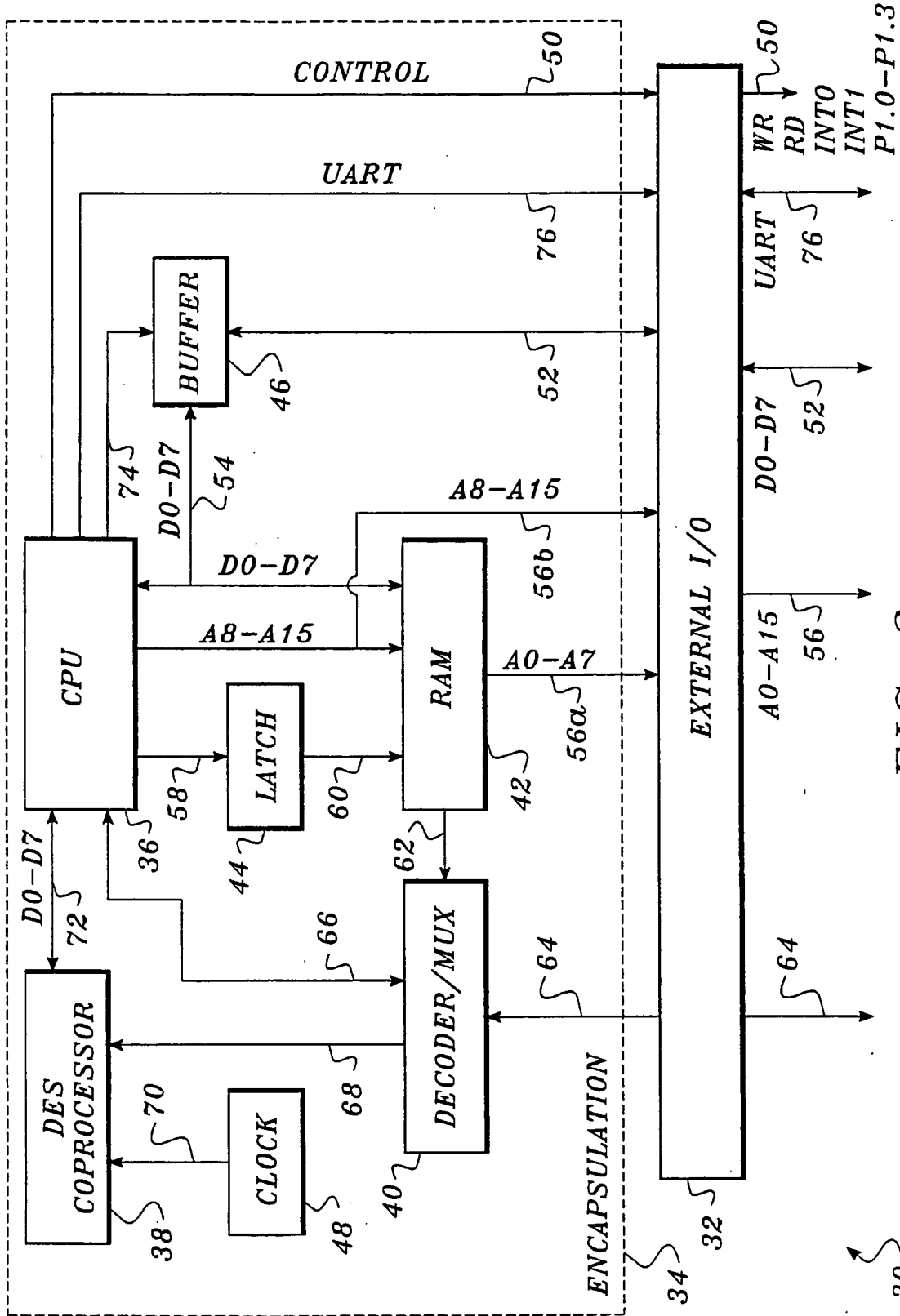
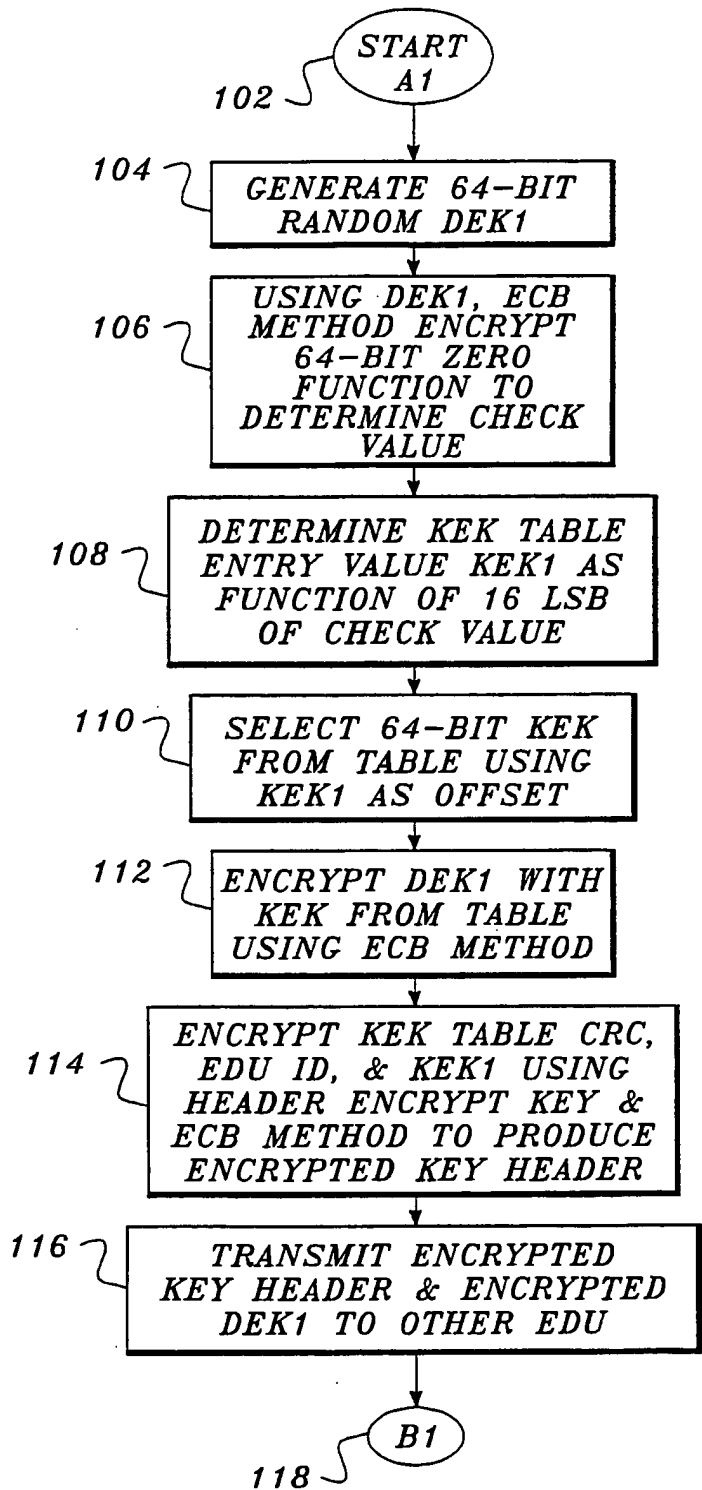


FIG. 2.

3/5



100 ↗

FIG. 3.

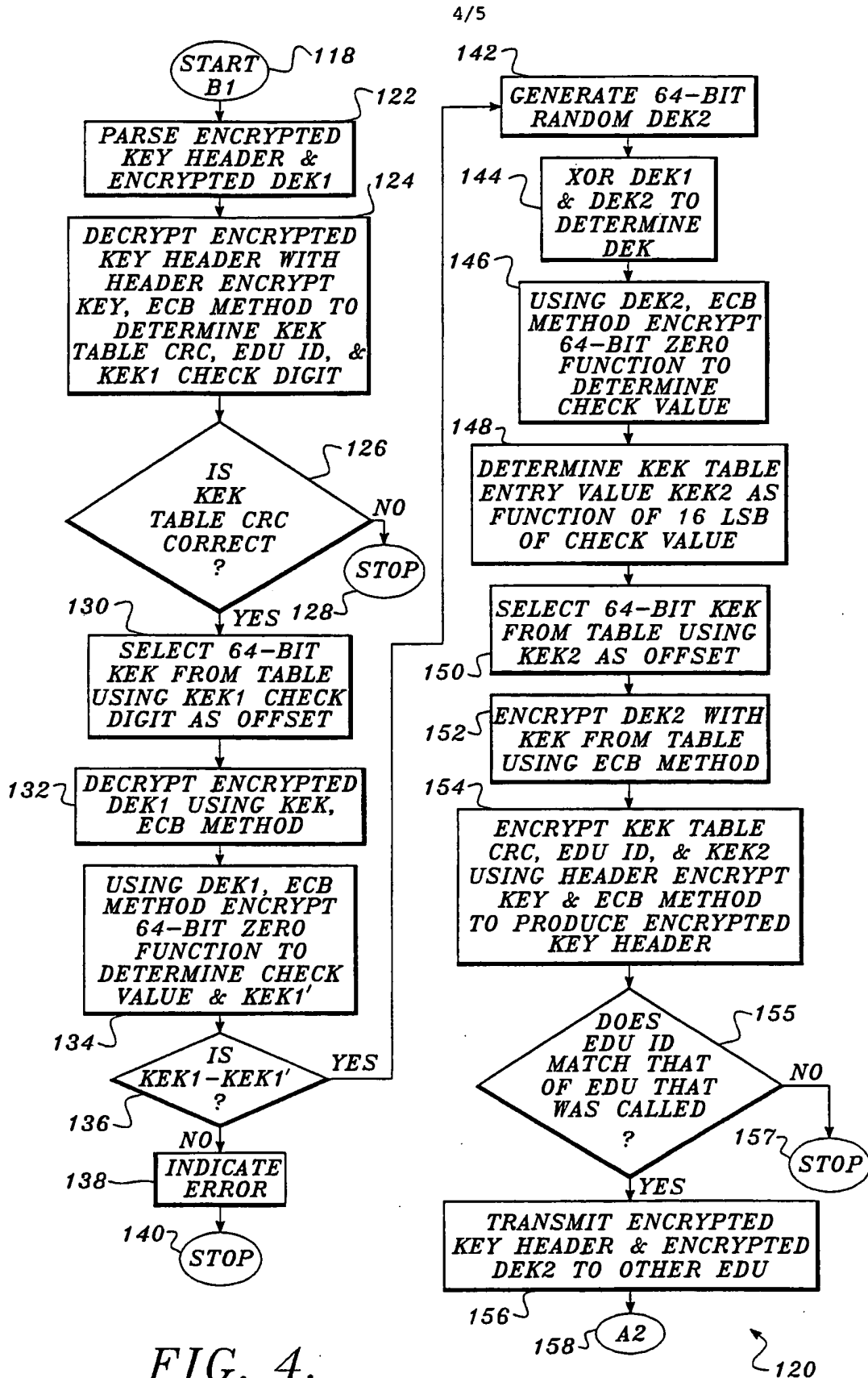


FIG. 4.

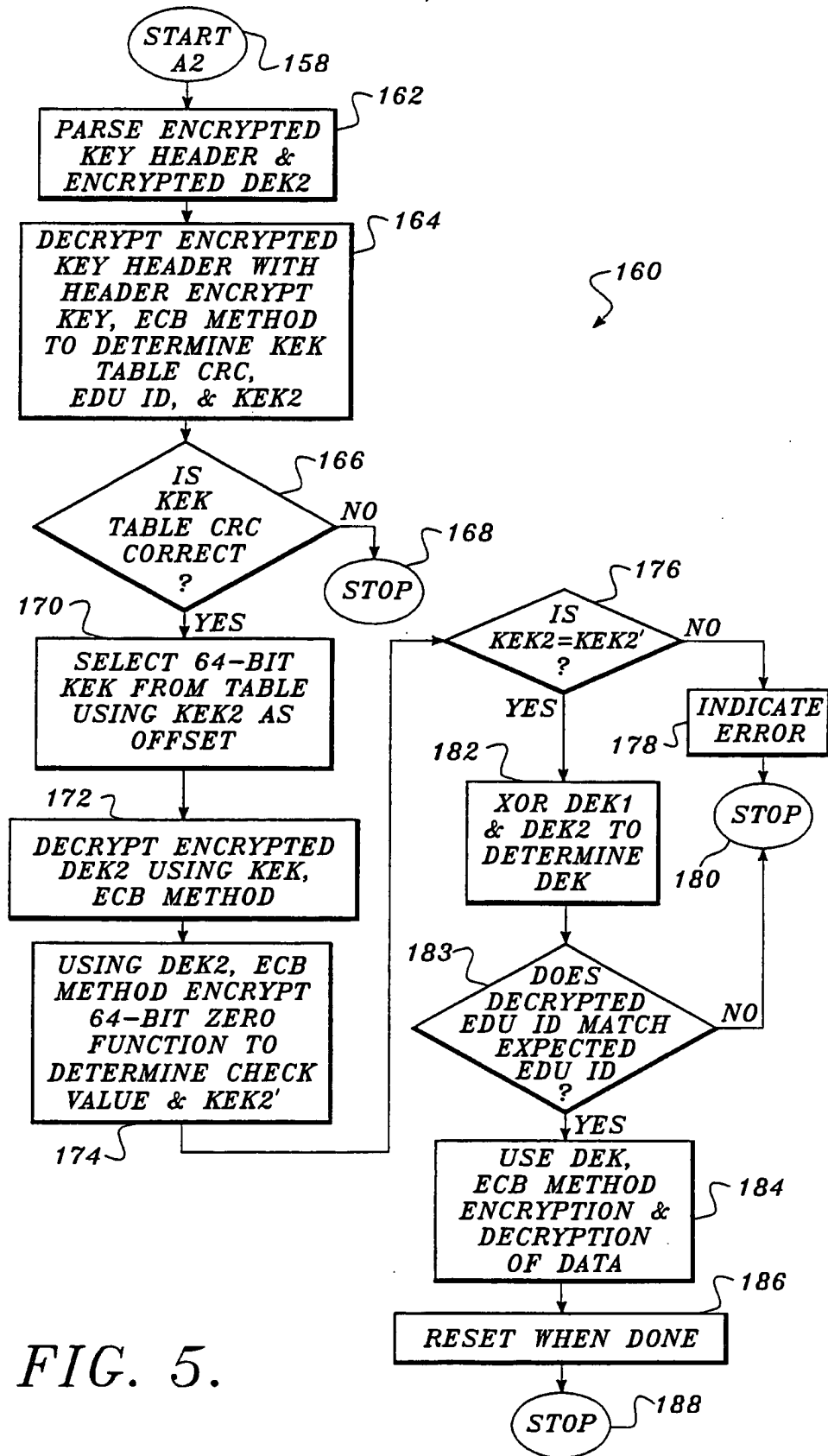
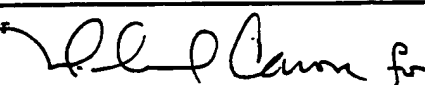


FIG. 5.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US93/04340

A. CLASSIFICATION OF SUBJECT MATTER		
IPC(5) :HO4L 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
<input type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> 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  Telephone No. 308-0463	



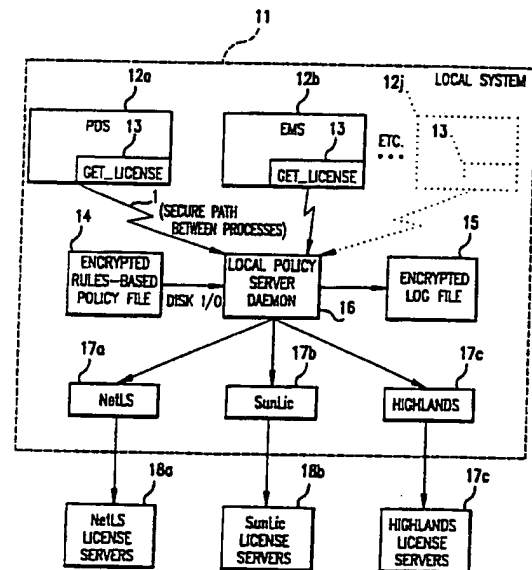
INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

<p>(51) International Patent Classification ⁵ : G06F 1/00, 11/34</p>	<p>A1</p>	<p>(11) International Publication Number: WO 93/11480 (43) International Publication Date: 10 June 1993 (10.06.93)</p>
<p>(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).</p>		<p>(81) Designated States: CA, JP, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>With international search report.</i></p>

(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	FR	France	MR	Mauritania
AU	Australia	GA	Gabon	MW	Malawi
BB	Barbados	GB	United Kingdom	NL	Netherlands
BE	Belgium	GN	Guinea	NO	Norway
BF	Burkina Faso	GR	Greece	NZ	New Zealand
BG	Bulgaria	HU	Hungary	PL	Poland
BJ	Benin	IE	Ireland	PT	Portugal
BR	Brazil	IT	Italy	RO	Romania
CA	Canada	JP	Japan	RU	Russian Federation
CF	Central African Republic	KP	Democratic People's Republic of Korea	SD	Sudan
CG	Congo	KR	Republic of Korea	SE	Sweden
CH	Switzerland	KZ	Kazakhstan	SK	Slovak Republic
CI	Côte d'Ivoire	LJ	Liechtenstein	SN	Senegal
CM	Cameroon	LK	Sri Lanka	SU	Soviet Union
CS	Czechoslovakia	LJ	Luxembourg	TD	Chad
CZ	Czech Republic	MC	Monaco	TG	Togo
DE	Germany	MG	Madagascar	UA	Ukraine
DK	Denmark	ML	Mali	US	United States of America
ES	Spain	MN	Mongolia	VN	Viet Nam
FI	Finland				

- 2 -

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

- 3 -

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
5 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
10 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
15 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
20 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.

25 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
30 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
35 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, or 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

- 5 -

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),
5 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,
10 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
15 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
20 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
25 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
30 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
35 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

5 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.

10 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
15 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
20 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
25 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
30 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
35 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

- 7 -

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
5 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.

10 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
15 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

20 followed by parameters identifying license details for the particular software product. A branching routine (which may be made available to all the software products, and called by the particular software product after this instruction) then
25 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

30 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;

35 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
5 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
10 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
15 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
20 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
25 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),
30 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
35 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

- 9 -

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

- 11 -

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 constituting 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

- 12 -

issued on a temporary basis when the server running the Licensing System becomes unavailable. A TUL is designed for emergency situations and is granted on a per user, per node, per software product, per usage history basis.

5 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
10 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
15 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
20 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
25 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
30 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
35 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

- 13 -

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 10 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 15 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 20 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 25 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 30 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 35 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

- 15 -

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
5 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
10 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
15 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
20 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).

25 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 acquisition processing (step 623) to attempt again to get a license from the license server. If the policy server database file 14 cannot be
30 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
35 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 step 723), the database is 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

- 17 -

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
5 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.

10 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 threshold. If it is, then the failure counter is cleared (step 88) and the license in question is moved to the
15 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.
20 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
25 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
30 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
35 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
5 then tested (step 931) to determine if it is above a threshold (here typically 3). If it is not above the threshold, 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 threshold, the
10 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
15 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
20 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.

- 19 -

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
5 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
10 (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
15 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
20 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
25 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
30 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
35 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.

5 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
10 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
15 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
20 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,
25 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:

30 (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
35 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

- 21 -

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
5 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.

10 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
15 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
20 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
25 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
30 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
35 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

5 (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
10 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
15 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.

20 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
25 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
30 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
35 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.

5 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
10 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,
15 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:

20 (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

25 (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
30 (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
35 computer is achieved on the basis of the license policy maintained at the given computer as well as applicable data from the license server.

- 25 -

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.

15

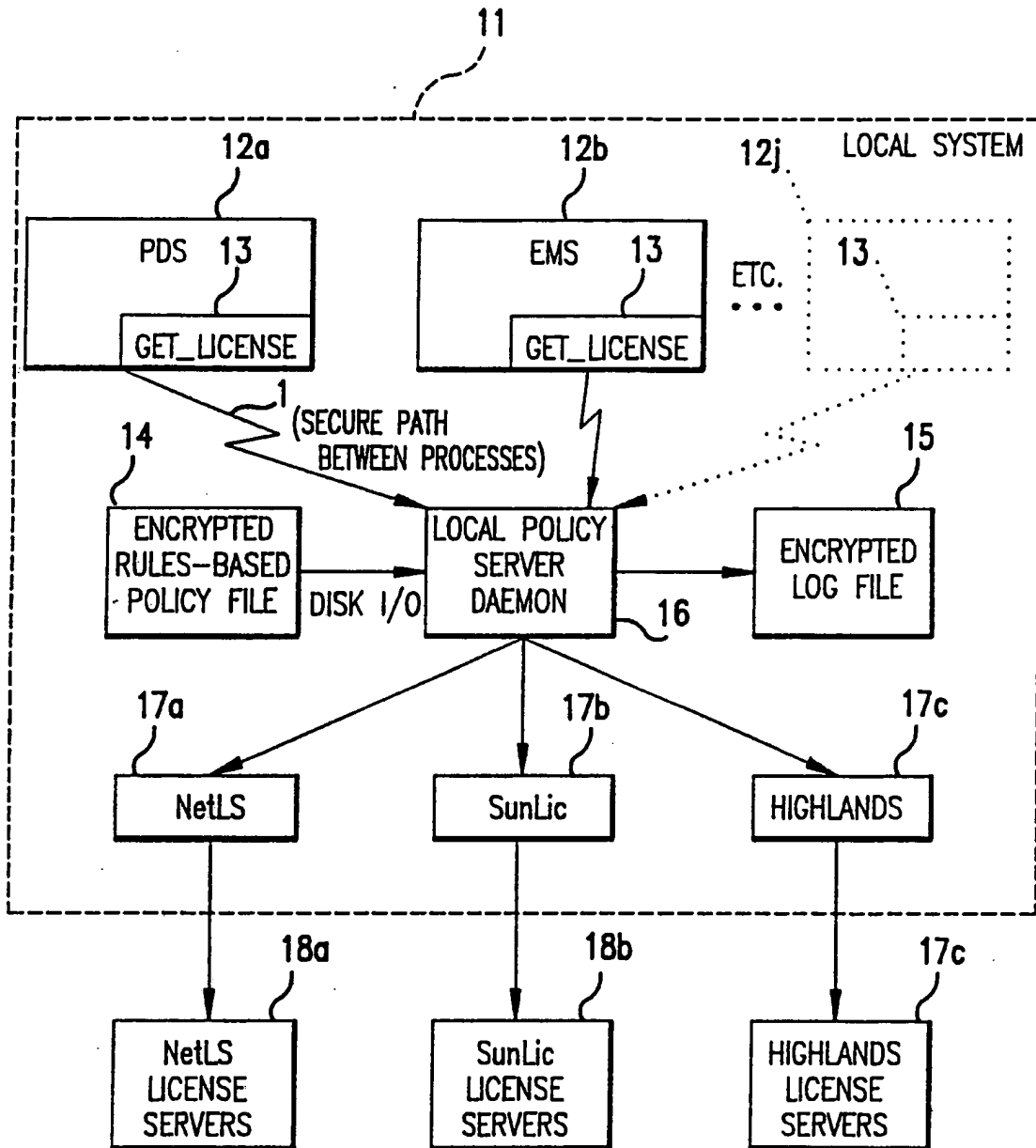
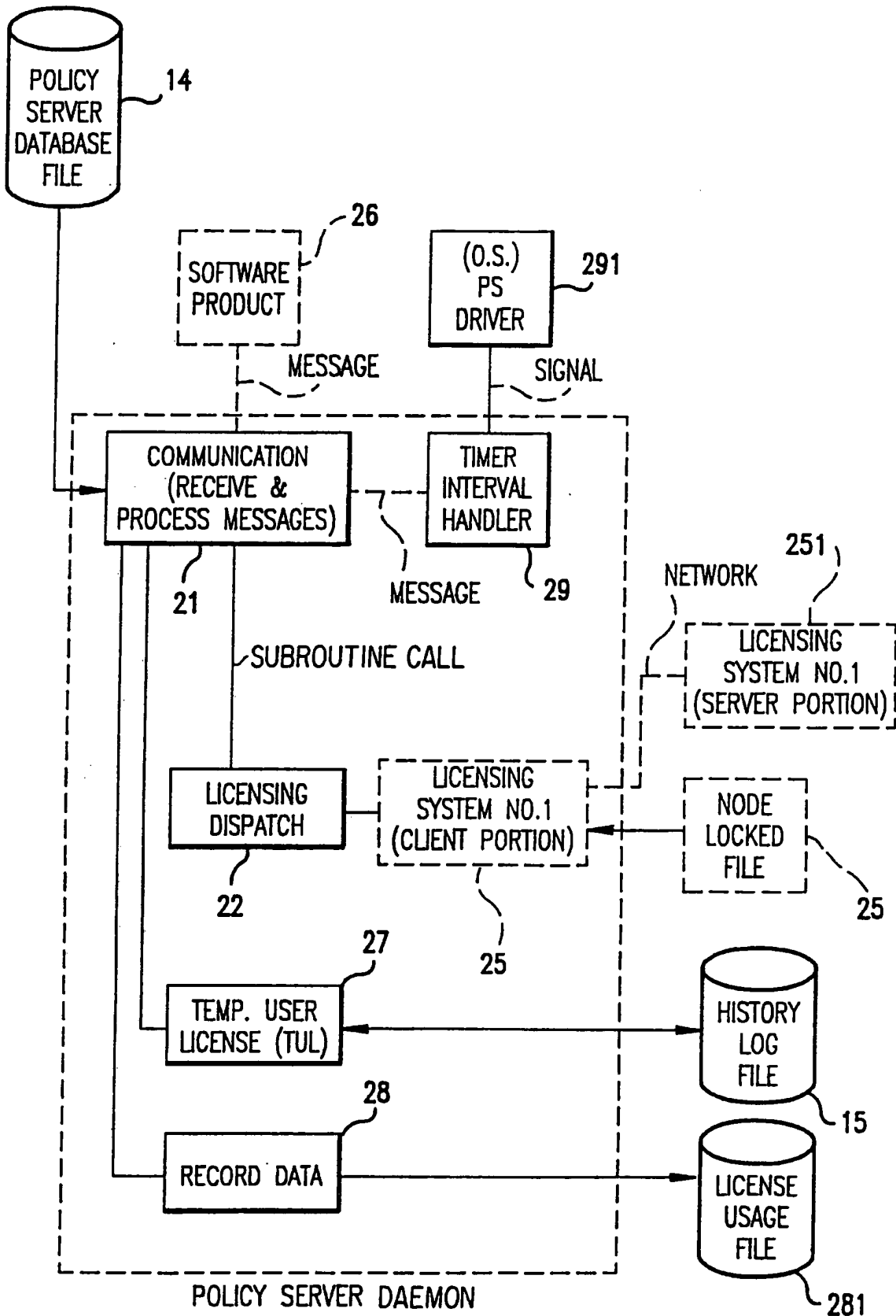


FIG. 1



SUBSTITUTE SHEET

FIG.2

3/9

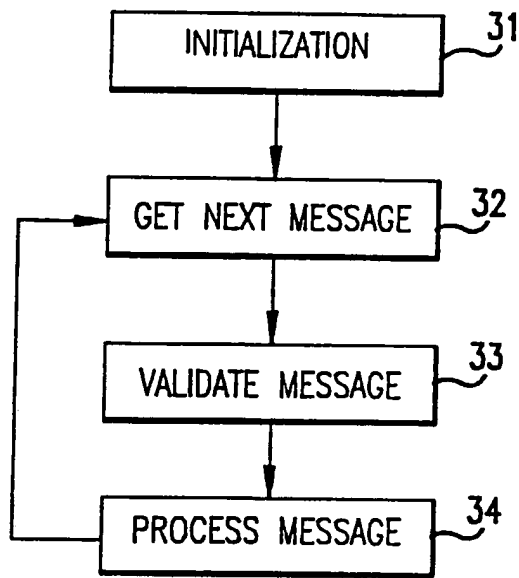


FIG.3

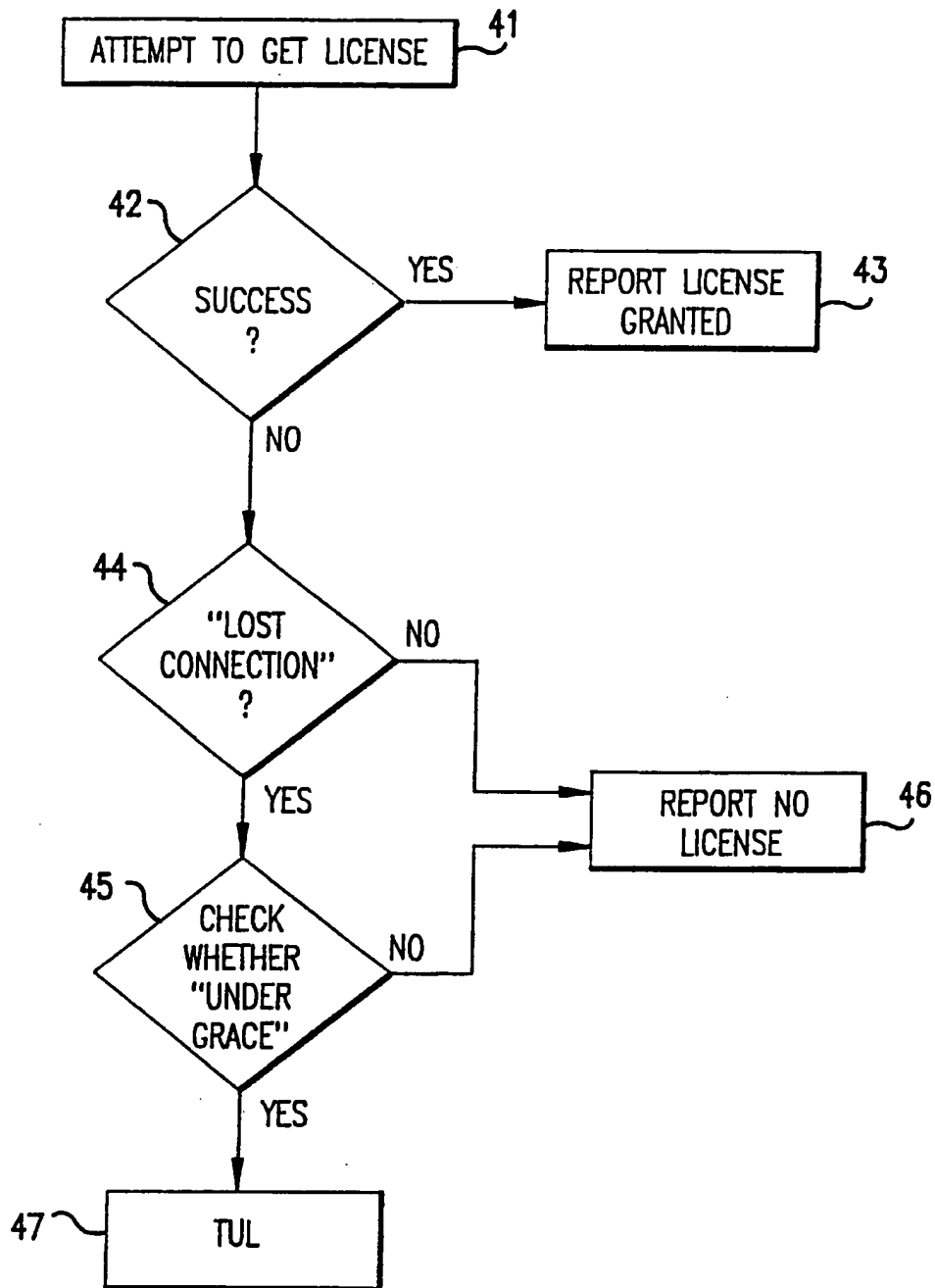


FIG.4

SUBSTITUTE SHEET

5/9

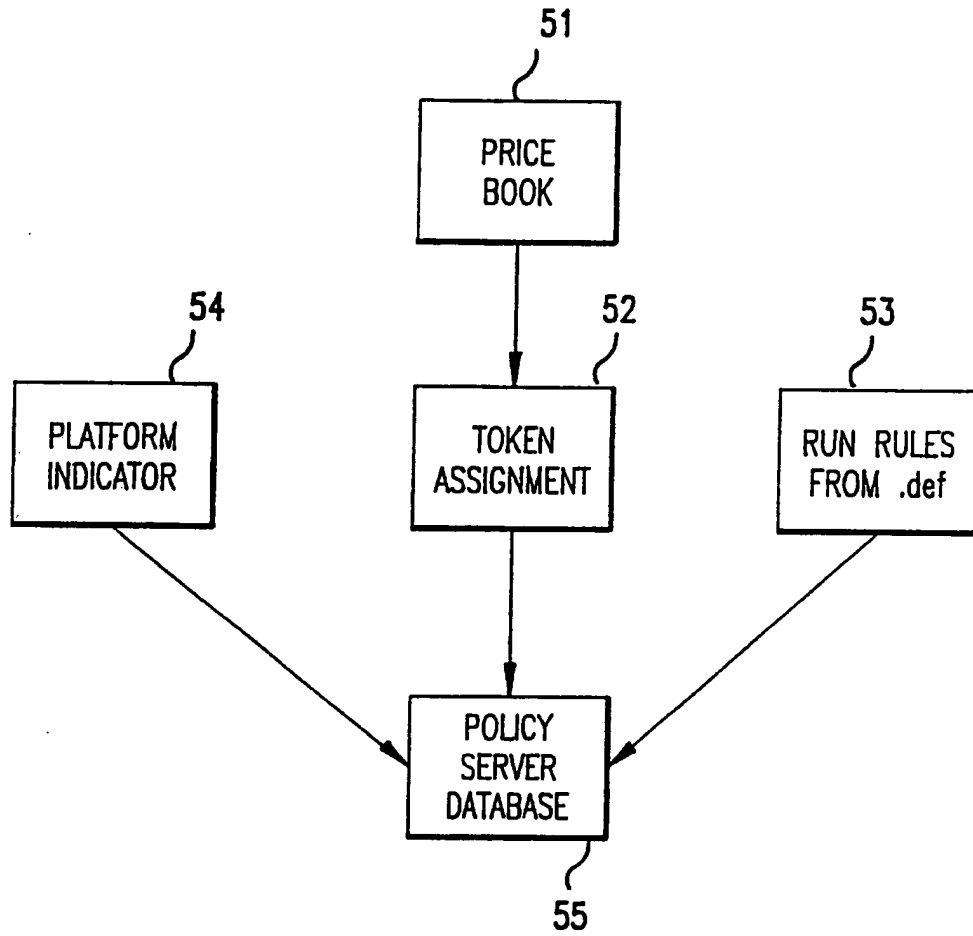


FIG.5

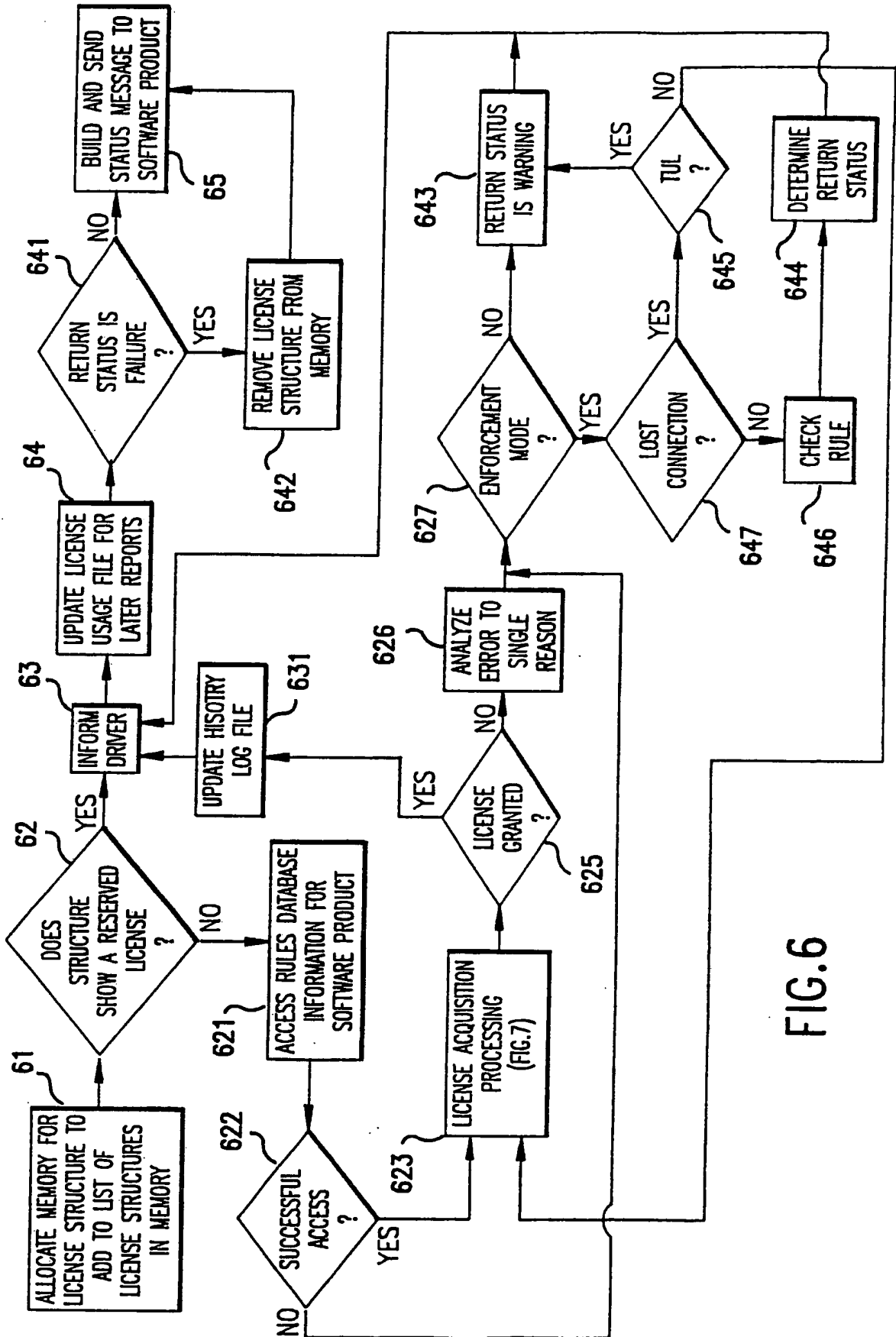


FIG.6

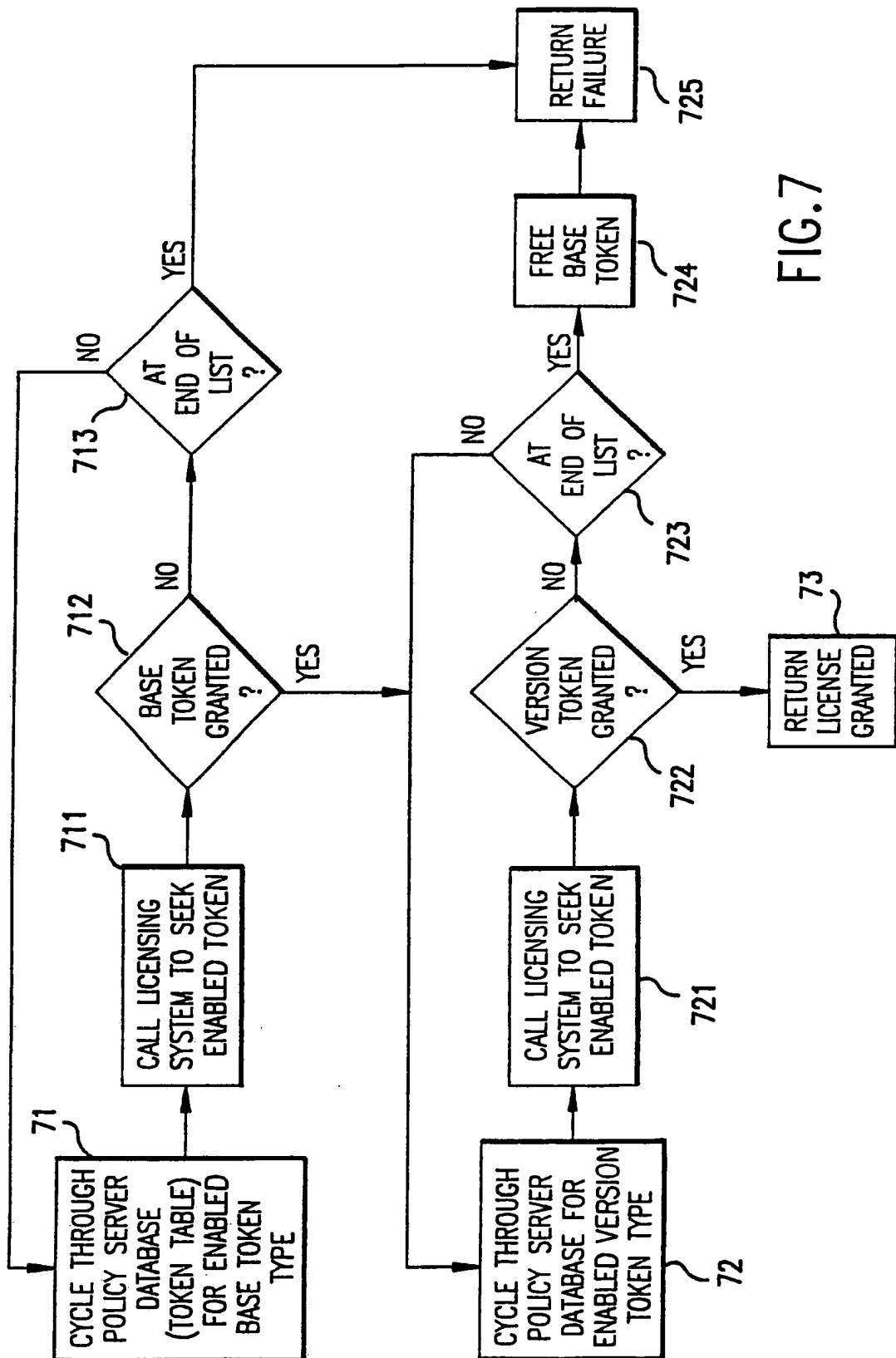


FIG. 7

ⓈUBSTITUTE SHEET

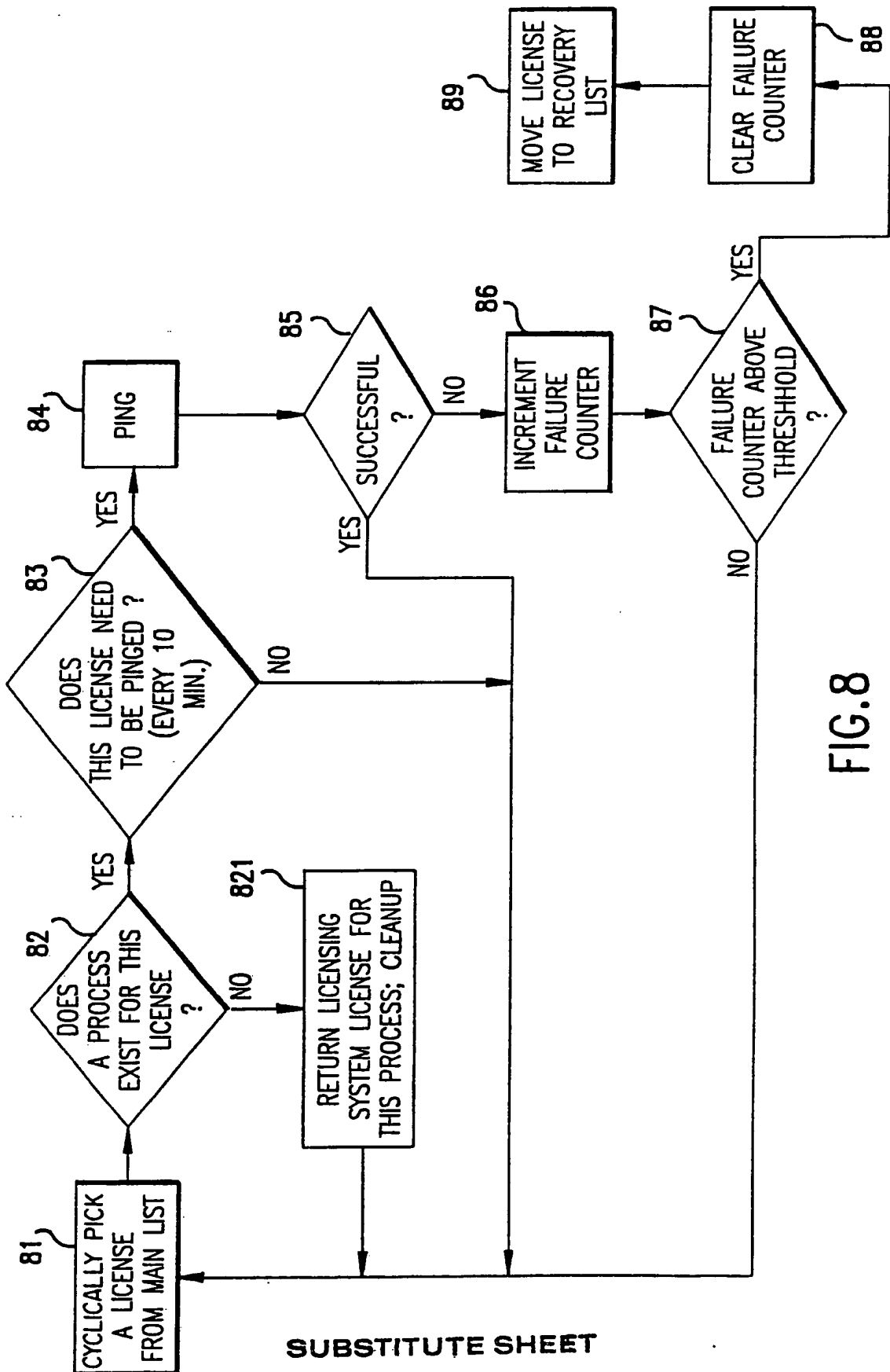


FIG. 8

SUBSTITUTE SHEET

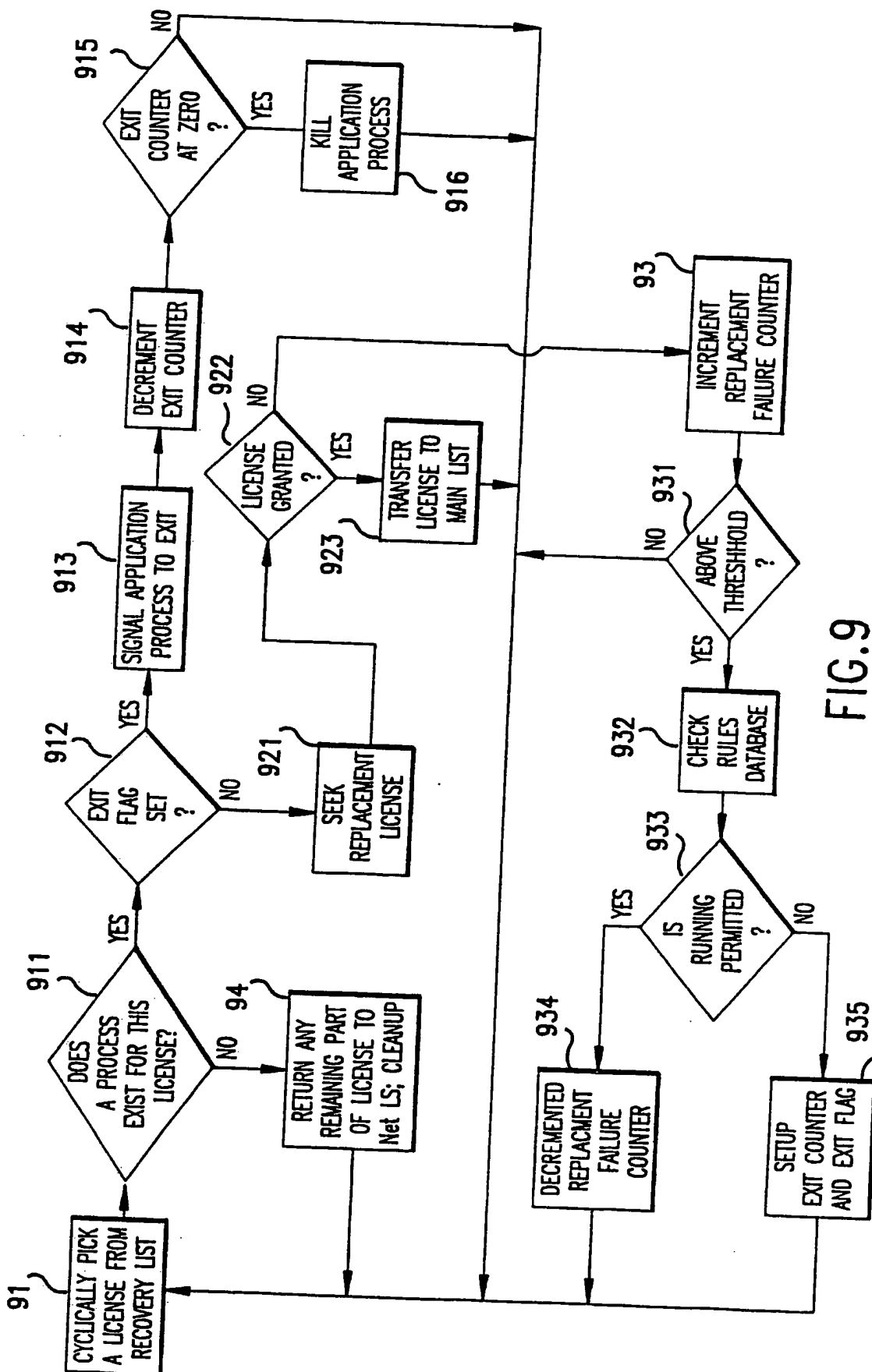


FIG. 9

INTERNATIONAL SEARCH REPORT

PCT/US 92/10215

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; G06F11/34		
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 RELEVANT⁹		
Category ¹⁰	Citation of Document, ¹¹ with indication, where appropriate, of the relevant passages ¹²	Relevant to Claim No. ¹³
A	GB,A,2 236 604 (SUN MICROSYSTEMS, INC.) 10 April 1991 see page 9, line 11 - page 10, line 28 see page 12, line 16 - page 13, line 13 see page 14, line 20 - page 16, line 27 see figures 1-3 ---	1-7, 13-15, 18-19
A	US,A,5 023 907 (APOLLO COMPUTER) 11 June 1991 see column 2, line 49 - column 5, line 42 see figures 1,2 --- -/--	1-5, 13-14, 18-19
<p>¹⁰ Special categories of cited documents : ¹⁰</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier document but published on or after the international filing date</p> <p>"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)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p> <p>"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</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step</p> <p>"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.</p> <p>"A" document member of the same patent family</p>		
IV. CERTIFICATION		
Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report	
26 FEBRUARY 1993	23.03.93	
International Searching Authority	Signature of Authorized Officer	
EUROPEAN PATENT OFFICE	JOHANSSON U.C.	

III. DOCUMENTS CONSIDERED TO BE RELEVANT (CONTINUED FROM THE SECOND SHEET)		Relevant to Claim No.
Category °	Citation of Document, with indication, where appropriate, of the relevant passages.	
A	EP, A, 0 332 304 (DIGITAL EQUIPMENT CORP.) 13 September 1989 see column 3, line 31 - column 6, line 8 see column 6, line 42 - column 7, line 23 see column 8, line 33 - column 9, line 44 see figure 1 -----	1,2, 10-13,18

Form PCT/ISA/210 (extra sheet) (January 1985)

**ANNEX TO THE INTERNATIONAL SEARCH REPORT
ON INTERNATIONAL PATENT APPLICATION NO.**

US 9210215
SA 67461

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 The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information. 26/02/93

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
GB-A-2236604	10-04-91	US-A- 5138712	11-08-92
		CA-A- 2025434	03-04-91
		JP-A- 4100148	02-04-92

US-A-5023907	11-06-91	None	

EP-A-0332304	13-09-89	US-A- 4937863	26-06-90
		JP-A- 2014321	18-01-90

EPO FORM P0079

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

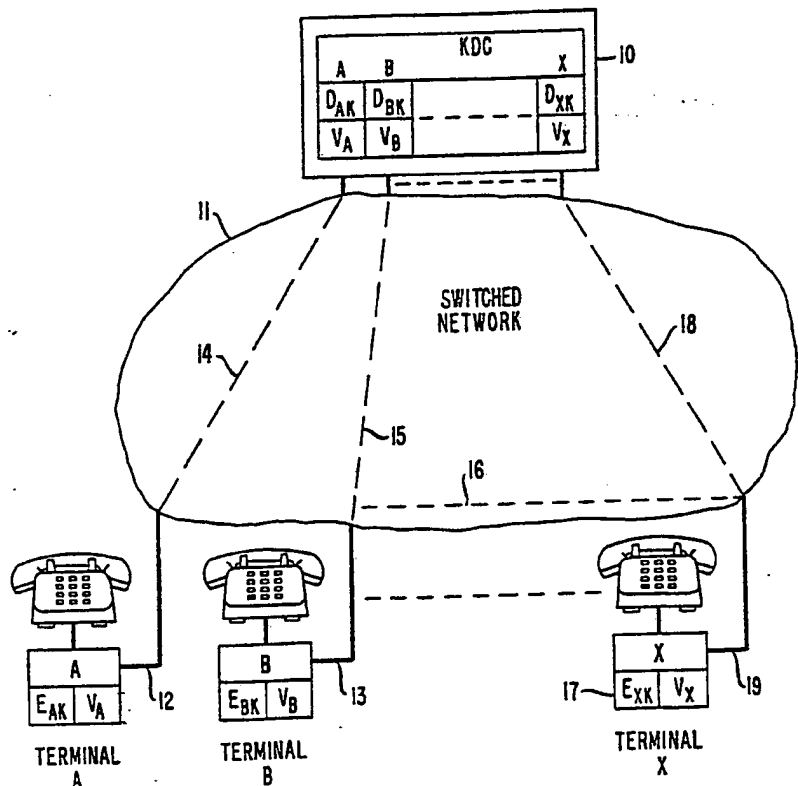
<p>(51) International Patent Classification³ : H04L 9/00</p>	<p>A1</p>	<p>(11) International Publication Number: WO 83/ 04461 (43) International Publication Date: 22 December 1983 (22.12.83)</p>
<p>(21) International Application Number: PCT/US83/00030 (22) International Filing Date: 11 January 1983 (11.01.83) (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).</p>	<p>(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 <i>With international search report.</i></p>	

(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.

KDC CONFIGURATION



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		

- 1 -

ENCRYPTION SYSTEM KEY DISTRIBUTION
METHOD AND APPARATUS

Background of the Invention

5 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
10 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
15 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
20 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.

25 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
30 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.
35 While such attempts have found some degree of success they all suffer from the problem that they are subject to



- 2 -

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

SUBSTITUTE SHEET

Petitioner Apple Inc. - Exhibit 1002, p. 257.

BUREAU
OMPI

- 3 -

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

5 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

10 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

15 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

20 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

25 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.

30 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

35 communicating terminals, and also must continue the above substitutions on the channels for an indefinite period of time or risk detection.

SUBSTITUTE SHEET

Petitioner Apple Inc. - Exhibit 1002, p. 258



- 4 -

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

SUBSTITUTE SHEET

Petitioner Apple Inc. - Exhibit 1002, p. 259



- 5 -

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

SUBSTITUTE SHEET



- 6 -

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
5 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
10 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
15 key stored in the terminal-unique key storage at the KDC under the address of that terminal. In addition, a random number, U_a 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
20 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,
25 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
30 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
35 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

SUBSTITUTE SHEET

- 7 -

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.

5 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.

10 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

15 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

20 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

25 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

30 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

35 performs the same function regardless of encryption or decryption). The output will be the inputted bits encrypted or decrypted with the supplied key. This

SUBSTITUTE SHEET



- 8 -

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

SUBSTITUTE SHEET

- 9 -

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.

5 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,
10 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
15 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
20 the aforementioned paper by Diffie.

There is a semi-permanent storage at the KDC, registers 214 and 216, which stores verification information V_a and terminal-unique decryption key information D_{ak} between calls. Semi-permanent
25 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
30 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
35 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

SUBSTITUTE SHEET



- 10 -

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:

$$Val' = f (Val, E1) \text{ and } Va2' = f (Va2, E2)$$

where ' denotes updated and $ValVa2 = Va$. 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 V1, 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

SUBSTITUTE SHEET



- 11 -

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

SUBSTITUTE SHEET

- 12 -

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.

5 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.

10 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*.

15 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.

20 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

SUBSTITUTE SHEET

Petitioner Apple Inc. - Exhibit 1002, WFO 267.

BUREAU
OMPI

- 13 -

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
5 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
10 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
15 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
20 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
25 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
30 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
35 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.

SUBSTITUTE SHEET

- 14 -

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 V_a information and both decrypt keys D_{ka} and D_{ba} 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 D_{ka} 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 E_{ak} 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 E_{ab} which was generated at B and sent through the KDC to terminal A. The third piece of information is the updated verification information V_a , 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

SUBSTITUTE SHEET

- 15 -

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

SUBSTITUTE SHEET

- 16 -

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.

SUBSTITUTE SHEET

- 17 -

Claims

1. A key distribution method for communicating cipher keys between two terminals via a key distribution center, KDC, said method comprising
- 5 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
- 10 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.
- 15 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.
- 20 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
- 25 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
- 30 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
- 35 exchanging information with said accessed terminal using, as a foundation for said exchange, said priorly established communication cipher keys, and

SUBSTITUTE SHEET



- 18 -

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
5 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.

10 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.

15 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.

20 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
25 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
30 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
35 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.

SUBSTITUTE SHEET

FIG. 1
KDC CONFIGURATION

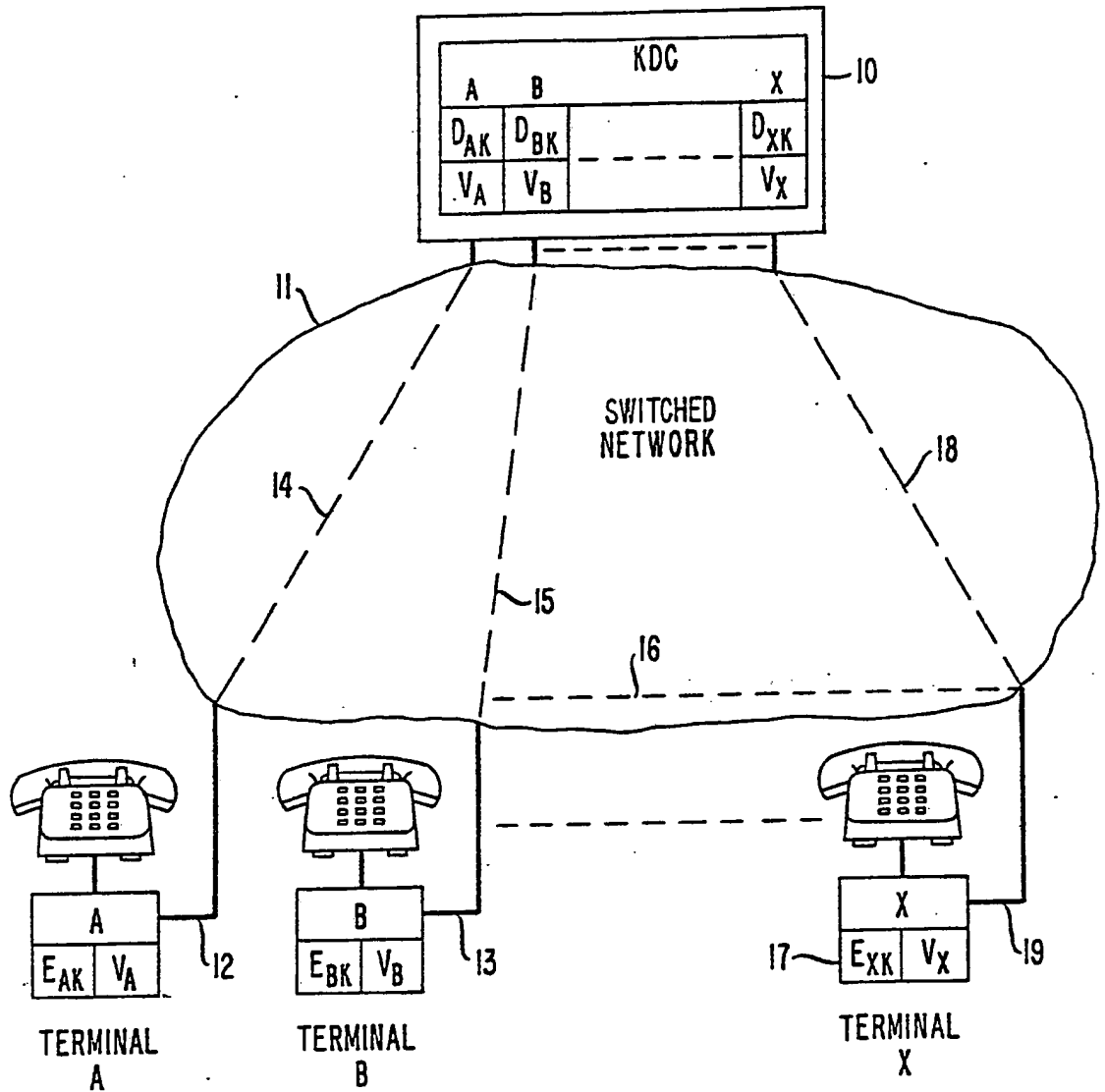
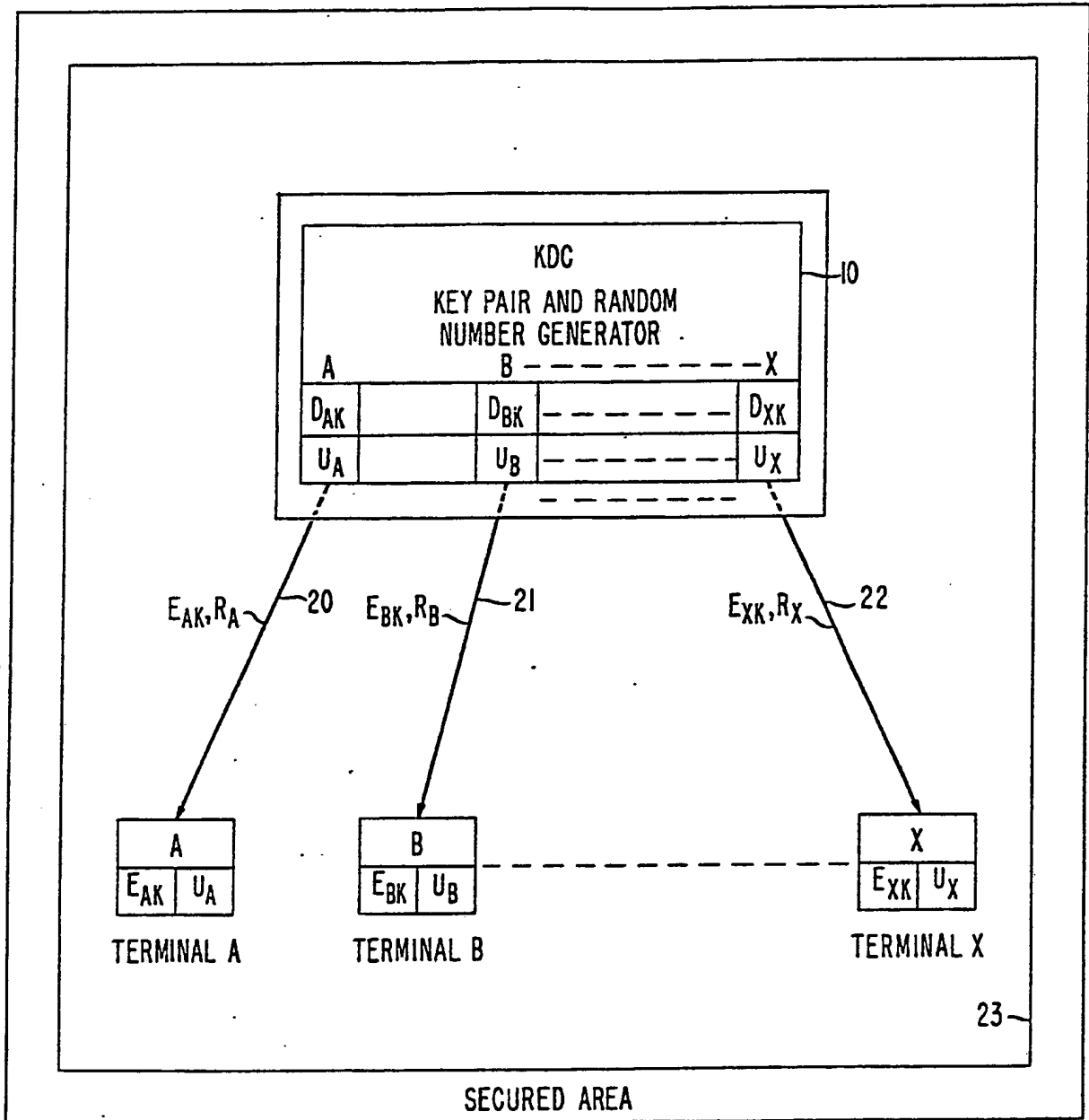
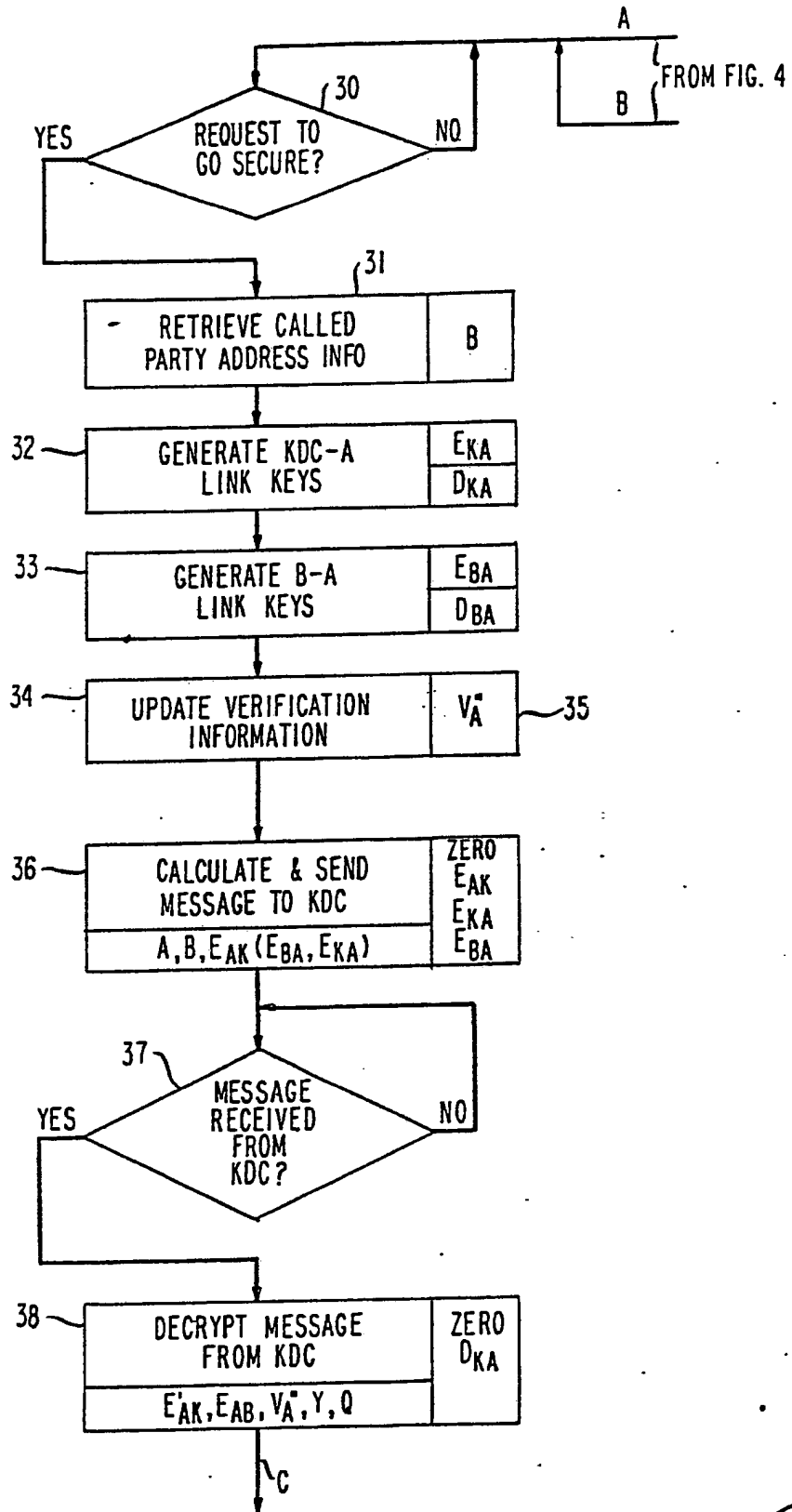


FIG. 2
INITIAL SYSTEM SETUP



3/17

FIG. 3
TERMINAL A



TO FIG. 4



FIG. 4
TERMINAL A

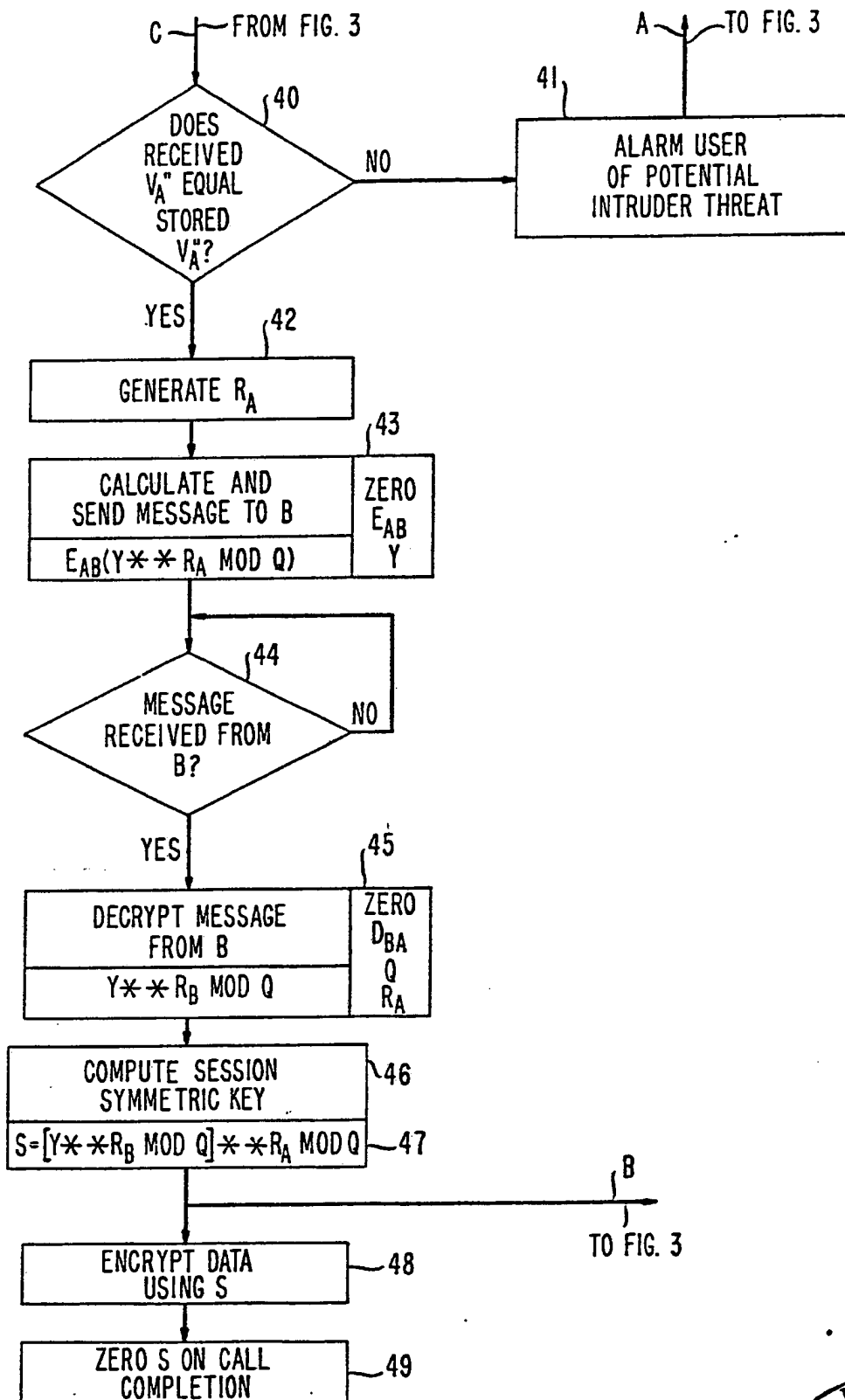
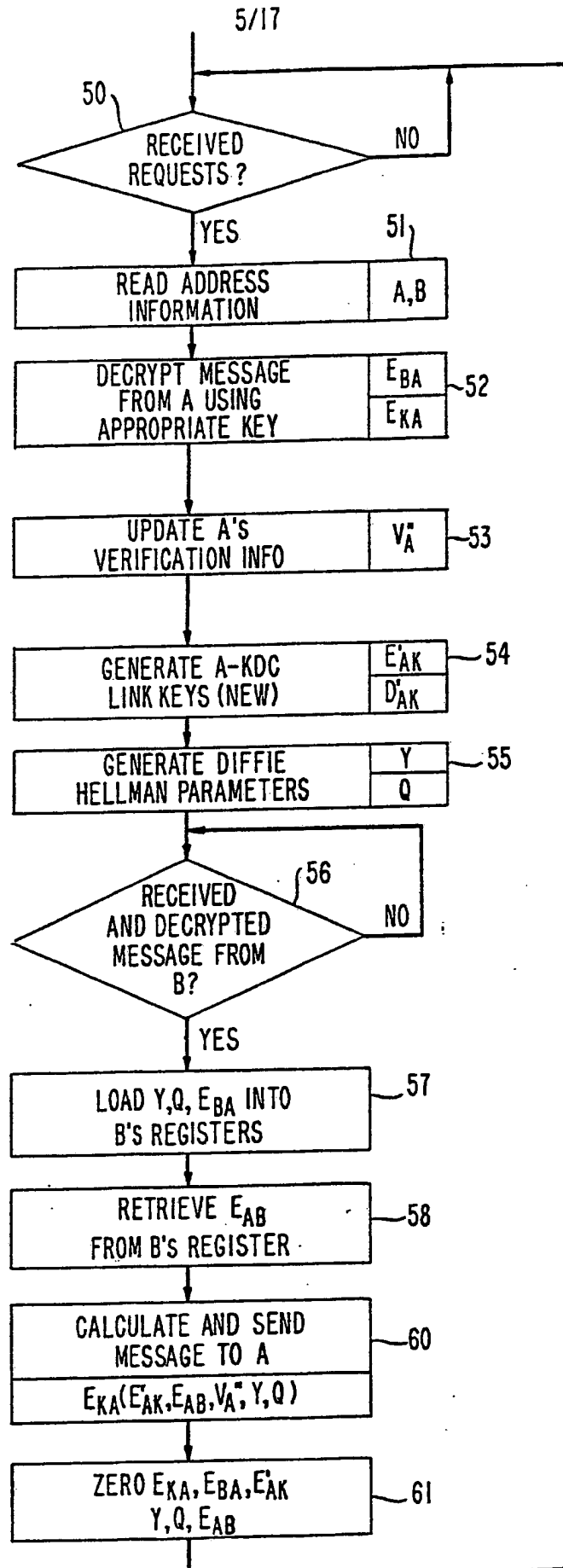


FIG. 5
KDC



6/17

FIG. 6 BETWEEN CALLS IDLE STATE (FOR TERMINAL A)

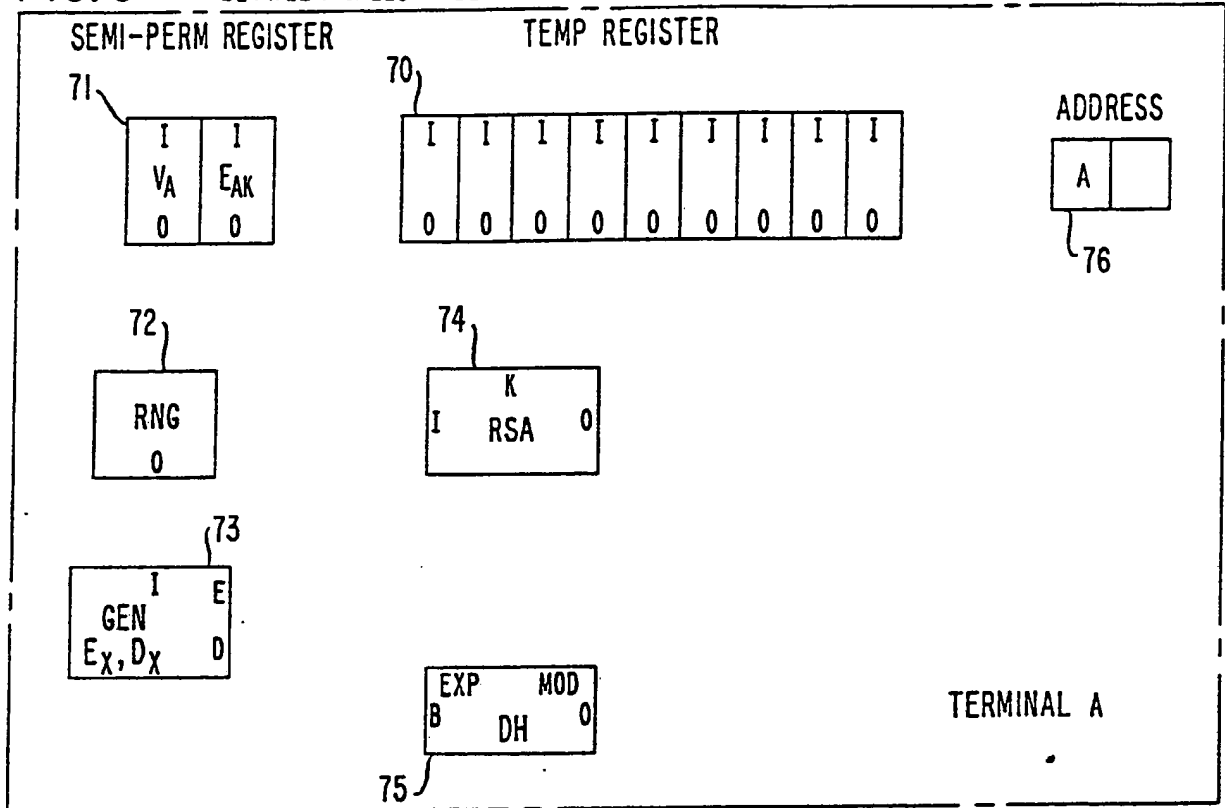
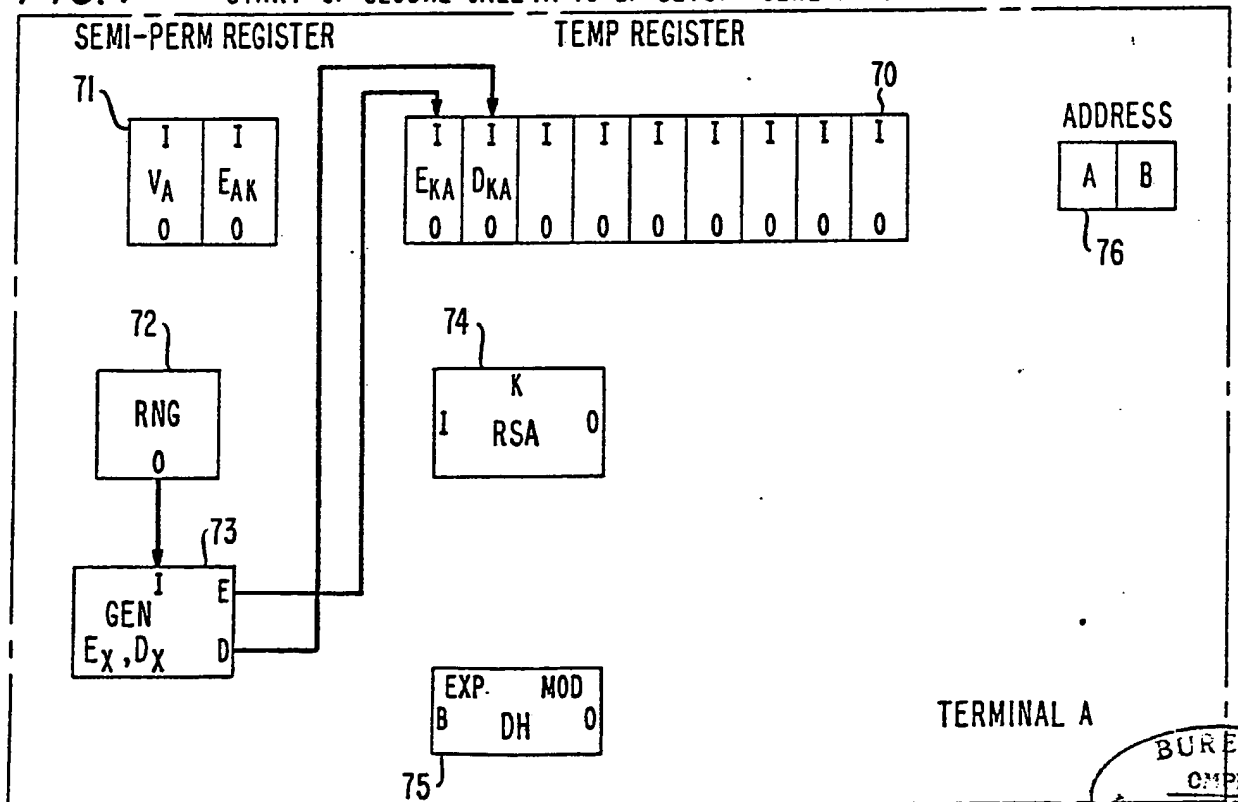


FIG. 7 START OF SECURE CALL (A TO B) SETUP - GENERATION OF KDC-A LINK KEYS



BUREAU
OMPI

FIG. 8 GENERATION OF B-A LINK KEYS

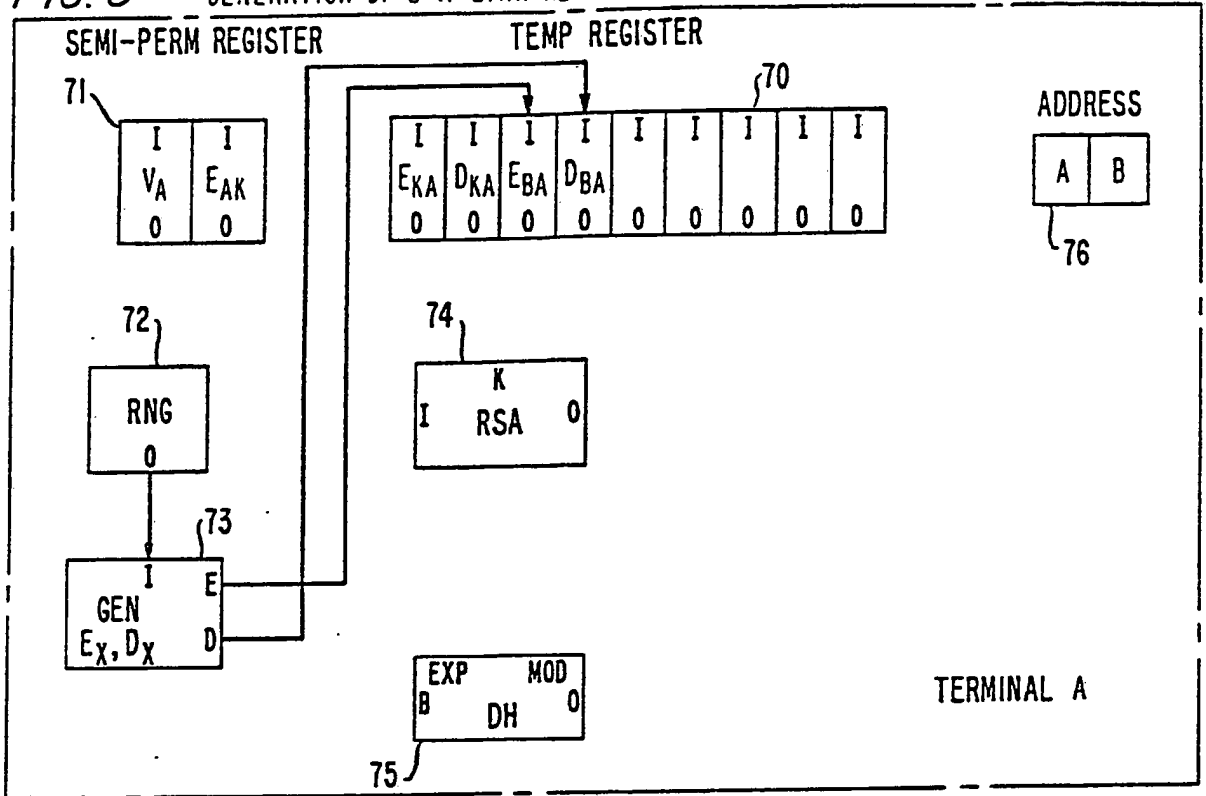


FIG. 9 FIRST STEP IN UPDATE OF VERIFICATION INFORMATION

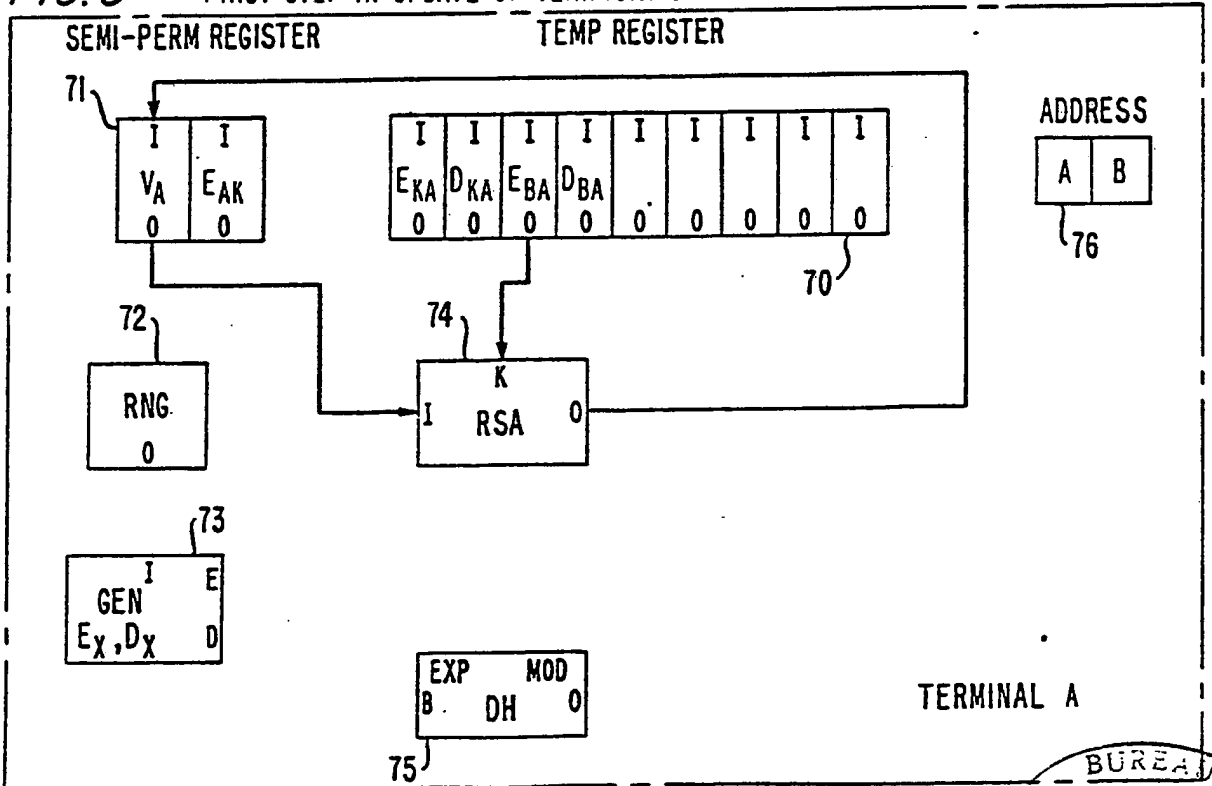


FIG. 10 SECOND STEP IN UPDATE OF VERIFICATION INFORMATION

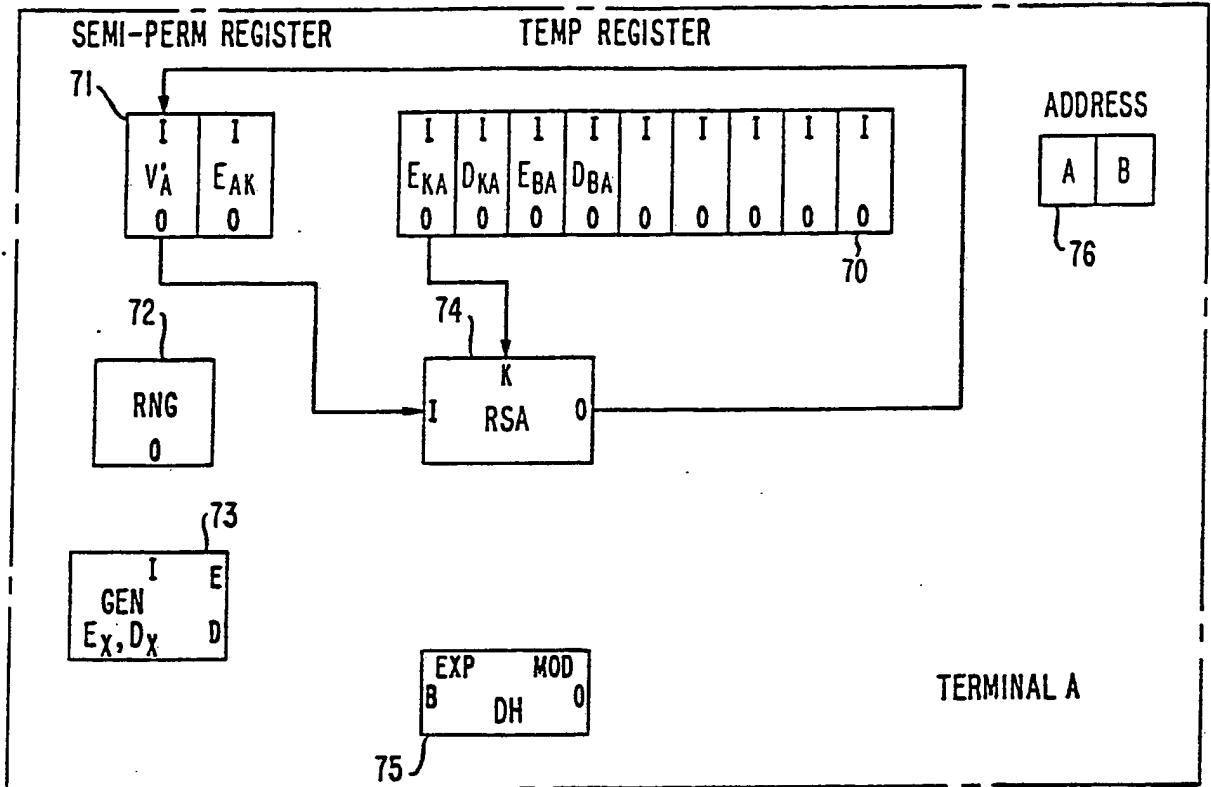
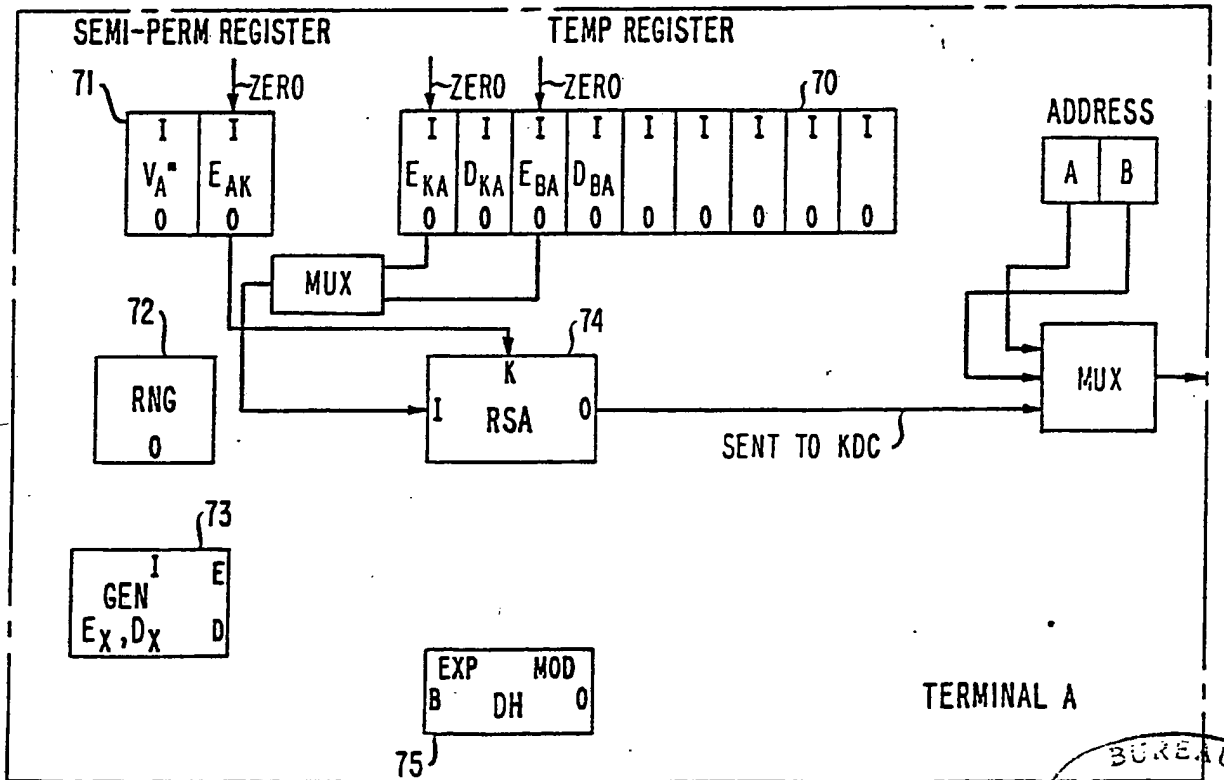


FIG. 11 COMPUTATION OF A-KDC MESSAGE



BUREAU

9/17

FIG. 12 IDLE STATE WHILE WAITING FOR RETURN MESSAGE FROM KDC

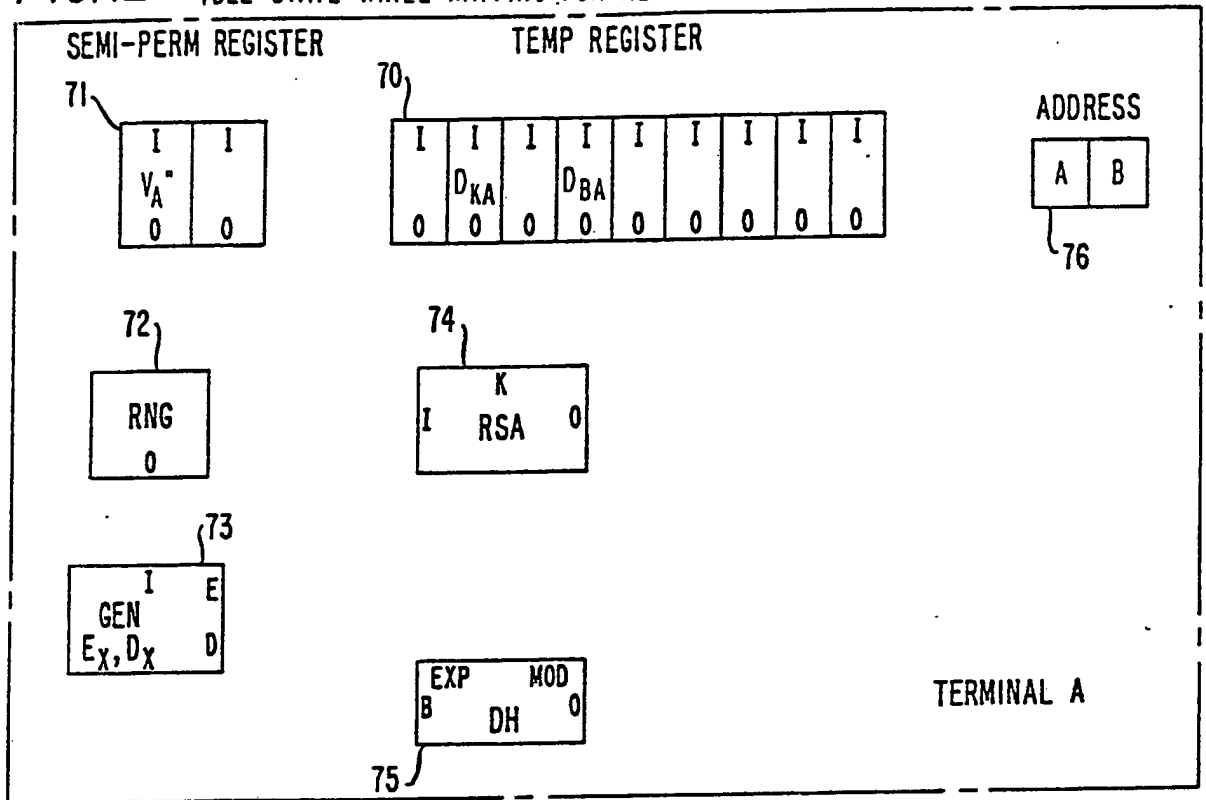
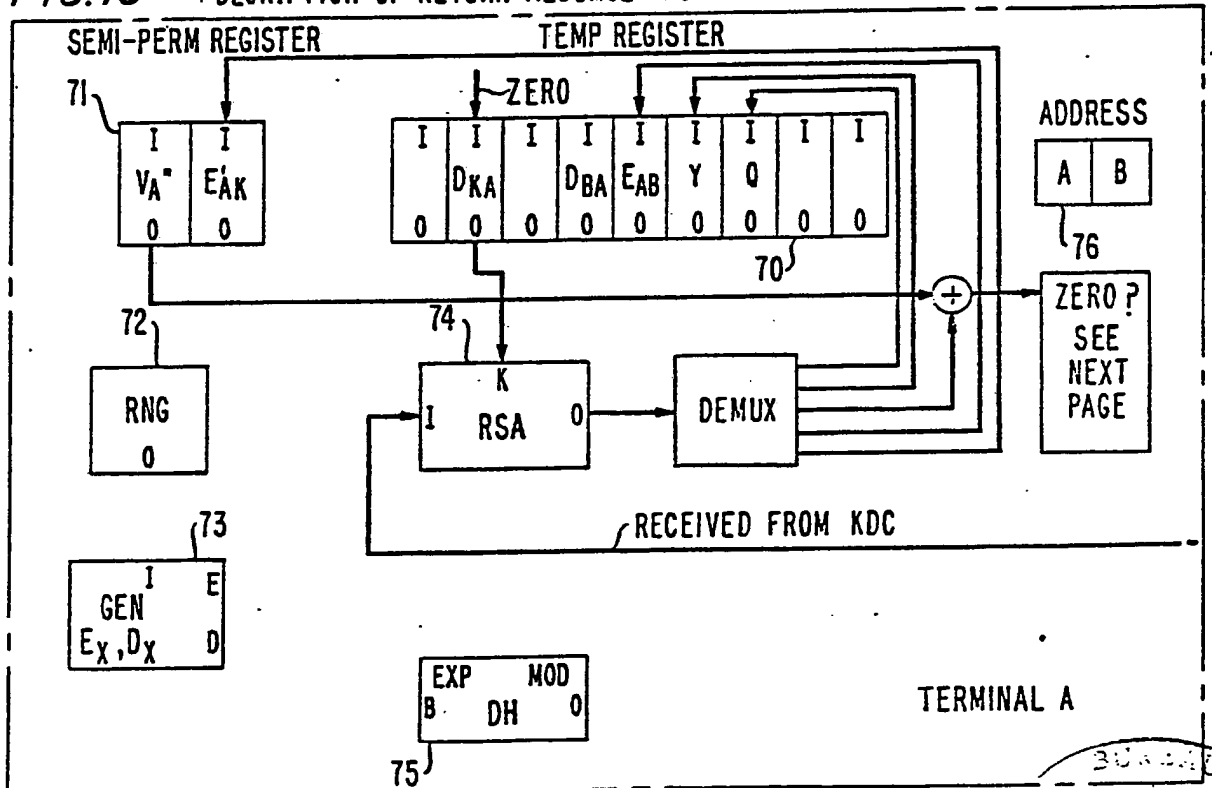


FIG. 13 DECRYPTION OF RETURN MESSAGE FROM KDC



10/17

FIG. 14 VERIFICATION CHECK

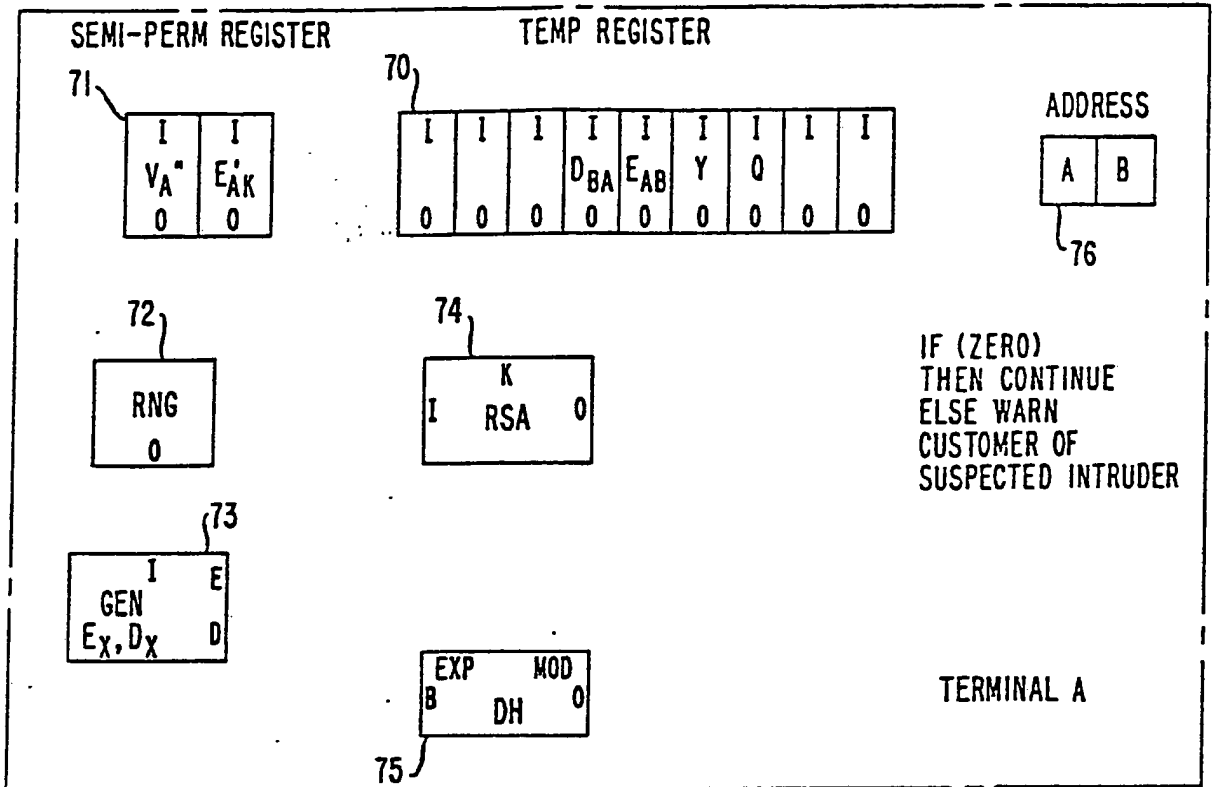
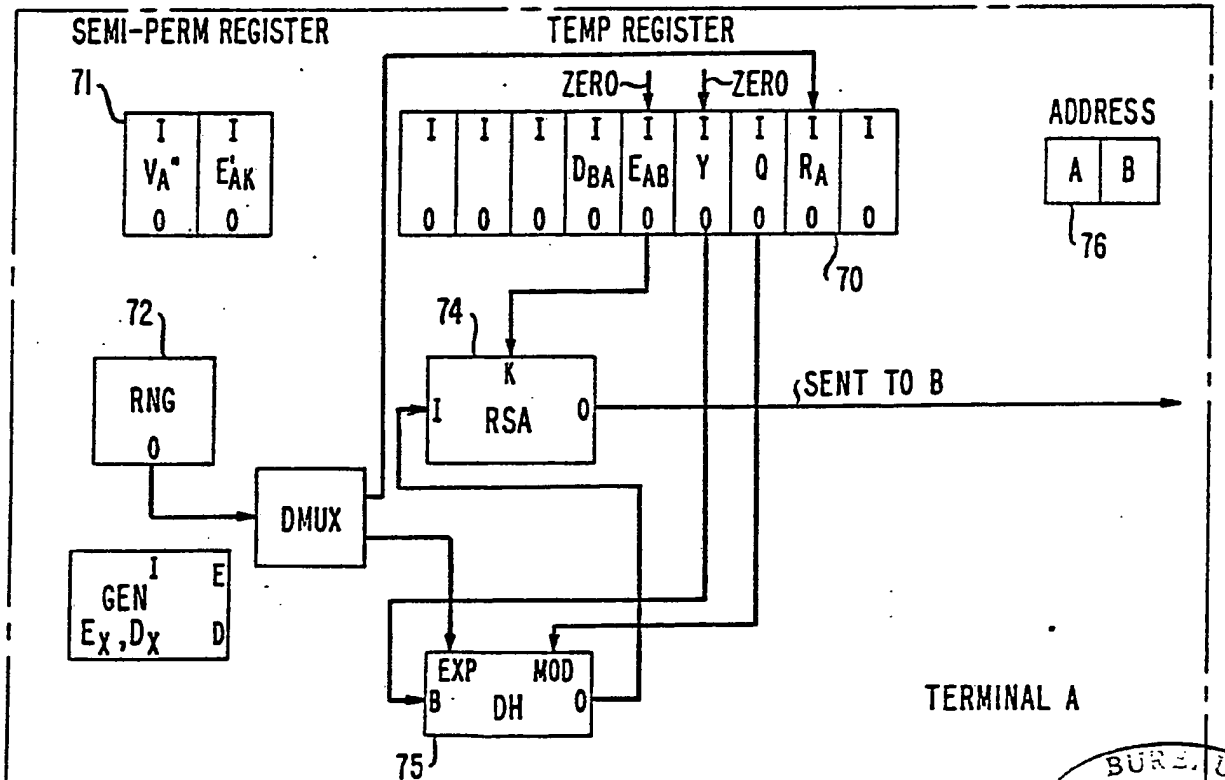


FIG. 15 START OF KEY EXCHANGE WITH B CALCULATION OF DIFFIE-HELLMAN KEYS



11/17

FIG. 16 IDLE WAIT STATE FOR RETURN MESSAGE FROM B

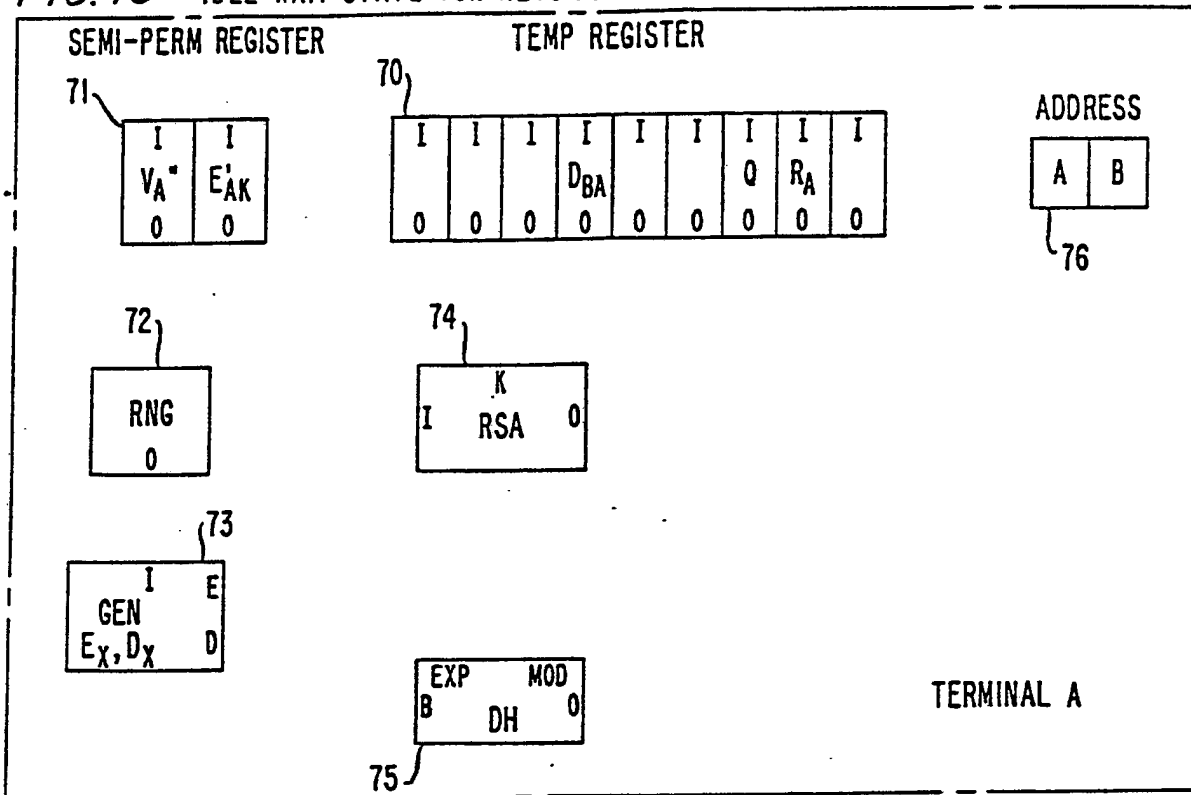
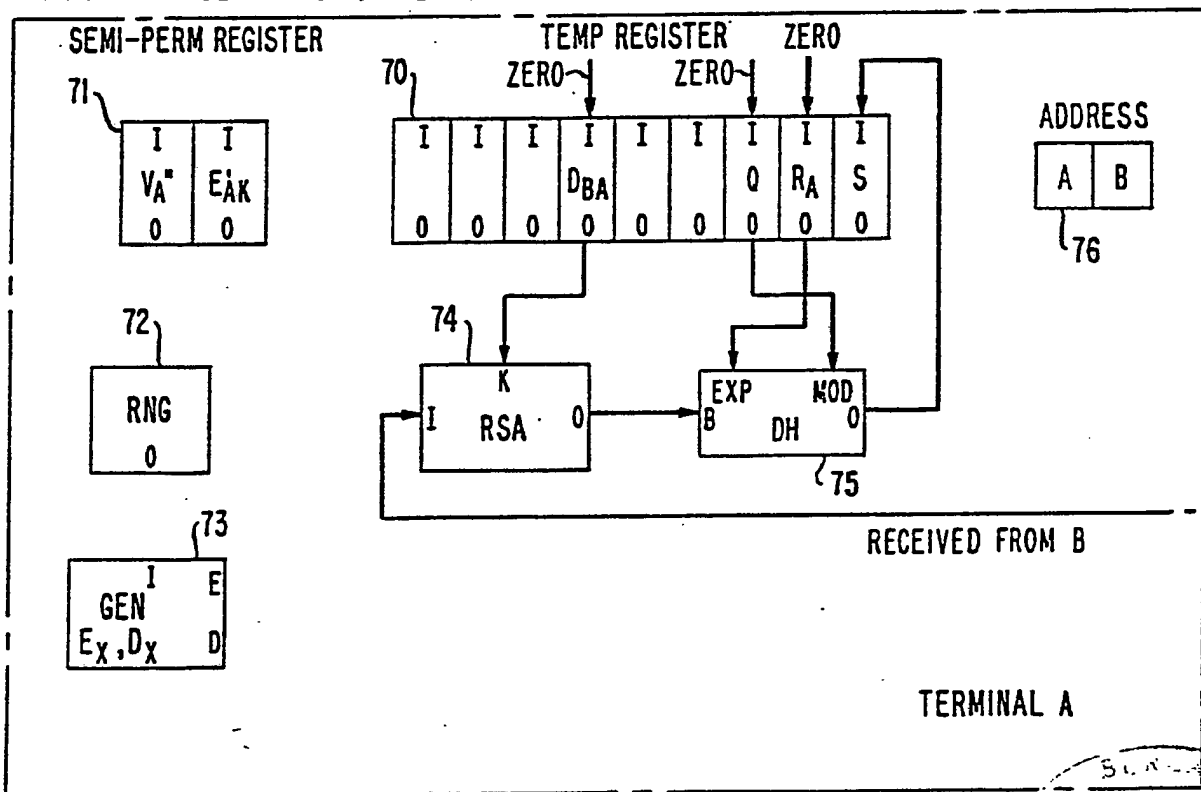


FIG. 17 DECRYPTION OF MESSAGE FROM B AND CALCULATION OF SESSION KEY-S



12/17

FIG. 18 KEY STORAGE DURING CALL

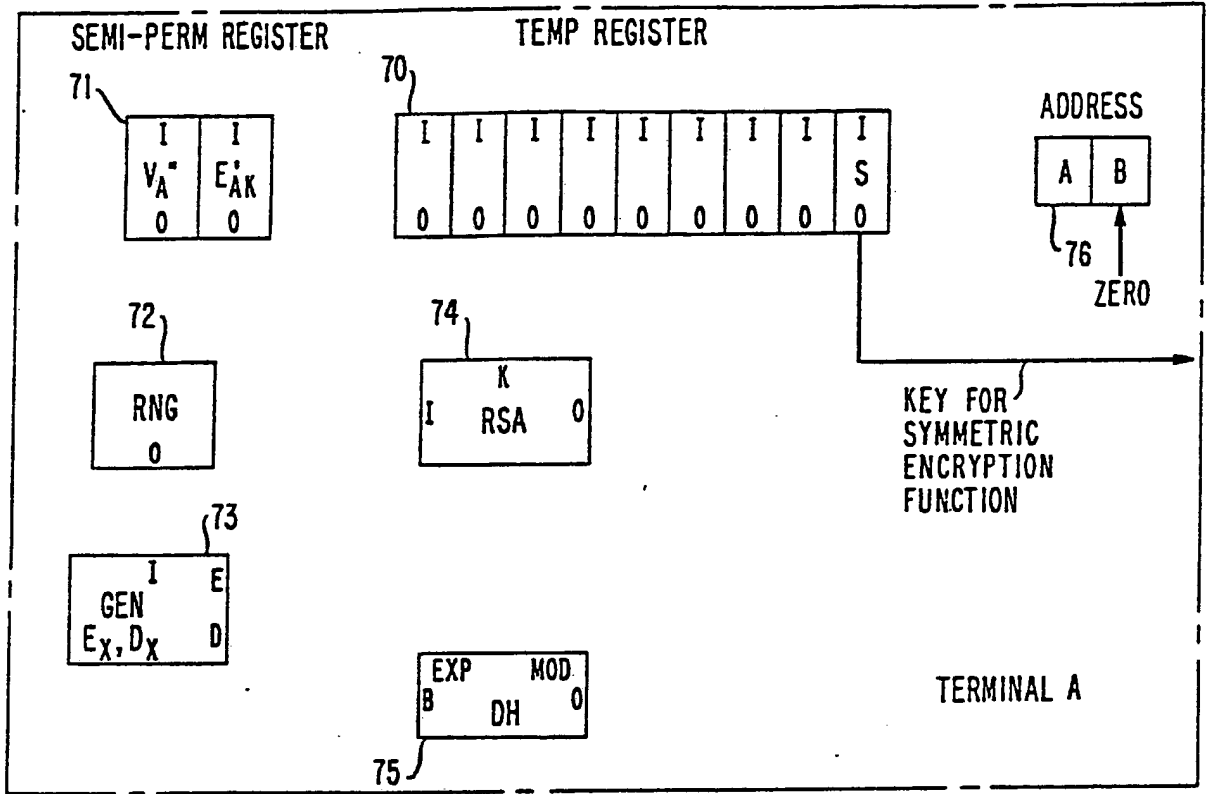


FIG. 19 IDLE STATE FOLLOWING CALL COMPLETION

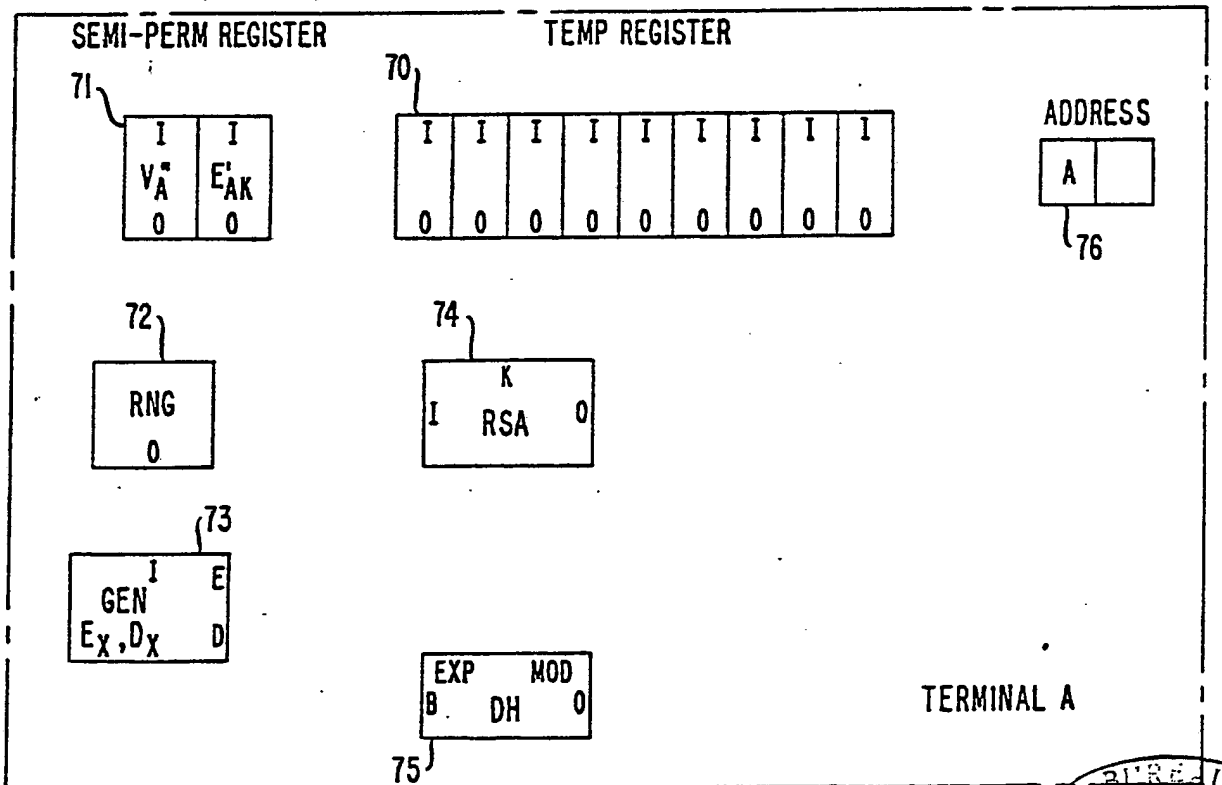


FIG. 20 IDLE STATE BETWEEN CALLS

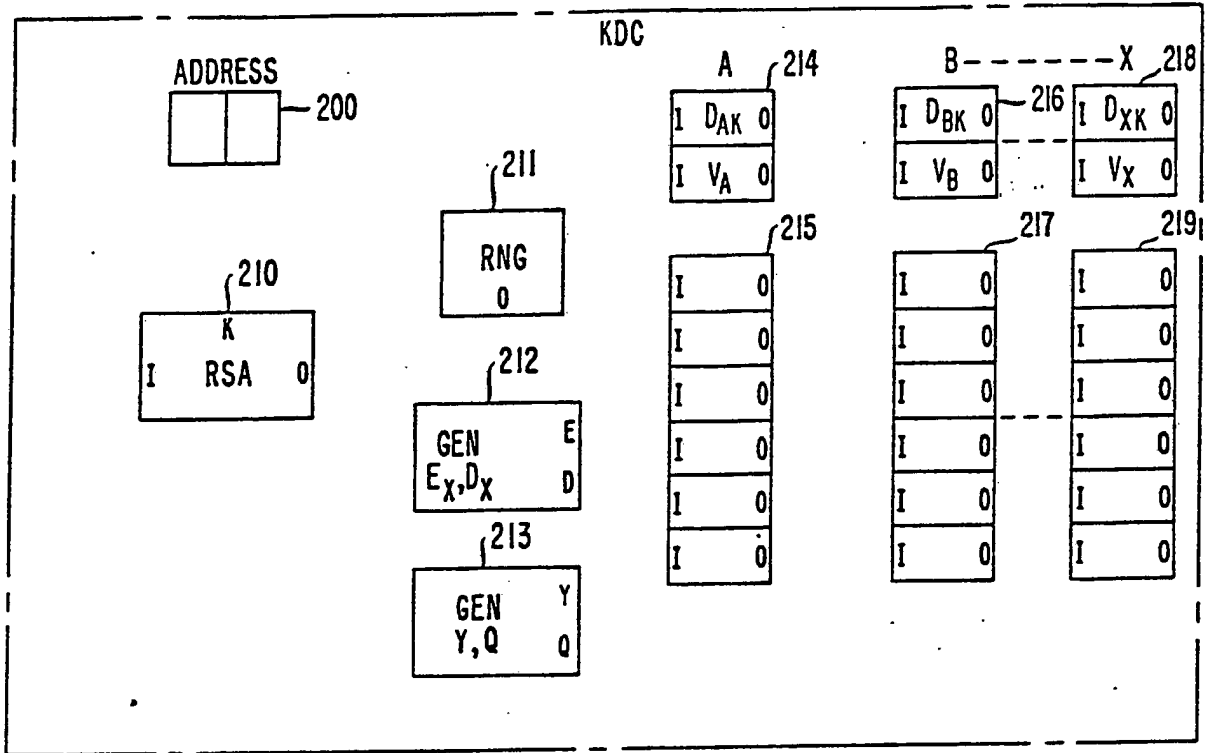


FIG. 21 DECRYPTION OF MESSAGE FROM A

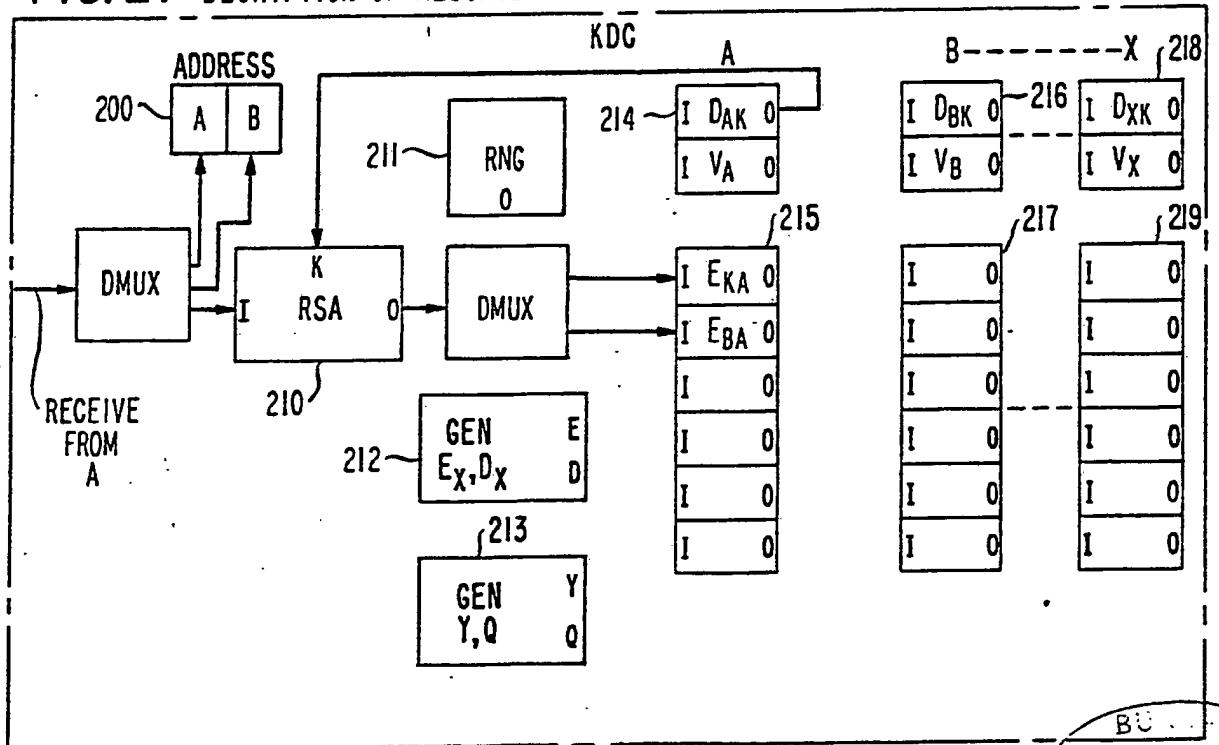


FIG. 22 FIRST STEP IN THE UPDATE OF VERIFICATION INFORMATION

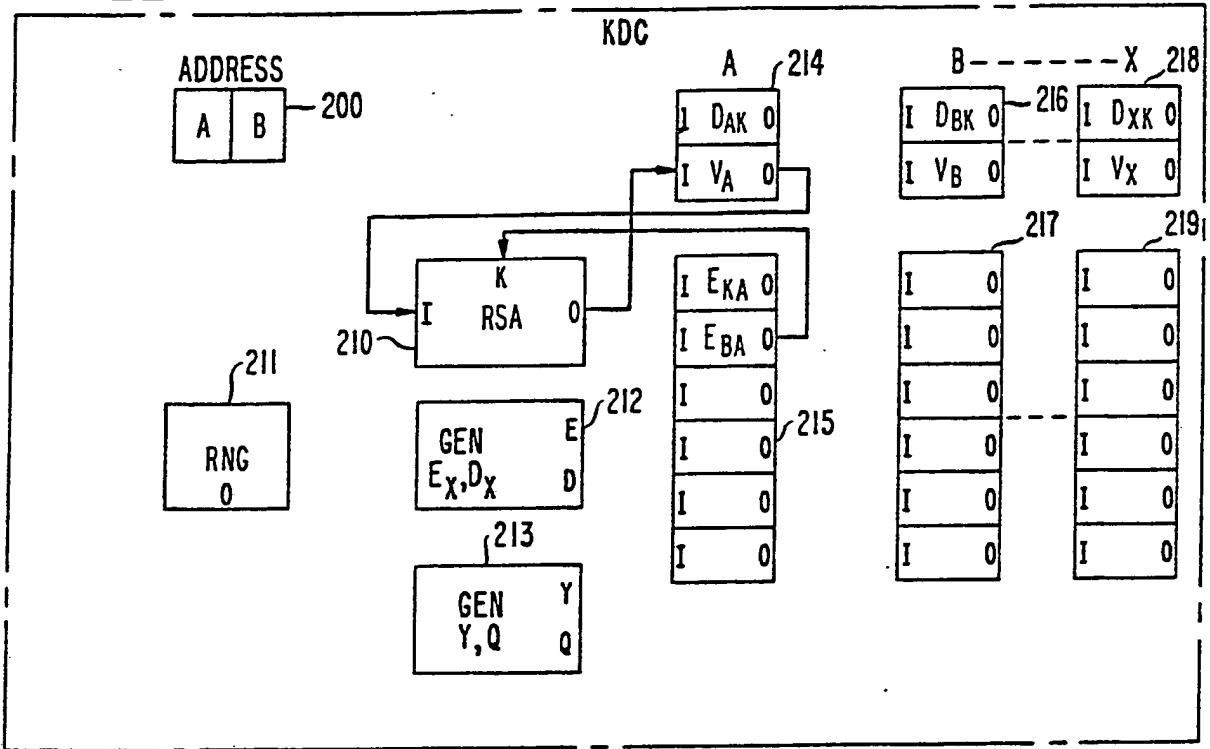


FIG. 23 SECOND STEP IN THE UPDATE OF VERIFICATION INFORMATION

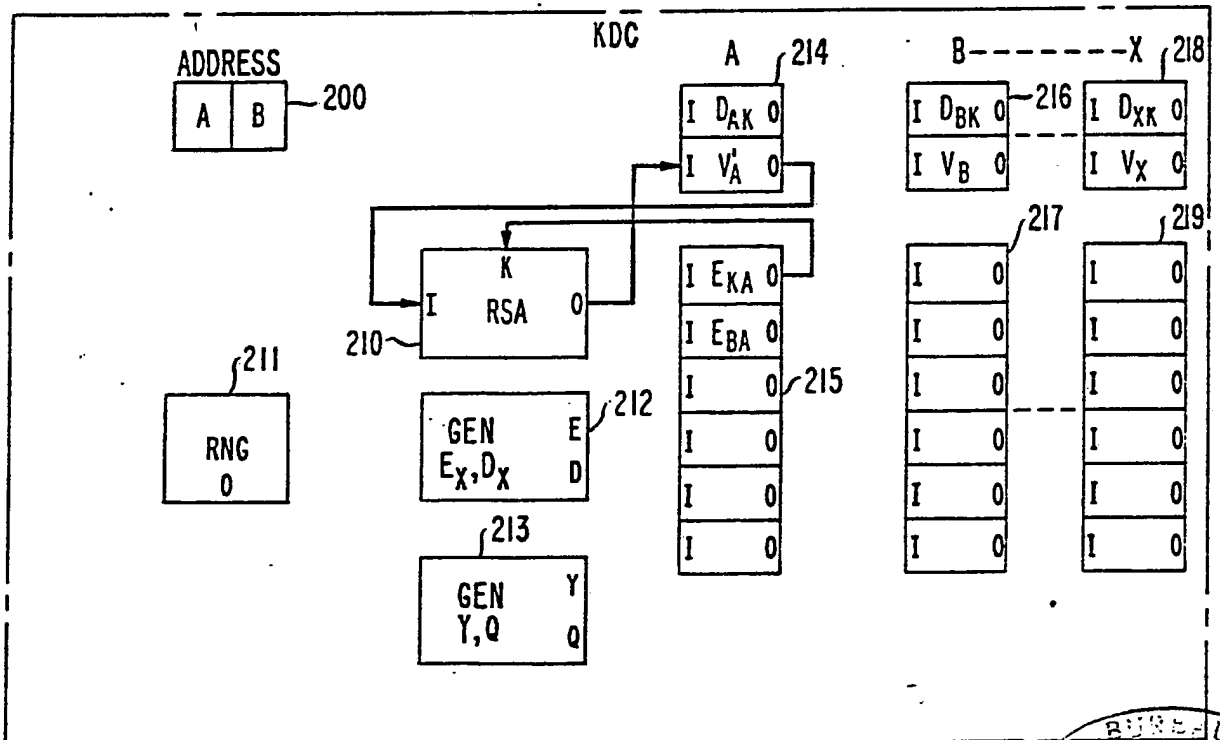


FIG. 24 GENERATION OF NEW KDC-A LINK KEYS

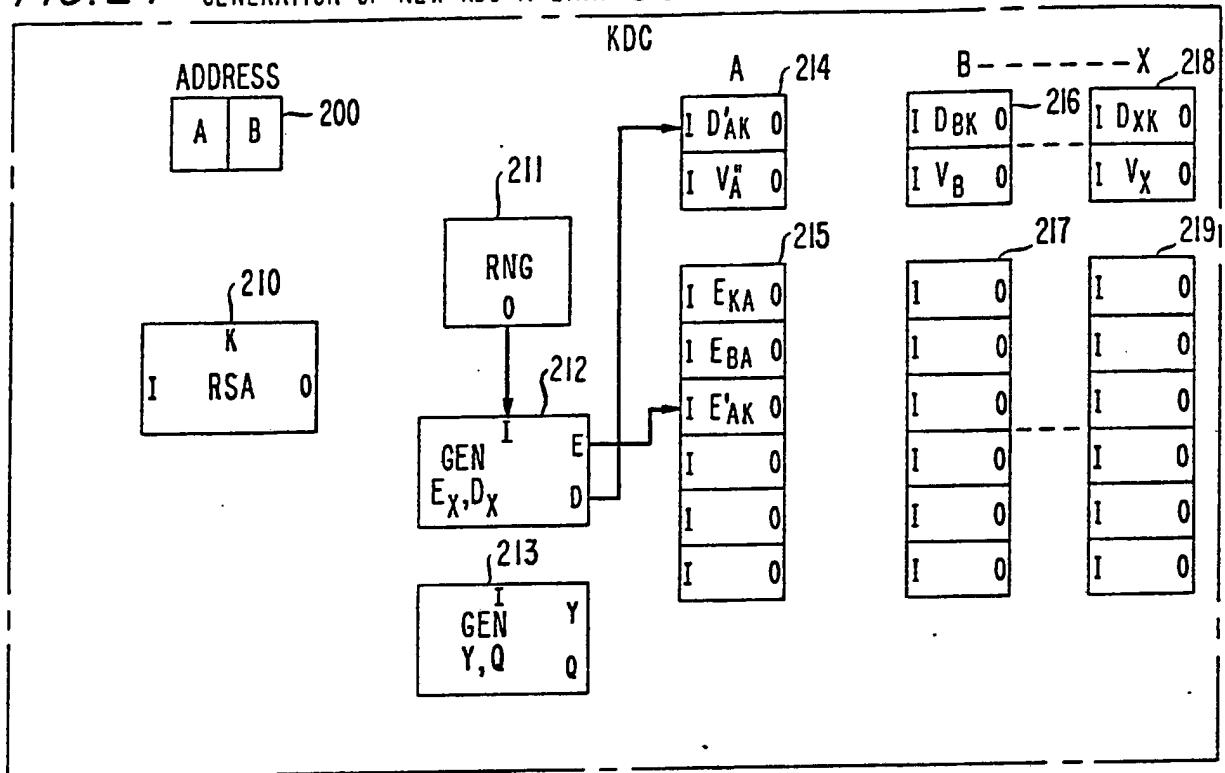


FIG. 25 GENERATION OF DIFFIE-HELLMAN ALGORITHM PARAMETERS

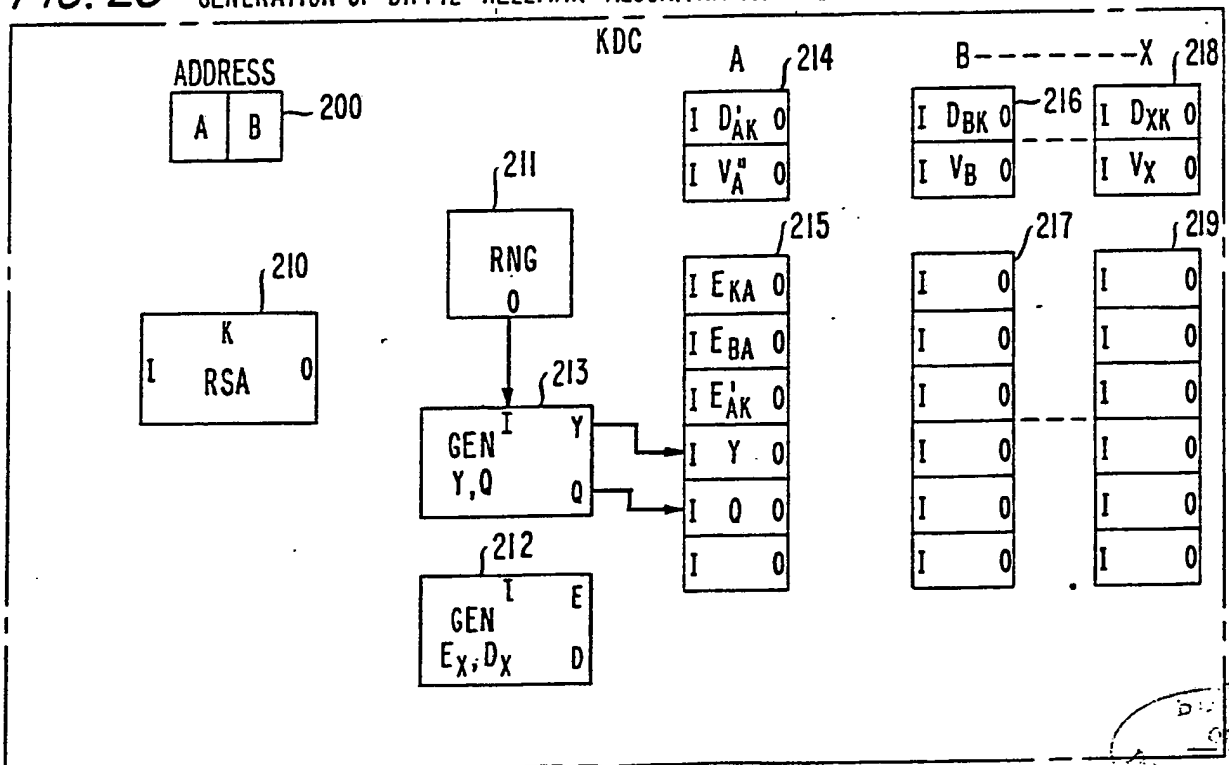


FIG. 26 INTERNAL EXCHANGE OF INFORMATION BETWEEN A & B'S REGISTERS

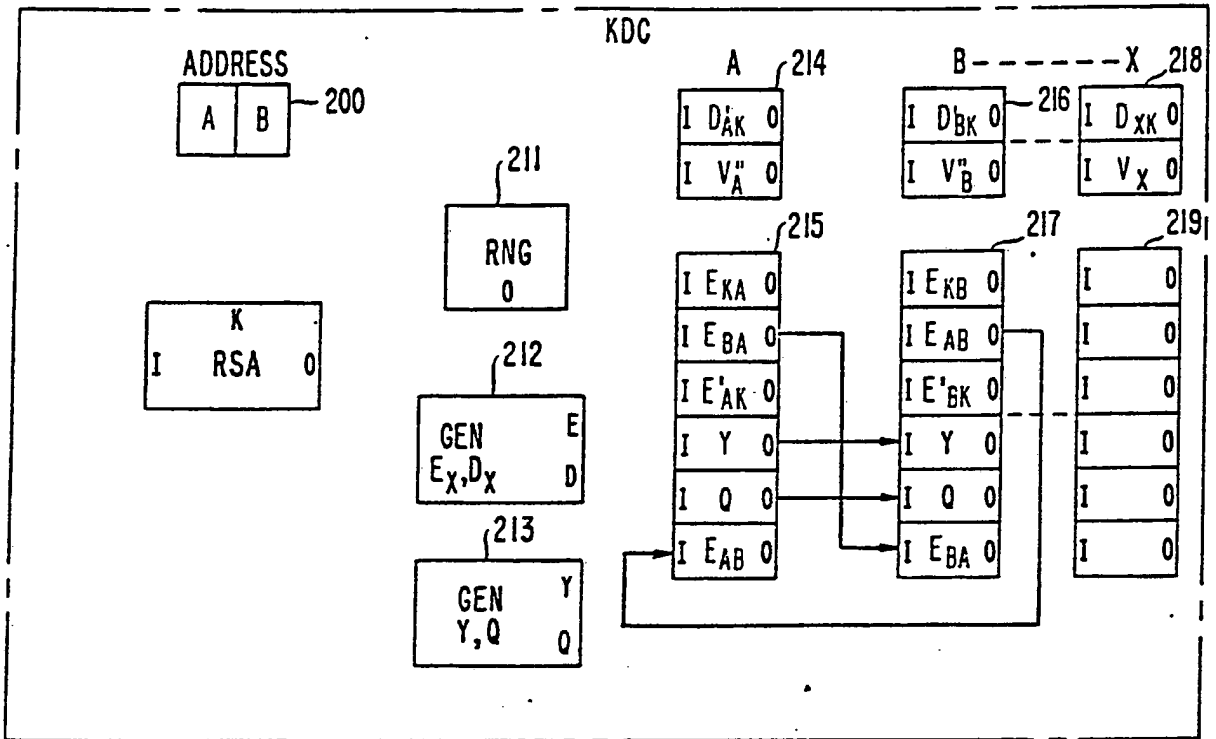


FIG. 27 COMPUTATION OF MESSAGE TO A

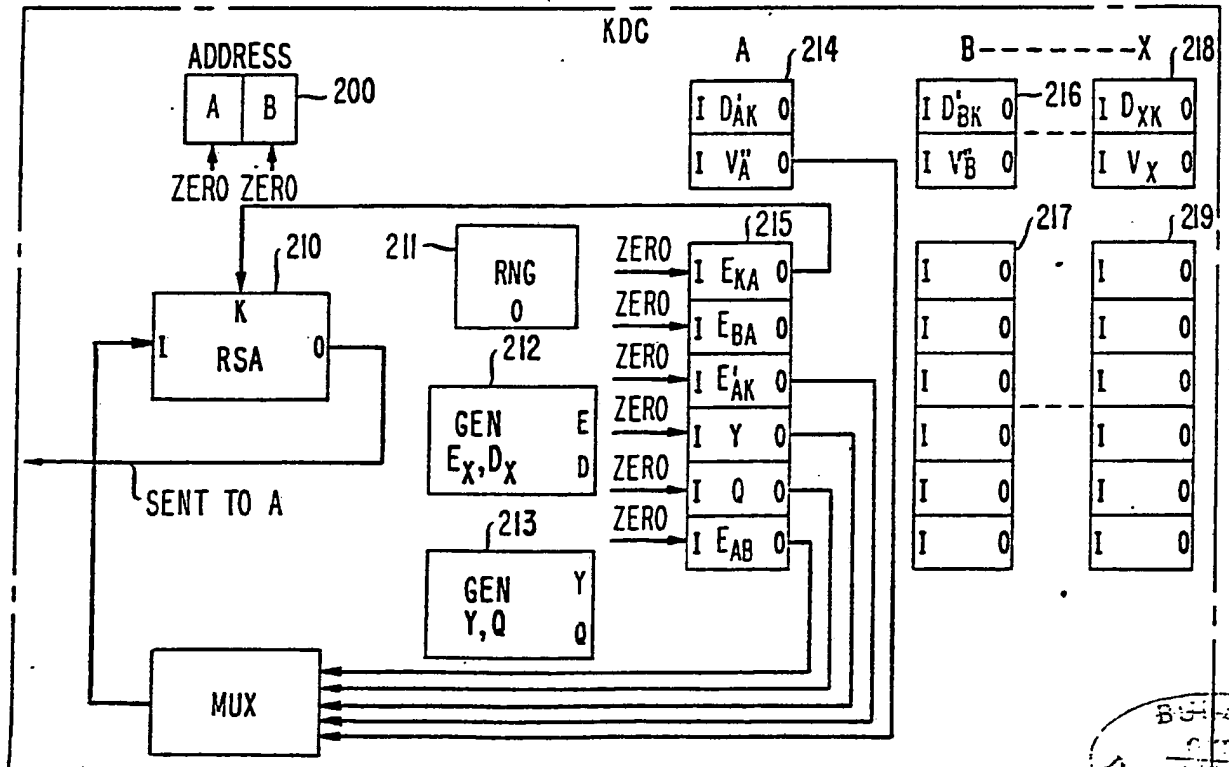
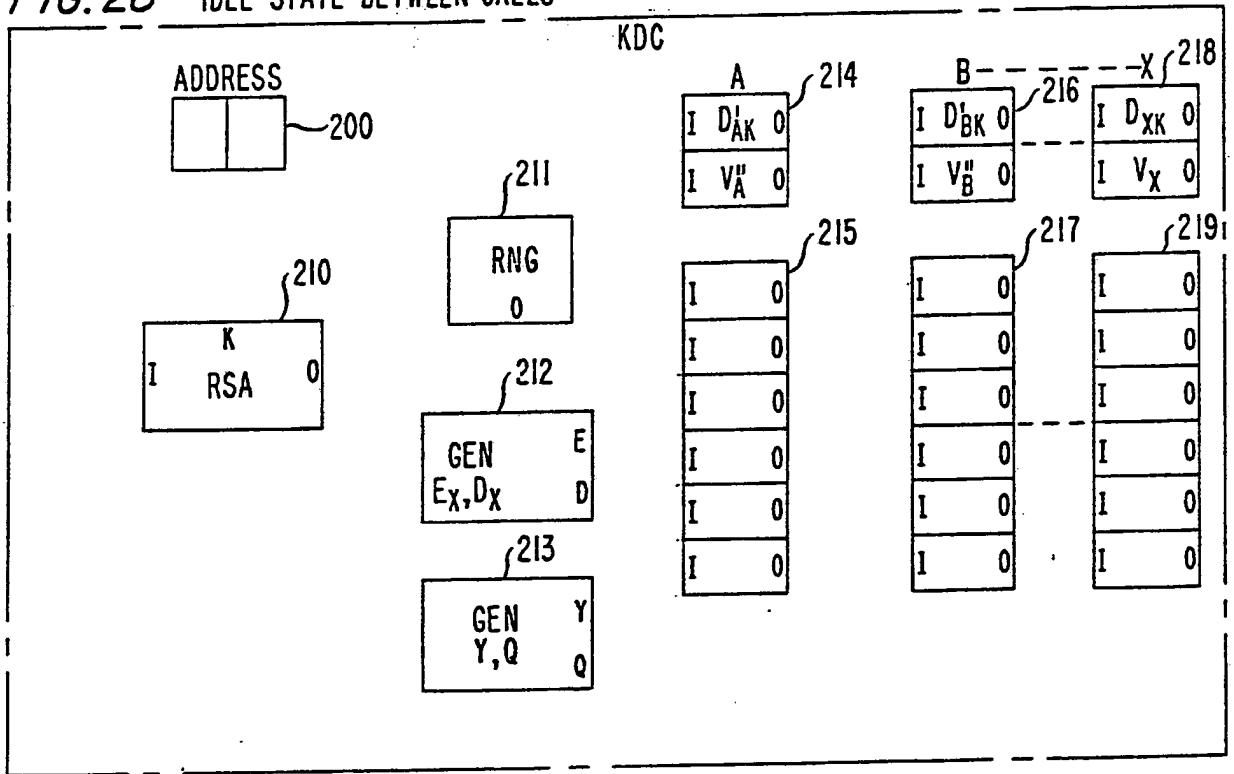
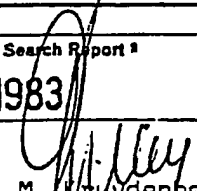


FIG. 28 IDLE STATE BETWEEN CALLS



INTERNATIONAL SEARCH REPORT

International Application No **PCT/US 83/00030**

I. CLASSIFICATION OF SUBJECT MATTER (If several classification symbols apply, indicate all) ²				
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 ⁶	Citation of Document, ¹⁵ 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.:	./.		
<table style="width: 100%; border: none;"> <tr> <td style="width: 50%; border: none; vertical-align: top;"> <p>¹⁶ Special categories of cited documents:</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier document but published on or after the international filing date</p> <p>"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)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p> </td> <td style="width: 50%; border: none; vertical-align: top;"> <p>"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</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step</p> <p>"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.</p> <p>"&" document member of the same patent family</p> </td> </tr> </table>			<p>¹⁶ Special categories of cited documents:</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier document but published on or after the international filing date</p> <p>"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)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p>	<p>"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</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step</p> <p>"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.</p> <p>"&" document member of the same patent family</p>
<p>¹⁶ Special categories of cited documents:</p> <p>"A" document defining the general state of the art which is not considered to be of particular relevance</p> <p>"E" earlier document but published on or after the international filing date</p> <p>"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)</p> <p>"O" document referring to an oral disclosure, use, exhibition or other means</p> <p>"P" document published prior to the international filing date but later than the priority date claimed</p>	<p>"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</p> <p>"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step</p> <p>"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.</p> <p>"&" document member of the same patent family</p>			
IV. CERTIFICATION				
Date of the Actual Completion of the International Search ¹⁹	Date of Mailing of this International Search Report ²⁰			
20th April 1983	03 MAI 1983			
International Searching Authority ¹	Signature of Authorized Officer ²⁰			
EUROPEAN PATENT OFFICE	 G.L.M. Kruidenberg			

**CORRECTED
VERSION***

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau



INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F	A2	(11) International Publication Number: WO 96/27155 (43) International Publication Date: 6 September 1996 (06.09.96)
(21) International Application Number: PCT/US96/02303 (22) International Filing Date: 13 February 1996 (13.02.96) (30) Priority Data: 08/388,107 13 February 1995 (13.02.95) US (71) Applicant: ELECTRONIC PUBLISHING RESOURCES, INC. [US/US]; 5203 Battery Lane, Bethesda, MD 20814 (US). (72) Inventors: GINTER, Karl, L.; 10404 43rd Avenue, Beltsville, MD 20705 (US). SHEAR, Victor, H.; 5203 Battery Lane, Bethesda, MD 20814 (US). SPAHN, Francis, J.; 2410 Edwards Avenue, El Cerrito, CA 94530 (US). VAN WIE, David, M.; 1250 Lakeside Drive, Sunnyvale, CA 94086 (US). (74) Agent: FARIS, Robert, W.; Nixon & Vanderhye P.C., 1100 North Glebe Road, Arlington, VA 22201-4714 (US).		(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IS, JP, KE, KG, KP, KR, KZ, LK, LR, 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 (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 <i>Without international search report and to be republished upon receipt of that report.</i>
(54) Title: SYSTEMS AND METHODS FOR SECURE TRANSACTION MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION		
(57) Abstract <p>The present invention provides systems and methods for electronic commerce including secure transaction management and electronic rights protection. Electronic appliances such as computers employed in accordance with the present invention help to ensure that information is accessed and used only in authorized ways, and maintain the integrity, availability, and/or confidentiality of the information. Secure subsystems used with such electronic appliances provide a distributed virtual distribution environment (VDE) that may enforce a secure chain of handling and control, for example, to control and/or meter or otherwise monitor use of electronically stored or disseminated information. Such a virtual distribution environment may be used to protect rights of various participants in electronic commerce and other electronic or electronic-facilitated transactions. Secure distributed and other operating system environments and architectures, employing, for example, secure semiconductor processing arrangements that may establish secure, protected environments at each node. These techniques may be used to support an end-to-end electronic information distribution capability that may be used, for example, utilizing the "electronic highway".</p>		

* (Referred to in PCT Gazette No. 52/1996, Section II)

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	Kyrgyzstan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic	KR	Republic of Korea	SE	Sweden
CG	Congo	KZ	Kazakhstan	SG	Singapore
CH	Switzerland	LI	Liechtenstein	SI	Slovenia
CI	Côte d'Ivoire	LK	Sri Lanka	SK	Slovakia
CM	Cameroon	LR	Liberia	SN	Senegal
CN	China	LT	Lithuania	SZ	Swaziland
CS	Czechoslovakia	LU	Luxembourg	TD	Chad
CZ	Czech Republic	LV	Latvia	TG	Togo
DE	Germany	MC	Monaco	TJ	Tajikistan
DK	Denmark	MD	Republic of Moldova	TT	Trinidad and Tobago
EE	Estonia	MG	Madagascar	UA	Ukraine
ES	Spain	ML	Mali	UG	Uganda
FI	Finland	MN	Mongolia	US	United States of America
FR	France	MR	Mauritania	UZ	Uzbekistan
GA	Gabon			VN	Viet Nam

**SYSTEMS AND METHODS FOR SECURE TRANSACTION
MANAGEMENT AND ELECTRONIC RIGHTS PROTECTION**

Field(s) of the Invention(s)

This invention generally relates to computer and/or
electronic security.

5

More particularly, this invention relates to systems and
techniques for secure transaction management. This invention
also relates to computer-based and other electronic appliance-
based technologies that help to ensure that information is
10 accessed and/or otherwise used only in authorized ways, and
maintains the integrity, availability, and/or confidentiality of
such information and processes related to such use.

10

The invention also relates to systems and methods for
15 protecting rights of various participants in electronic commerce
and other electronic or electronically-facilitated transactions.

15

The invention also relates to secure chains of handling and
control for both information content and information employed to
20 regulate the use of such content and consequences of such use. It
also relates to systems and techniques that manage, including
meter and/or limit and/or otherwise monitor use of electronically
stored and/or disseminated information. The invention

20

particularly relates to transactions, conduct and arrangements that make use of, including consequences of use of, such systems and/or techniques.

5 The invention also relates to distributed and other operating systems, environments and architectures. It also generally relates to secure architectures, including, for example, tamper-resistant hardware-based processors, that can be used to establish security at each node of a distributed system.

10

Background and Summary of the Invention(s)

Telecommunications, financial transactions, government processes, business operations, entertainment, and personal business productivity all now depend on electronic appliances. Millions of these electronic appliances have been electronically connected together. These interconnected electronic appliances comprise what is increasingly called the "information highway." Many businesses, academicians, and government leaders are concerned about how to protect the rights of citizens and organizations who use this information (also "electronic" or

15

20

25

"digital") highway.

Electronic Content

Today, virtually anything that can be represented by words, numbers, graphics, or system of commands and

instructions can be formatted into electronic digital information.

Television, cable, satellite transmissions, and on-line services transmitted over telephone lines, compete to distribute digital information and entertainment to homes and businesses. The owners and marketers of this content include software developers, motion picture and recording companies, publishers of books, magazines, and newspapers, and information database providers. The popularization of on-line services has also enabled the individual personal computer user to participate as a content provider. It is estimated that the worldwide market for electronic information in 1992 was approximately \$40 billion and is expected to grow to \$200 billion by 1997, according to Microsoft Corporation. The present invention can materially enhance the revenue of content providers, lower the distribution costs and the costs for content, better support advertising and usage information gathering, and better satisfy the needs of electronic information users. These improvements can lead to a significant increase in the amount and variety of electronic information and the methods by which such information is distributed.

The inability of conventional products to be shaped to the needs of electronic information providers and users is sharply in contrast to the present invention. Despite the attention devoted by a cross-section of America's largest telecommunications, computer, entertainment and information provider companies to

some of the problems addressed by the present invention, only the present invention provides commercially secure, effective solutions for configurable, general purpose electronic commerce transaction/distribution control systems.

5

Controlling Electronic Content

The present invention provides a new kind of "virtual distribution environment" (called "VDE" in this document) that secures, administers, and audits electronic information use. VDE also features fundamentally important capabilities for managing content that travels "across" the "information highway." These capabilities comprise a rights protection solution that serves all electronic community members. These members include content creators and distributors, financial service providers, end-users, and others. VDE is the first general purpose, configurable, transaction control/rights protection solution for users of computers, other electronic appliances, networks, and the information highway.

10
15
20

A fundamental problem for electronic content providers is extending their ability to control the use of proprietary information. Content providers often need to limit use to authorized activities and amounts. Participants in a business model involving, for example, provision of movies and advertising on optical discs may include actors, directors, script and other

25

writers, musicians, studios, publishers, distributors, retailers, advertisers, credit card services, and content end-users. These participants need the ability to embody their range of agreements and requirements, including use limitations, into an "extended" agreement comprising an overall electronic business model. This extended agreement is represented by electronic content control information that can automatically enforce agreed upon rights and obligations. Under VDE, such an extended agreement may comprise an electronic contract involving all business model participants. Such an agreement may alternatively, or in addition, be made up of electronic agreements between subsets of the business model participants. Through the use of VDE, electronic commerce can function in the same way as traditional commerce—that is commercial relationships regarding products and services can be shaped through the negotiation of one or more agreements between a variety of parties.

Commercial content providers are concerned with ensuring proper compensation for the use of their electronic information. Electronic digital information, for example a CD recording, can today be copied relatively easily and inexpensively. Similarly, unauthorized copying and use of software programs deprives rightful owners of billions of dollars in annual revenue according to the International Intellectual Property Alliance. Content providers and distributors have devised a number of limited

function rights protection mechanisms to protect their rights. Authorization passwords and protocols, license servers, “lock/unlock” distribution methods, and non-electronic contractual limitations imposed on users of shrink-wrapped software are a few of the more prevalent content protection schemes. In a commercial context, these efforts are inefficient and limited solutions.

Providers of “electronic currency” have also created protections for their type of content. These systems are not sufficiently adaptable, efficient, nor flexible enough to support the generalized use of electronic currency. Furthermore, they do not provide sophisticated auditing and control configuration capabilities. This means that current electronic currency tools lack the sophistication needed for many real-world financial business models. VDE provides means for anonymous currency and for “conditionally” anonymous currency, wherein currency related activities remain anonymous except under special circumstances.

20

VDE Control Capabilities

VDE allows the owners and distributors of electronic digital information to reliably bill for, and securely control, audit, and budget the use of, electronic information. It can reliably

25

detect and monitor the use of commercial information products.

VDE uses a wide variety of different electronic information delivery means: including, for example, digital networks, digital broadcast, and physical storage media such as optical and magnetic disks. VDE can be used by major network providers, hardware manufacturers, owners of electronic information, providers of such information, and clearinghouses that gather usage information regarding, and bill for the use of, electronic information.

VDE provides comprehensive and configurable transaction management, metering and monitoring technology. It can change how electronic information products are protected, marketed, packaged, and distributed. When used, VDE should result in higher revenues for information providers and greater user satisfaction and value. Use of VDE will normally result in lower usage costs, decreased transaction costs, more efficient access to electronic information, re-usability of rights protection and other transaction management implementations, greatly improved flexibility in the use of secured information, and greater standardization of tools and processes for electronic transaction management. VDE can be used to create an adaptable environment that fulfills the needs of electronic information owners, distributors, and users; financial clearinghouses; and usage information analyzers and resellers.

Rights and Control Information

In general, the present invention can be used to protect the rights of parties who have:

- 5 (a) proprietary or confidentiality interests in electronic information. It can, for example, help ensure that information is used only in authorized ways;
- 10 (b) financial interests resulting from the use of electronically distributed information. It can help ensure that content providers will be paid for use of distributed information; and
- 15 (c) interests in electronic credit and electronic currency storage, communication, and/or use including electronic cash, banking, and purchasing.

20 Protecting the rights of electronic community members involves a broad range of technologies. VDE combines these technologies in a way that creates a "distributed" electronic rights protection "environment." This environment secures and protects transactions and other processes important for rights protection. VDE, for example, provides the ability to prevent, or impede, interference with and/or observation of, important rights

25 related transactions and processes. VDE, in its preferred

embodiment, uses special purpose tamper resistant Secure Processing Units (SPUs) to help provide a high level of security for VDE processes and information storage and communication.

5 The rights protection problems solved by the present invention are electronic versions of basic societal issues. These issues include protecting property rights, protecting privacy rights, properly compensating people and organizations for their work and risk, protecting money and credit, and generally
10 protecting the security of information. VDE employs a system that uses a common set of processes to manage rights issues in an efficient, trusted, and cost-effective way.

 VDE can be used to protect the rights of parties who create
15 electronic content such as, for example: records, games, movies, newspapers, electronic books and reference materials, personal electronic mail, and confidential records and communications. The invention can also be used to protect the rights of parties who provide electronic products, such as publishers and
20 distributors; the rights of parties who provide electronic credit and currency to pay for use of products, for example, credit clearinghouses and banks; the rights to privacy of parties who use electronic content (such as consumers, business people, governments); and the privacy rights of parties described by
25 electronic information, such as privacy rights related to

information contained in a medical record, tax record, or
personnel record.

5 In general, the present invention can protect the rights of
parties who have:

- 10 (a) commercial interests in electronically distributed
information -- the present invention can help ensure,
for example, that parties, will be paid for use of
distributed information in a manner consistent with
their agreement;
- 15 (b) proprietary and/or confidentiality interests in
electronic information -- the present invention can,
for example, help ensure that data is used only in
authorized ways;
- 20 (c) interests in electronic credit and electronic currency
storage, communication, and/or use -- this can
include electronic cash, banking, and purchasing;
and
- (d) interests in electronic information derived, at least
in part, from use of other electronic information.

25 **VDE Functional Properties**

VDE is a cost-effective and efficient rights protection solution that provides a unified, consistent system for securing and managing transaction processing. VDE can:

- 5 (a) audit and analyze the use of content,
- (b) ensure that content is used only in authorized ways,
 and
- 10 (c) allow information regarding content usage to be used
 only in ways approved by content users.

In addition, VDE:

- 15 (a) is very configurable, modifiable, and re-usable;
- (b) supports a wide range of useful capabilities that may
 be combined in different ways to accommodate most
 potential applications;
- 20 (c) operates on a wide variety of electronic appliances
 ranging from hand-held inexpensive devices to large
 mainframe computers;

- (d) is able to ensure the various rights of a number of different parties, and a number of different rights protection schemes, simultaneously;
- 5 (e) is able to preserve the rights of parties through a series of transactions that may occur at different times and different locations;
- 10 (f) is able to flexibly accommodate different ways of securely delivering information and reporting usage; and
- 15 (g) provides for electronic analogues to "real" money and credit, including anonymous electronic cash, to pay for products and services and to support personal (including home) banking and other financial activities.

VDE economically and efficiently fulfills the rights protection needs of electronic community members. Users of VDE will not require additional rights protection systems for different information highway products and rights problems—nor will they be required to install and learn a new system for each new information highway application.

25

VDE provides a unified solution that allows all content creators, providers, and users to employ the same electronic rights protection solution. Under authorized circumstances, the participants can freely exchange content and associated content control sets. This means that a user of VDE may, if allowed, use the same electronic system to work with different kinds of content having different sets of content control information. The content and control information supplied by one group can be used by people who normally use content and control information supplied by a different group. VDE can allow content to be exchanged "universally" and users of an implementation of the present invention can interact electronically without fear of incompatibilities in content control, violation of rights, or the need to get, install, or learn a new content control system.

15

The VDE securely administers transactions that specify protection of rights. It can protect electronic rights including, for example:

20

(a) the property rights of authors of electronic content,

(b) the commercial rights of distributors of content,

(c) the rights of any parties who facilitated the distribution of content,

25

- (d) the privacy rights of users of content,
- (e) the privacy rights of parties portrayed by stored
and/or distributed content, and
- 5
- (f) any other rights regarding enforcement of electronic
agreements.

VDE can enable a very broad variety of electronically enforced
10 commercial and societal agreements. These agreements can
include electronically implemented contracts, licenses, laws,
regulations, and tax collection.

Contrast With Traditional Solutions

15 Traditional content control mechanisms often require users
to purchase more electronic information than the user needs or
desires. For example, infrequent users of shrink-wrapped
software are required to purchase a program at the same price as
frequent users, even though they may receive much less value
20 from their less frequent use. Traditional systems do not scale
cost according to the extent or character of usage and traditional
systems can not attract potential customers who find that a fixed
price is too high. Systems using traditional mechanisms are also
not normally particularly secure. For example, shrink-wrapping

does not prevent the constant illegal pirating of software once removed from either its physical or electronic package.

Traditional electronic information rights protection
5 systems are often inflexible and inefficient and may cause a content provider to choose costly distribution channels that increase a product's price. In general these mechanisms restrict product pricing, configuration, and marketing flexibility. These compromises are the result of techniques for controlling
10 information which cannot accommodate both different content models and content models which reflect the many, varied requirements, such as content delivery strategies, of the model participants. This can limit a provider's ability to deliver sufficient overall value to justify a given product's cost in the eyes
15 of many potential users. VDE allows content providers and distributors to create applications and distribution networks that reflect content providers' and users' preferred business models. It offers users a uniquely cost effective and feature rich system that supports the ways providers want to distribute information
20 and the ways users want to use such information. VDE supports content control models that ensure rights and allow content delivery strategies to be shaped for maximum commercial results.

Chain of Handling and Control

VDE can protect a collection of rights belonging to various parties having in rights in, or to, electronic information. This information may be at one location or dispersed across (and/or moving between) multiple locations. The information may pass through a "chain" of distributors and a "chain" of users. Usage information may also be reported through one or more "chains" of parties. In general, VDE enables parties that (a) have rights in electronic information, and/or (b) act as direct or indirect agents for parties who have rights in electronic information, to ensure that the moving, accessing, modifying, or otherwise using of information can be securely controlled by rules regarding how, when, where, and by whom such activities can be performed.

VDE Applications and Software

VDE is a secure system for regulating electronic conduct and commerce. Regulation is ensured by control information put in place by one or more parties. These parties may include content providers, electronic hardware manufacturers, financial service providers, or electronic "infrastructure" companies such as cable or telecommunications companies. The control information implements "Rights Applications." Rights applications "run on" the "base software" of the preferred embodiment. This base software serves as a secure, flexible, general purpose foundation that can accommodate many

different rights applications, that is, many different business models and their respective participant requirements.

5 A rights application under VDE is made up of special purpose pieces, each of which can correspond to one or more basic electronic processes needed for a rights protection environment. These processes can be combined together like building blocks to create electronic agreements that can protect the rights, and may enforce fulfillment of the obligations, of electronic information users and providers. One or more providers of electronic information can easily combine selected building blocks to create a rights application that is unique to a specific content distribution model. A group of these pieces can represent the capabilities needed to fulfill the agreement(s) between users and providers. These pieces accommodate many requirements of electronic commerce including:

10

15

! the distribution of permissions to use electronic information;

20

! the persistence of the control information and sets of control information managing these permissions;

25

! configurable control set information that can be selected by users for use with such information;

- ! data security and usage auditing of electronic information; and
 - ! a secure system for currency, compensation and debit management.
- 5

For electronic commerce, a rights application, under the preferred embodiment of the present invention, can provide electronic enforcement of the business agreements between all participants. Since different groups of components can be put together for different applications, the present invention can provide electronic control information for a wide variety of different products and markets. This means the present invention can provide a "unified," efficient, secure, and cost-effective system for electronic commerce and data security. This allows VDE to serve as a single standard for electronic rights protection, data security, and electronic currency and banking.

10

15

20 In a VDE, the separation between a rights application and its foundation permits the efficient selection of sets of control information that are appropriate for each of many different types of applications and uses. These control sets can reflect both rights of electronic community members, as well as obligations (such as providing a history of one's use of a product or paying

25

taxes on one's electronic purchases). VDE flexibility allows its users to electronically implement and enforce common social and commercial ethics and practices. By providing a unified control system, the present invention supports a vast range of possible transaction related interests and concerns of individuals, communities, businesses, and governments. Due to its open design, VDE allows (normally under securely controlled circumstances) applications using technology independently created by users to be "added" to the system and used in conjunction with the foundation of the invention. In sum, VDE provides a system that can fairly reflect and enforce agreements among parties. It is a broad ranging and systematic solution that answers the pressing need for a secure, cost-effective, and fair electronic environment.

15

VDE Implementation

The preferred embodiment of the present invention includes various tools that enable system designers to directly insert VDE capabilities into their products. These tools include an Application Programmer's Interface ("API") and a Rights Permissioning and Management Language ("RPML"). The RPML provides comprehensive and detailed control over the use of the invention's features. VDE also includes certain user interface subsystems for satisfying the needs of content providers, distributors, and users.

25

Information distributed using VDE may take many forms. It may, for example, be "distributed" for use on an individual's own computer, that is the present invention can be used to provide security for locally stored data. Alternatively, VDE may
5 be used with information that is dispersed by authors and/or publishers to one or more recipients. This information may take many forms including: movies, audio recordings, games, electronic catalog shopping, multimedia, training materials, E-mail and personal documents, object oriented libraries,
10 software programming resources, and reference/record keeping information resources (such as business, medical, legal, scientific, governmental, and consumer databases).

Electronic rights protection provided by the present
15 invention will also provide an important foundation for trusted and efficient home and commercial banking, electronic credit processes, electronic purchasing, true or conditionally anonymous electronic cash, and EDI (Electronic Data Interchange). VDE provides important enhancements for improving data security in
20 organizations by providing "smart" transaction management features that can be far more effective than key and password based "go/no go" technology.

VDE normally employs an integration of cryptographic and
25 other security technologies (e.g. encryption, digital signatures,

etc.), with other technologies including: component, distributed, and event driven operating system technology, and related communications, object container, database, smart agent, smart card, and semiconductor design technologies.

5

I. Overview

A. VDE Solves Important Problems and Fills Critical Needs

The world is moving towards an integration of electronic information appliances. This interconnection of appliances provides a foundation for much greater electronic interaction and the evolution of electronic commerce. A variety of capabilities are required to implement an electronic commerce environment. VDE is the first system that provides many of these capabilities and therefore solves fundamental problems related to electronic dissemination of information.

10
15

Electronic Content

VDE allows electronic arrangements to be created involving two or more parties. These agreements can themselves comprise a collection of agreements between participants in a commercial value chain and/or a data security chain model for handling, auditing, reporting, and payment. It can provide efficient, reusable, modifiable, and consistent means for secure electronic content: distribution, usage control, usage payment,

20
25

usage auditing, and usage reporting. Content may, for example, include:

5 ! financial information such as electronic currency and credit;

 ! commercially distributed electronic information such as reference databases, movies, games, and advertising; and

10 ! electronic properties produced by persons and organizations, such as documents, e-mail, and proprietary database information.

15 VDE enables an electronic commerce marketplace that supports differing, competitive business partnerships, agreements, and evolving overall business models.

 The features of VDE allow it to function as the first trusted
20 electronic information control environment that can conform to, and support, the bulk of conventional electronic commerce and data security requirements. In particular, VDE enables the participants in a business value chain model to create an
 electronic version of traditional business agreement terms and
25 conditions and further enables these participants to shape and

evolve their electronic commerce models as they believe appropriate to their business requirements.

5 VDE offers an architecture that avoids reflecting specific distribution biases, administrative and control perspectives, and content types. Instead, VDE provides a broad-spectrum, fundamentally configurable and portable, electronic transaction control, distributing, usage, auditing, reporting, and payment operating environment. VDE is not limited to being an application or application specific toolset that covers only a limited subset of electronic interaction activities and participants. Rather, VDE supports systems by which such applications can be created, modified, and/or reused. As a result, the present invention answers pressing, unsolved needs by offering a system that supports a standardized control environment which facilitates interoperability of electronic appliances, interoperability of content containers, and efficient creation of electronic commerce applications and models through the use of a programmable, secure electronic transactions management foundation and reusable and extensible executable components. VDE can support a single electronic "world" within which most forms of electronic transaction activities can be managed.

10
15
20

To answer the developing needs of rights owners and content providers and to provide a system that can accommodate

25

the requirements and agreements of all parties that may be involved in electronic business models (creators, distributors, administrators, users, credit providers, etc.), VDE supplies an efficient, largely transparent, low cost and sufficiently secure system (supporting both hardware/ software and software only models). VDE provides the widely varying secure control and administration capabilities required for:

1. Different types of electronic content,
2. Differing electronic content delivery schemes,
3. Differing electronic content usage schemes,
4. Different content usage platforms, and
5. Differing content marketing and model strategies.

VDE may be combined with, or integrated into, many separate computers and/or other electronic appliances. These appliances typically include a secure subsystem that can enable control of content use such as displaying, encrypting, decrypting, printing, copying, saving, extracting, embedding, distributing, auditing usage, etc. The secure subsystem in the preferred embodiment comprises one or more "protected processing

environments", one or more secure databases, and secure
"component assemblies" and other items and processes that need
to be kept secured. VDE can, for example, securely control
electronic currency, payments, and/or credit management
5 (including electronic credit and/or currency receipt,
disbursement, encumbering, and/or allocation) using such a
"secure subsystem."

VDE provides a secure, distributed electronic transaction
10 management system for controlling the distribution and/or other
usage of electronically provided and/or stored information. VDE
controls auditing and reporting of electronic content and/or
appliance usage. Users of VDE may include content creators who
apply content usage, usage reporting, and/or usage payment
15 related control information to electronic content and/or
appliances for users such as end-user organizations, individuals,
and content and/or appliance distributors. VDE also securely
supports the payment of money owed (including money owed for
content and/or appliance usage) by one or more parties to one or
20 more other parties, in the form of electronic credit and/or
currency.

Electronic appliances under control of VDE represent VDE
'nodes' that securely process and control; distributed electronic
25 information and/or appliance usage, control information

formulation, and related transactions. VDE can securely manage the integration of control information provided by two or more parties. As a result, VDE can construct an electronic agreement between VDE participants that represent a “negotiation”
5 between, the control requirements of, two or more parties and enacts terms and conditions of a resulting agreement. VDE ensures the rights of each party to an electronic agreement regarding a wide range of electronic activities related to electronic information and/or appliance usage.

10

Through use of VDE’s control system, traditional content providers and users can create electronic relationships that reflect traditional, non-electronic relationships. They can shape and modify commercial relationships to accommodate the
15 evolving needs of, and agreements among, themselves. VDE does not require electronic content providers and users to modify their business practices and personal preferences to conform to a metering and control application program that supports limited, largely fixed functionality. Furthermore, VDE permits
20 participants to develop business models not feasible with non-electronic commerce, for example, involving detailed reporting of content usage information, large numbers of distinct transactions at hitherto infeasibly low price points, “pass-along” control information that is enforced without involvement or advance
25 knowledge of the participants, etc.

The present invention allows content providers and users
to formulate their transaction environment to accommodate:

- 5 (1) desired content models, content control models, and
content usage information pathways,
- (2) a complete range of electronic media and distribution
means,
- 10 (3) a broad range of pricing, payment, and auditing
strategies,
- (4) very flexible privacy and/or reporting models,
- 15 (5) practical and effective security architectures, and
- (6) other administrative procedures that together with
steps (1) through (5) can enable most "real world"
electronic commerce and data security models,
20 including models unique to the electronic world.

VDE's transaction management capabilities can enforce:

- (1) privacy rights of users related to information regarding their usage of electronic information and/or appliances,
- 5 (2) societal policy such as laws that protect rights of content users or require the collection of taxes derived from electronic transaction revenue, and
- 10 (3) the proprietary and/or other rights of parties related to ownership of, distribution of, and/or other commercial rights related to, electronic information.

VDE can support "real" commerce in an electronic form, that is the progressive creation of commercial relationships that form, over time, a network of interrelated agreements representing a value chain business model. This is achieved in part by enabling content control information to develop through the interaction of (negotiation between) securely created and independently submitted sets of content and/or appliance control information. Different sets of content and/or appliance control information can be submitted by different parties in an electronic business value chain enabled by the present invention. These parties create control information sets through the use of their respective VDE installations. Independently, securely deliverable, component based control information allows efficient

15
20
25

interaction among control information sets supplied by different parties.

5 VDE permits multiple, separate electronic arrangements to be formed between subsets of parties in a VDE supported electronic value chain model. These multiple agreements together comprise a VDE value chain "extended" agreement. VDE allows such constituent electronic agreements, and therefore overall VDE extended agreements, to evolve and
10 reshape over time as additional VDE participants become involved in VDE content and/or appliance control information handling. VDE electronic agreements may also be extended as new control information is submitted by existing participants. With VDE, electronic commerce participants are free to structure
15 and restructure their electronic commerce business activities and relationships. As a result, the present invention allows a competitive electronic commerce marketplace to develop since the use of VDE enables different, widely varying business models using the same or shared content.

20 A significant facet of the present invention's ability to broadly support electronic commerce is its ability to securely manage independently delivered VDE component objects containing control information (normally in the form of VDE
25 objects containing one or more methods, data, or load module

VDE components). This independently delivered control information can be integrated with senior and other pre-existing content control information to securely form derived control information using the negotiation mechanisms of the present invention. All requirements specified by this derived control information must be satisfied before VDE controlled content can be accessed or otherwise used. This means that, for example, all load modules and any mediating data which are listed by the derived control information as required must be available and securely perform their required function. In combination with other aspects of the present invention, securely, independently delivered control components allow electronic commerce participants to freely stipulate their business requirements and trade offs. As a result, much as with traditional, non-electronic commerce, the present invention allows electronic commerce (through a progressive stipulation of various control requirements by VDE participants) to evolve into forms of business that are the most efficient, competitive and useful.

VDE provides capabilities that rationalize the support of electronic commerce and electronic transaction management. This rationalization stems from the reusability of control structures and user interfaces for a wide variety of transaction management related activities. As a result, content usage control, data security, information auditing, and electronic

financial activities, can be supported with tools that are reusable,
convenient, consistent, and familiar. In addition, a rational
approach—a transaction/distribution control standard—allows
all participants in VDE the same foundation set of hardware
5 control and security, authoring, administration, and
management tools to support widely varying types of
information, business market model, and/or personal objectives.

Employing VDE as a general purpose electronic
10 transaction/distribution control system allows users to maintain
a single transaction management control arrangement on each of
their computers, networks, communication nodes, and/or other
electronic appliances. Such a general purpose system can serve
the needs of many electronic transaction management
15 applications without requiring distinct, different installations for
different purposes. As a result, users of VDE can avoid the
confusion and expense and other inefficiencies of different,
limited purpose transaction control applications for each different
content and/or business model. For example, VDE allows content
20 creators to use the same VDE foundation control arrangement for
both content authoring and for licensing content from other
content creators for inclusion into their products or for other use.
Clearinghouses, distributors, content creators, and other VDE
users can all interact, both with the applications running on their
25 VDE installations, and with each other, in an entirely consistent

manner, using and reusing (largely transparently) the same distributed tools, mechanisms, and consistent user interfaces, regardless of the type of VDE activity.

5 VDE prevents many forms of unauthorized use of electronic information, by controlling and auditing (and other administration of use) electronically stored and/or disseminated information. This includes, for example, commercially distributed content, electronic currency, electronic credit,
10 business transactions (such as EDI), confidential communications, and the like. VDE can further be used to enable commercially provided electronic content to be made available to users in user defined portions, rather than constraining the user to use portions of content that were "predetermined" by a content
15 creator and/or other provider for billing purposes.

VDE, for example, can employ:

- 20 (1) Secure metering means for budgeting and/or auditing electronic content and/or appliance usage;
- (2) Secure flexible means for enabling compensation and/or billing rates for content and/or appliance usage, including electronic credit and/or currency
25 mechanisms for payment means;

- 5
- (3) Secure distributed database means for storing control and usage related information (and employing validated compartmentalization and tagging schemes);
- 10
- (4) Secure electronic appliance control means;
- (5) A distributed, secure, "virtual black box" comprised of nodes located at every user (including VDE content container creators, other content providers, client users, and recipients of secure VDE content usage information) site. The nodes of said virtual black box normally include a secure subsystem having at least one secure hardware element (a semiconductor element or other hardware module for securely executing VDE control processes), said secure subsystems being distributed at nodes along a pathway of information storage, distribution, payment, usage, and/or auditing. In some
- 15
- 20
- embodiments, the functions of said hardware element, for certain or all nodes, may be performed by software, for example, in host processing environments of electronic appliances;
- 25
- (6) Encryption and decryption means;

- 5 (7) Secure communications means employing authentication, digital signaturing, and encrypted transmissions. The secure subsystems at said user nodes utilize a protocol that establishes and authenticates each node's and/or participant's identity, and establishes one or more secure host-to-host encryption keys for communications between the secure subsystems; and
- 10 (8) Secure control means that can allow each VDE installation to perform VDE content authoring (placing content into VDE containers with associated control information), content distribution, and content usage; as well as clearinghouse and other
- 15 administrative and analysis activities employing content usage information.

VDE may be used to migrate most non-electronic, traditional information delivery models (including entertainment, reference materials, catalog shopping, etc.) into an adequately secure digital distribution and usage management and payment context. The distribution and financial pathways managed by a VDE arrangement may include:

25 ! content creator(s),

- !
 - !
 - !
 - !
 - 5 !
 - !
- distributor(s),
redistributor(s),
client administrator(s),
client user(s),
financial and/or other clearinghouse(s),
and/or government agencies.

These distribution and financial pathways may also include:

- 10 !
 - !
 - !
- advertisers,
market survey organizations, and/or
other parties interested in the user usage of
information securely delivered and/or stored using
VDE.

15

Normally, participants in a VDE arrangement will employ the same secure VDE foundation. Alternate embodiments support VDE arrangements employing differing VDE foundations. Such alternate embodiments may employ procedures to ensure certain interoperability requirements are met.

20

Secure VDE hardware (also known as SPUs for Secure Processing Units), or VDE installations that use software to substitute for, or complement, said hardware (provided by Host Processing Environments (HPEs)), operate in conjunction with

25

secure communications, systems integration software, and distributed software control information and support structures, to achieve the electronic contract/rights protection environment of the present invention. Together, these VDE components

5 comprise a secure, virtual, distributed content and/or appliance control, auditing (and other administration), reporting, and payment environment. In some embodiments and where commercially acceptable, certain VDE participants, such as clearinghouses that normally maintain sufficiently physically

10 secure non-VDE processing environments, may be allowed to employ HPEs rather VDE hardware elements and interoperate, for example, with VDE end-users and content providers. VDE components together comprise a configurable, consistent, secure and "trusted" architecture for distributed, asynchronous control

15 of electronic content and/or appliance usage. VDE supports a "universe wide" environment for electronic content delivery, broad dissemination, usage reporting, and usage related payment activities.

20 VDE provides generalized configurability. This results, in part, from decomposition of generalized requirements for supporting electronic commerce and data security into a broad range of constituent "atomic" and higher level components (such as load modules, data elements, and methods) that may be

25 variously aggregated together to form control methods for

electronic commerce applications, commercial electronic agreements, and data security arrangements. VDE provides a secure operating environment employing VDE foundation elements along with secure independently deliverable VDE components that enable electronic commerce models and relationships to develop. VDE specifically supports the unfolding of distribution models in which content providers, over time, can expressly agree to, or allow, subsequent content providers and/or users to participate in shaping the control information for, and consequences of, use of electronic content and/or appliances. A very broad range of the functional attributes important for supporting simple to very complex electronic commerce and data security activities are supported by capabilities of the present invention. As a result, VDE supports most types of electronic information and/or appliance: usage control (including distribution), security, usage auditing, reporting, other administration, and payment arrangements.

VDE, in its preferred embodiment, employs object software technology and uses object technology to form "containers" for delivery of information that is (at least in part) encrypted or otherwise secured. These containers may contain electronic content products or other electronic information and some or all of their associated permissions (control) information. These container objects may be distributed along pathways involving

content providers and/or content users. They may be securely moved among nodes of a Virtual Distribution Environment (VDE) arrangement, which nodes operate VDE foundation software and execute control methods to enact electronic information usage control and/or administration models. The containers delivered through use of the preferred embodiment of the present invention may be employed both for distributing VDE control instructions (information) and/or to encapsulate and electronically distribute content that has been at least partially secured.

Content providers who employ the present invention may include, for example, software application and game publishers, database publishers, cable, television, and radio broadcasters, electronic shopping vendors, and distributors of information in electronic document, book, periodical, e-mail and/or other forms. Corporations, government agencies, and/or individual "end-users" who act as storers of, and/or distributors of, electronic information, may also be VDE content providers (in a restricted model, a user provides content only to himself and employs VDE to secure his own confidential information against unauthorized use by other parties). Electronic information may include proprietary and/or confidential information for personal or internal organization use, as well as information, such as software applications, documents, entertainment materials,

and/or reference information, which may be provided to other parties. Distribution may be by, for example, physical media delivery, broadcast and/or telecommunication means, and in the form of "static" files and/or streams of data. VDE may also be
5 used, for example, for multi-site "real-time" interaction such as teleconferencing, interactive games, or on-line bulletin boards, where restrictions on, and/or auditing of, the use of all or portions of communicated information is enforced.

10 VDE provides important mechanisms for both enforcing commercial agreements and enabling the protection of privacy rights. VDE can securely deliver information from one party to another concerning the use of commercially distributed electronic content. Even if parties are separated by several "steps" in a
15 chain (pathway) of handling for such content usage information, such information is protected by VDE through encryption and/or other secure processing. Because of that protection, the accuracy of such information is guaranteed by VDE, and the information can be trusted by all parties to whom it is delivered.

20 Furthermore, VDE guarantees that all parties can trust that such information cannot be received by anyone other than the intended, authorized, party(ies) because it is encrypted such that only an authorized party, or her agents, can decrypt it. Such information may also be derived through a secure VDE process at
25 a previous pathway-of-handling location to produce secure VDE

reporting information that is then communicated securely to its intended recipient's VDE secure subsystem. Because VDE can deliver such information securely, parties to an electronic agreement need not trust the accuracy of commercial usage and/or other information delivered through means other than those under control of VDE.

VDE participants in a commercial value chain can be "commercially" confident (that is, sufficiently confident for commercial purposes) that the direct (constituent) and/or "extended" electronic agreements they entered into through the use of VDE can be enforced reliably. These agreements may have both "dynamic" transaction management related aspects, such as content usage control information enforced through budgeting, metering, and/or reporting of electronic information and/or appliance use, and/or they may include "static" electronic assertions, such as an end-user using the system to assert his or her agreement to pay for services, not to pass to unauthorized parties electronic information derived from usage of content or systems, and/or agreeing to observe copyright laws. Not only can electronically reported transaction related information be trusted under the present invention, but payment may be automated by the passing of payment tokens through a pathway of payment (which may or may not be the same as a pathway for reporting). Such payment can be contained within a VDE container created

automatically by a VDE installation in response to control information (located, in the preferred embodiment, in one or more permissions records) stipulating the "withdrawal" of credit or electronic currency (such as tokens) from an electronic account (for example, an account securely maintained by a user's VDE installation secure subsystem) based upon usage of VDE controlled electronic content and/or appliances (such as governments, financial credit providers, and users).

VDE allows the needs of electronic commerce participants to be served and it can bind such participants together in a universe wide, trusted commercial network that can be secure enough to support very large amounts of commerce. VDE's security and metering secure subsystem core will be present at all physical locations where VDE related content is (a) assigned usage related control information (rules and mediating data), and/or (b) used. This core can perform security and auditing functions (including metering) that operate within a "virtual black box," a collection of distributed, very secure VDE related hardware instances that are interconnected by secured information exchange (for example, telecommunication) processes and distributed database means. VDE further includes highly configurable transaction operating system technology, one or more associated libraries of load modules along with affiliated data, VDE related administration, data preparation, and analysis

applications, as well as system software designed to enable VDE
integration into host environments and applications. VDE's
usage control information, for example, provide for property
content and/or appliance related: usage authorization, usage
5 auditing (which may include audit reduction), usage billing,
usage payment, privacy filtering, reporting, and security related
communication and encryption techniques.

VDE extensively employs methods in the form of software
10 objects to augment configurability, portability, and security of the
VDE environment. It also employs a software object architecture
for VDE content containers that carries protected content and
may also carry both freely available information (e.g, summary,
table of contents) and secured content control information which
15 ensures the performance of control information. Content control
information governs content usage according to criteria set by
holders of rights to an object's contents and/or according to
parties who otherwise have rights associated with distributing
such content (such as governments, financial credit providers,
20 and users).

In part, security is enhanced by object methods employed
by the present invention because the encryption schemes used to
protect an object can efficiently be further used to protect the
25 associated content control information (software control

information and relevant data) from modification. Said object techniques also enhance portability between various computer and/or other appliance environments because electronic information in the form of content can be inserted along with (for
5 example, in the same object container as) content control information (for said content) to produce a "published" object. As a result, various portions of said control information may be specifically adapted for different environments, such as for diverse computer platforms and operating systems, and said
10 various portions may all be carried by a VDE container.

An objective of VDE is supporting a transaction/distribution control standard. Development of such a standard has many obstacles, given the security requirements
15 and related hardware and communications issues, widely differing environments, information types, types of information usage, business and/or data security goals, varieties of participants, and properties of delivered information. A significant feature of VDE accommodates the many, varying
20 distribution and other transaction variables by, in part, decomposing electronic commerce and data security functions into generalized capability modules executable within a secure hardware SPU and/or corresponding software subsystem and further allowing extensive flexibility in assembling, modifying,
25 and/or replacing, such modules (e.g. load modules and/or

methods) in applications run on a VDE installation foundation. This configurability and reconfigurability allows electronic commerce and data security participants to reflect their priorities and requirements through a process of iteratively shaping an
5 evolving extended electronic agreement (electronic control model). This shaping can occur as content control information passes from one VDE participant to another and to the extent allowed by "in place" content control information. This process allows users of VDE to recast existing control information and/or
10 add new control information as necessary (including the elimination of no longer required elements).

VDE supports trusted (sufficiently secure) electronic information distribution and usage control models for both
15 commercial electronic content distribution and data security applications. It can be configured to meet the diverse requirements of a network of interrelated participants that may include content creators, content distributors, client administrators, end users, and/or clearinghouses and/or other
20 content usage information users. These parties may constitute a network of participants involved in simple to complex electronic content dissemination, usage control, usage reporting, and/or usage payment. Disseminated content may include both
25 originally provided and VDE generated information (such as content usage information) and content control information may

persist through both chains (one or more pathways) of content and content control information handling, as well as the direct usage of content. The configurability provided by the present invention is particularly critical for supporting electronic commerce, that is enabling businesses to create relationships and evolve strategies that offer competitive value. Electronic commerce tools that are not inherently configurable and interoperable will ultimately fail to produce products (and services) that meet both basic requirements and evolving needs of most commerce applications.

VDE's fundamental configurability will allow a broad range of competitive electronic commerce business models to flourish. It allows business models to be shaped to maximize revenues sources, end-user product value, and operating efficiencies. VDE can be employed to support multiple, differing models, take advantage of new revenue opportunities, and deliver product configurations most desired by users. Electronic commerce technologies that do not, as the present invention does:

- ! support a broad range of possible, complementary revenue activities,
- ! offer a flexible array of content usage features most desired by customers, and
- ! exploit opportunities for operating efficiencies,

will result in products that are often intrinsically more costly and less appealing and therefore less competitive in the marketplace.

5 Some of the key factors contributing to the configurability
intrinsic to the present invention include:

- 10 (a) integration into the fundamental control
 environment of a broad range of electronic
 appliances through portable API and programming
 language tools that efficiently support merging of
 control and auditing capabilities in nearly any
 electronic appliance environment while maintaining
 overall system security;
- 15 (b) modular data structures;
- (c) generic content model;
- 20 (d) general modularity and independence of foundation
 architectural components;
- (e) modular security structures;
- 25 (f) variable length and multiple branching chains of
 control; and

(g) independent, modular control structures in the form of executable load modules that can be maintained in one or more libraries, and assembled into control methods and models, and where such model control schemes can “evolve” as control information passes through the VDE installations of participants of a pathway of VDE content control information handling.

10 Because of the breadth of issues resolved by the present invention, it can provide the emerging “electronic highway” with a single transaction/distribution control system that can, for a very broad range of commercial and data security models, ensure against unauthorized use of confidential and/or proprietary
15 information and commercial electronic transactions. VDE’s electronic transaction management mechanisms can enforce the electronic rights and agreements of all parties participating in widely varying business and data security models, and this can be efficiently achieved through a single VDE implementation
20 within each VDE participant’s electronic appliance. VDE supports widely varying business and/or data security models that can involve a broad range of participants at various “levels” of VDE content and/or content control information pathways of handling. Different content control and/or auditing models and
25 agreements may be available on the same VDE installation.

5 These models and agreements may control content in relationship to, for example, VDE installations and/or users in general; certain specific users, installations, classes and/or other groupings of installations and/or users; as well as to electronic content generally on a given installation, to specific properties, property portions, classes and/or other groupings of content.

10 Distribution using VDE may package both the electronic content and control information into the same VDE container, and/or may involve the delivery to an end-user site of different pieces of the same VDE managed property from plural separate remote locations and/or in plural separate VDE content containers and/or employing plural different delivery means. Content control information may be partially or fully delivered
15 separately from its associated content to a user VDE installation in one or more VDE administrative objects. Portions of said control information may be delivered from one or more sources. Control information may also be available for use by access from
20 a user's VDE installation secure sub-system to one or more remote VDE secure sub-systems and/or VDE compatible, certified secure remote locations. VDE control processes such as metering, budgeting, decrypting and/or fingerprinting, may as relates to a certain user content usage activity, be performed in a user's local VDE installation secure subsystem, or said processes
25 may be divided amongst plural secure subsystems which may be

located in the same user VDE installations and/or in a network server and in the user installation. For example, a local VDE installation may perform decryption and save any, or all of, usage metering information related to content and/or electronic appliance usage at such user installation could be performed at the server employing secure (e.g., encrypted) communications between said secure subsystems. Said server location may also be used for near real time, frequent, or more periodic secure receipt of content usage information from said user installation, with, for example, metered information being maintained only temporarily at a local user installation.

Delivery means for VDE managed content may include electronic data storage means such as optical disks for delivering one portion of said information and broadcasting and/or telecommunicating means for other portions of said information. Electronic data storage means may include magnetic media, optical media, combined magneto-optical systems, flash RAM memory, bubble memory, and/or other memory storage means such as huge capacity optical storage systems employing holographic, frequency, and/or polarity data storage techniques. Data storage means may also employ layered disc techniques, such as the use of generally transparent and/or translucent materials that pass light through layers of data carrying discs which themselves are physically packaged together as one

thicker disc. Data carrying locations on such discs may be, at least in part, opaque.

5 VDE supports a general purpose foundation for secure transaction management, including usage control, auditing, reporting, and/or payment. This general purpose foundation is called "VDE Functions" ("VDEFs"). VDE also supports a collection of "atomic" application elements (e.g., load modules) that can be selectively aggregated together to form various VDEF 10 capabilities called control methods and which serve as VDEF applications and operating system functions. When a host operating environment of an electronic appliance includes VDEF capabilities, it is called a "Rights Operating System" (ROS). VDEF load modules, associated data, and methods form a body of 15 information that for the purposes of the present invention are called "control information." VDEF control information may be specifically associated with one or more pieces of electronic content and/or it may be employed as a general component of the operating system capabilities of a VDE installation.

20

VDEF transaction control elements reflect and enact content specific and/or more generalized administrative (for example, general operating system) control information. VDEF capabilities which can generally take the form of applications 25 (application models) that have more or less configurability which

can be shaped by VDE participants, through the use, for example, of VDE templates, to employ specific capabilities, along, for example, with capability parameter data to reflect the elements of one or more express electronic agreements between VDE participants in regards to the use of electronic content such as commercially distributed products. These control capabilities manage the use of, and/or auditing of use of, electronic content, as well as reporting information based upon content use, and any payment for said use. VDEF capabilities may "evolve" to reflect the requirements of one or more successive parties who receive or otherwise contribute to a given set of control information.

Frequently, for a VDE application for a given content model (such as distribution of entertainment on CD-ROM, content delivery from an Internet repository, or electronic catalog shopping and advertising, or some combination of the above) participants would be able to securely select from amongst available, alternative control methods and apply related parameter data, wherein such selection of control method and/or submission of data would constitute their "contribution" of control information.

Alternatively, or in addition, certain control methods that have been expressly certified as securely interoperable and compatible with said application may be independently submitted by a participant as part of such a contribution. In the most general example, a generally certified load module (certified for a given VDE arrangement and/or content class) may be used with many

or any VDE application that operates in nodes of said arrangement. These parties, to the extent they are allowed, can independently and securely add, delete, and/or otherwise modify the specification of load modules and methods, as well as add,
5 delete or otherwise modify related information.

Normally the party who creates a VDE content container defines the general nature of the VDEF capabilities that will and/or may apply to certain electronic information. A VDE
10 content container is an object that contains both content (for example, commercially distributed electronic information products such as computer software programs, movies, electronic publications or reference materials, etc.) and certain control information related to the use of the object's content. A creating
15 party may make a VDE container available to other parties. Control information delivered by, and/or otherwise available for use with, VDE content containers comprise (for commercial content distribution purposes) VDEF control capabilities (and any associated parameter data) for electronic content. These
20 capabilities may constitute one or more "proposed" electronic agreements (and/or agreement functions available for selection and/or use with parameter data) that manage the use and/or the consequences of use of such content and which can enact the terms and conditions of agreements involving multiple parties
25 and their various rights and obligations.

A VDE electronic agreement may be explicit, through a user interface acceptance by one or more parties, for example by a "junior" party who has received control information from a "senior" party, or it may be a process amongst equal parties who individually assert their agreement. Agreement may also result from an automated electronic process during which terms and conditions are "evaluated" by certain VDE participant control information that assesses whether certain other electronic terms and conditions attached to content and/or submitted by another party are acceptable (do not violate acceptable control information criteria). Such an evaluation process may be quite simple, for example a comparison to ensure compatibility between a portion of, or all senior, control terms and conditions in a table of terms and conditions and the submitted control information of a subsequent participant in a pathway of content control information handling, or it may be a more elaborate process that evaluates the potential outcome of, and/or implements a negotiation process between, two or more sets of control information submitted by two or more parties. VDE also accommodates a semi-automated process during which one or more VDE participants directly, through user interface means, resolve "disagreements" between control information sets by accepting and/or proposing certain control information that may be acceptable to control information representing one or more other parties interests and/or responds to certain user interface

queries for selection of certain alternative choices and/or for certain parameter information, the responses being adopted if acceptable to applicable senior control information.

5 When another party (other than the first applier of rules), perhaps through a negotiation process, accepts, and/or adds to and/or otherwise modifies, "in place" content control information, a VDE agreement between two or more parties related to the use of such electronic content may be created (so long as any
10 modifications are consistent with senior control information). Acceptance of terms and conditions related to certain electronic content may be direct and express, or it may be implicit as a result of use of content (depending, for example, on legal requirements, previous exposure to such terms and conditions,
15 and requirements of in place control information).

 VDEF capabilities may be employed, and a VDE agreement may be entered into, by a plurality of parties without the VDEF capabilities being directly associated with the
20 controlling of certain, specific electronic information. For example, certain one or more VDEF capabilities may be present at a VDE installation, and certain VDE agreements may have been entered into during the registration process for a content distribution application, to be used by such installation for
25 securely controlling VDE content usage, auditing, reporting

and/or payment. Similarly, a specific VDE participant may enter into a VDE user agreement with a VDE content or electronic appliance provider when the user and/or her appliance register with such provider as a VDE installation and/or user. In such events, VDEF in place control information available to the user VDE installation may require that certain VDEF methods are employed, for example in a certain sequence, in order to be able to use all and/or certain classes, of electronic content and/or VDE applications.

10

VDE ensures that certain prerequisites necessary for a given transaction to occur are met. This includes the secure execution of any required load modules and the availability of any required, associated data. For example, required load modules and data (e.g. in the form of a method) might specify that sufficient credit from an authorized source must be confirmed as available. It might further require certain one or more load modules execute as processes at an appropriate time to ensure that such credit will be used in order to pay for user use of the content. A certain content provider might, for example, require metering the number of copies made for distribution to employees of a given software program (a portion of the program might be maintained in encrypted form and require the presence of a VDE installation to run). This would require the execution of a metering method for copying of the property each time a copy

25

was made for another employee. This same provider might also charge fees based on the total number of different properties licensed from them by the user and a metering history of their licensing of properties might be required to maintain this
5 information.

VDE provides organization, community, and/or universe wide secure environments whose integrity is assured by processes securely controlled in VDE participant user
10 installations (nodes). VDE installations, in the preferred embodiment, may include both software and tamper resistant hardware semiconductor elements. Such a semiconductor arrangement comprises, at least in part, special purpose circuitry that has been designed to protect against tampering with, or
15 unauthorized observation of, the information and functions used in performing the VDE's control functions. The special purpose secure circuitry provided by the present invention includes at least one of: a dedicated semiconductor arrangement known as a Secure Processing Unit (SPU) and/or a standard microprocessor,
20 microcontroller, and/or other processing logic that accommodates the requirements of the present invention and functions as an SPU. VDE's secure hardware may be found incorporated into, for example, a fax/modem chip or chip pack, I/O controller, video display controller, and/or other available digital processing
25 arrangements. It is anticipated that portions of the present

invention's VDE secure hardware capabilities may ultimately be standard design elements of central processing units (CPUs) for computers and various other electronic devices.

5 Designing VDE capabilities into one or more standard
microprocessor, microcontroller and/or other digital processing
components may materially reduce VDE related hardware costs
by employing the same hardware resources for both the
transaction management uses contemplated by the present
10 invention and for other, host electronic appliance functions. This
means that a VDE SPU can employ (share) circuitry elements of
a "standard" CPU. For example, if a "standard" processor can
operate in protected mode and can execute VDE related
instructions as a protected activity, then such an embodiment
15 may provide sufficient hardware security for a variety of
applications and the expense of a special purpose processor might
be avoided. Under one preferred embodiment of the present
invention, certain memory (e.g., RAM, ROM, NVRAM) is
maintained during VDE related instruction processing in a
20 protected mode (for example, as supported by protected mode
microprocessors). This memory is located in the same package as
the processing logic (e.g. processor). Desirably, the packaging
and memory of such a processor would be designed using security
techniques that enhance its resistance to tampering.

25

The degree of overall security of the VDE system is primarily dependent on the degree of tamper resistance and concealment of VDE control process execution and related data storage activities. Employing special purpose semiconductor packaging techniques can significantly contribute to the degree of security. Concealment and tamper-resistance in semiconductor memory (e.g., RAM, ROM, NVRAM) can be achieved, in part, by employing such memory within an SPU package, by encrypting data before it is sent to external memory (such as an external RAM package) and decrypting encrypted data within the CPU/RAM package before it is executed. This process is used for important VDE related data when such data is stored on unprotected media, for example, standard host storage, such as random access memory, mass storage, etc. In that event, a VDE SPU would encrypt data that results from a secure VDE execution before such data was stored in external memory.

Summary of Some Important Features Provided by VDE in Accordance With the Present Invention

VDE employs a variety of capabilities that serve as a foundation for a general purpose, sufficiently secure distributed electronic commerce solution. VDE enables an electronic commerce marketplace that supports divergent, competitive business partnerships, agreements, and evolving overall business models. For example, VDE includes features that:

“sufficiently” impede unauthorized and/or uncompensated use of electronic information and/or appliances through the use of secure communication, storage, and transaction management technologies.

5 VDE supports a model wide, distributed security implementation which creates a single secure “virtual” transaction processing and information storage environment. VDE enables distributed VDE installations to securely store and communicate

10 information and remotely control the execution processes and the character of use of electronic information at other VDE installations and in a wide variety of ways;

15 support low-cost, efficient, and effective security architectures for transaction control, auditing, reporting, and related communications and information storage. VDE may employ tagging related security techniques, the time-ageing of

20 encryption keys, the compartmentalization of both stored control information (including differentially tagging such stored information to ensure against substitution and tampering) and distributed content (to, for many content applications, employ one or

25 more content encryption keys that are unique to the

specific VDE installation and/or user), private key techniques such as triple DES to encrypt content, public key techniques such as RSA to protect communications and to provide the benefits of digital signature and authentication to securely bind together the nodes of a VDE arrangement, secure processing of important transaction management executable code, and a combining of a small amount of highly secure, hardware protected storage space with a much larger "exposed" mass media storage space storing secured (normally encrypted and tagged) control and audit information. VDE employs special purpose hardware distributed throughout some or all locations of a VDE implementation: a) said hardware controlling important elements of: content preparation (such as causing such content to be placed in a VDE content container and associating content control information with said content), content and/or electronic appliance usage auditing, content usage analysis, as well as content usage control; and b) said hardware having been designed to securely handle processing load module control activities, wherein said control processing activities may involve a sequence of required control factors;

! support dynamic user selection of information
subsets of a VDE electronic information product
(VDE controlled content). This contrasts with the
constraints of having to use a few high level
5 individual, pre-defined content provider information
increments such as being required to select a whole
information product or product section in order to
acquire or otherwise use a portion of such product or
section. VDE supports metering and usage control
10 over a variety of increments (including "atomic"
increments, and combinations of different increment
types) that are selected ad hoc by a user and
represent a collection of pre-identified one or more
increments (such as one or more blocks of a
15 preidentified nature, e.g., bytes, images, logically
related blocks) that form a generally arbitrary, but
logical to a user, content "deliverable." VDE control
information (including budgeting, pricing and
metering) can be configured so that it can specifically
20 apply, as appropriate, to ad hoc selection of different,
unanticipated variable user selected aggregations of
information increments and pricing levels can be, at
least in part, based on quantities and/or nature of
mixed increment selections (for example, a certain
25 quantity of certain text could mean associated

images might be discounted by 15%; a greater quantity of text in the "mixed" increment selection might mean the images are discounted 20%). Such user selected aggregated information increments can reflect the actual requirements of a user for information and is more flexible than being limited to a single, or a few, high level, (e.g. product, document, database record) predetermined increments. Such high level increments may include quantities of information not desired by the user and as a result be more costly than the subset of information needed by the user if such a subset was available. In sum, the present invention allows information contained in electronic information products to be supplied according to user specification. Tailoring to user specification allows the present invention to provide the greatest value to users, which in turn will generate the greatest amount of electronic commerce activity. The user, for example, would be able to define an aggregation of content derived from various portions of an available content product, but which, as a deliverable for use by the user, is an entirely unique aggregated increment. The user may, for example, select certain numbers of bytes of information from

5 various portions of an information product, such as a
reference work, and copy them to disc in
unencrypted form and be billed based on total
number of bytes plus a surcharge on the number of
10 "articles" that provided the bytes. A content
provider might reasonably charge less for such a
user defined information increment since the user
does not require all of the content from all of the
articles that contained desired information. This
15 process of defining a user desired information
increment may involve artificial intelligence
database search tools that contribute to the location
of the most relevant portions of information from an
information product and cause the automatic display
20 to the user of information describing search criteria
hits for user selection or the automatic extraction
and delivery of such portions to the user. VDE
further supports a wide variety of predefined
increment types including:
! bytes,
! images,
! content over time for audio or video, or any
other increment that can be identified by content
provider data mapping efforts, such as:
25 ! sentences,

! paragraphs,
! articles,
! database records, and
! byte offsets representing increments of
5 logically related information.

VDE supports as many simultaneous predefined increment types as may be practical for a given type of content and business model.

10 ! securely store at a user's site potentially highly detailed information reflective of a user's usage of a variety of different content segment types and employing both inexpensive "exposed" host mass storage for maintaining detailed information in the
15 form of encrypted data and maintaining summary information for security testing in highly secure special purpose VDE installation nonvolatile memory (if available).

20 ! support trusted chain of handling capabilities for pathways of distributed electronic information and/or for content usage related information. Such chains may extend, for example, from a content creator, to a distributor, a redistributor, a client
25 user, and then may provide a pathway for securely

reporting the same and/or differing usage
information to one or more auditors, such as to one
or more independent clearinghouses and then back
to the content providers, including content creators.

5 The same and/or different pathways employed for
certain content handling, and related content control
information and reporting information handling,
may also be employed as one or more pathways for
electronic payment handling (payment is
10 characterized in the present invention as
administrative content) for electronic content and/or
appliance usage. These pathways are used for
conveyance of all or portions of content, and/or
content related control information. Content
15 creators and other providers can specify the
pathways that, partially or fully, must be used to
disseminate commercially distributed property
content, content control information, payment
administrative content, and/or associated usage
20 reporting information. Control information specified
by content providers may also specify which specific
parties must or may (including, for example, a group
of eligible parties from which a selection may be
made) handle conveyed information. It may also
25 specify what transmission means (for example

telecommunication carriers or media types) and transmission hubs must or may be used.

5 support flexible auditing mechanisms, such as employing "bitmap meters," that achieve a high degree of efficiency of operation and throughput and allow, in a practical manner, the retention and ready recall of information related to previous usage activities and related patterns. This flexibility is adaptable to a wide variety of billing and security control strategies such as:

- 10 P upgrade pricing (e.g. suite purchases),
- P pricing discounts (including quantity discounts),
- 15 P billing related time duration variables such as discounting new purchases based on the timing of past purchases, and
- P security budgets based on quantity of different, logically related units of electronic
- 20 information used over an interval of time.

25 Use of bitmap meters (including "regular" and "wide" bitmap meters) to record usage and/or purchase of information, in conjunction with other elements of the preferred embodiment of the present invention,

5 uniquely supports efficient maintenance of usage
history for: (a) rental, (b) flat fee licensing or
purchase, (c) licensing or purchase discounts based
upon historical usage variables, and (d) reporting to
users in a manner enabling users to determine
whether a certain item was acquired, or acquired
within a certain time period (without requiring the
use of conventional database mechanisms, which are
highly inefficient for these applications). Bitmap
10 meter methods record activities associated with
electronic appliances, properties, objects, or portions
thereof, and/or administrative activities that are
independent of specific properties, objects, etc.,
performed by a user and/or electronic appliance such
15 that a content and/or appliance provider and/or
controller of an administrative activity can
determine whether a certain activity has occurred at
some point, or during a certain period, in the past
(for example, certain use of a commercial electronic
20 content product and/or appliance). Such
determinations can then be used as part of pricing
and/or control strategies of a content and/or
appliance provider, and/or controller of an
administrative activity. For example, the content
25 provider may choose to charge only once for access to

a portion of a property, regardless of the number of times that portion of the property is accessed by a user.

5 support "launchable" content, that is content that
can be provided by a content provider to an end-user,
who can then copy or pass along the content to other
end-user parties without requiring the direct
participation of a content provider to register and/or
10 otherwise initialize the content for use. This content
goes "out of (the traditional distribution) channel" in
the form of a "traveling object." Traveling objects are
containers that securely carry at least some
permissions information and/or methods that are
15 required for their use (such methods need not be
carried by traveling objects if the required methods
will be available at, or directly available to, a
destination VDE installation). Certain travelling
objects may be used at some or all VDE installations
20 of a given VDE arrangement since they can make
available the content control information necessary
for content use without requiring the involvement of
a commercial VDE value chain participant or data
security administrator (e.g. a control officer or
25 network administrator). As long as traveling object

control information requirements are available at
the user VDE installation secure subsystem (such as
the presence of a sufficient quantity of financial
credit from an authorized credit provider), at least
5 some travelling object content may be used by a
receiving party without the need to establish a
connection with a remote VDE authority (until, for
example, budgets are exhausted or a time content
usage reporting interval has occurred). Traveling
10 objects can travel "out-of-channel," allowing, for
example, a user to give a copy of a traveling object
whose content is a software program, a movie or a
game, to a neighbor, the neighbor being able to use
the traveling object if appropriate credit (e.g. an
15 electronic clearinghouse account from a
clearinghouse such as VISA or AT&T) is available.
Similarly, electronic information that is generally
available on an Internet, or a similar network,
repository might be provided in the form of a
20 traveling object that can be downloaded and
subsequently copied by the initial downloader and
then passed along to other parties who may pass the
object on to additional parties.

5 provide very flexible and extensible user
identification according to individuals, installations,
by groups such as classes, and by function and
hierarchical identification employing a hierarchy of
levels of client identification (for example, client
organization ID, client department ID, client
network ID, client project ID, and client employee
ID, or any appropriate subset of the above).

10 provide a general purpose, secure, component based
content control and distribution system that
functions as a foundation transaction operating
system environment that employs executable code
pieces crafted for transaction control and auditing.
15 These code pieces can be reused to optimize
efficiency in creation and operation of trusted,
distributed transaction management arrangements.
VDE supports providing such executable code in the
form of "atomic" load modules and associated data.
20 Many such load modules are inherently configurable,
aggregatable, portable, and extensible and
singularly, or in combination (along with associated
data), run as control methods under the VDE
transaction operating environment. VDE can satisfy
25 the requirements of widely differing electronic

commerce and data security applications by, in part,
employing this general purpose transaction
management foundation to securely process VDE
transaction related control methods. Control
5 methods are created primarily through the use of
one or more of said executable, reusable load module
code pieces (normally in the form of executable object
components) and associated data. The component
nature of control methods allows the present
10 invention to efficiently operate as a highly
configurable content control system. Under the
present invention, content control models can be
iteratively and asynchronously shaped, and
otherwise updated to accommodate the needs of VDE
15 participants to the extent that such shaping and
otherwise updating conforms to constraints applied
by a VDE application, if any (e.g., whether new
component assemblies are accepted and, if so, what
certification requirements exist for such component
20 assemblies or whether any or certain participants
may shape any or certain control information by
selection amongst optional control information
(permissions record) control methods. This iterative
(or concurrent) multiple participant process occurs
25 as a result of the submission and use of secure,

control information components (executable code
such as load modules and/or methods, and/or
associated data). These components may be
contributed independently by secure communication
5 between each control information influencing VDE
participant's VDE installation and may require
certification for use with a given application, where
such certification was provided by a certification
service manager for the VDE arrangement who
10 ensures secure interoperability and/or reliability
(e.g., bug control resulting from interaction) between
appliances and submitted control methods. The
transaction management control functions of a VDE
electronic appliance transaction operating
15 environment interact with non-secure transaction
management operating system functions to properly
direct transaction processes and data related to
electronic information security, usage control,
auditing, and usage reporting. VDE provides the
20 capability to manages resources related to secure
VDE content and/or appliance control information
execution and data storage.

! facilitate creation of application and/or system
25 functionality under VDE and to facilitate integration

into electronic appliance environments of load modules and methods created under the present invention. To achieve this, VDE employs an Application Programmer's Interface (API) and/or a transaction operating system (such as a ROS) programming language with incorporated functions, both of which support the use of capabilities and can be used to efficiently and tightly integrate VDE functionality into commercial and user applications.

support user interaction through: (a) "Pop-Up" applications which, for example, provide messages to users and enable users to take specific actions such as approving a transaction, (b) stand-alone VDE applications that provide administrative environments for user activities such as: end-user preference specifications for limiting the price per transaction, unit of time, and/or session, for accessing history information concerning previous transactions, for reviewing financial information such as budgets, expenditures (e.g. detailed and/or summary) and usage analysis information, and (c) VDE aware applications which, as a result of the use of a VDE API and/or a transaction management (for

example, ROS based) programming language
embeds VDE "awareness" into commercial or
internal software (application programs, games, etc.)
so that VDE user control information and services
5 are seamlessly integrated into such software and can
be directly accessed by a user since the underlying
functionality has been integrated into the
commercial software's native design. For example,
in a VDE aware word processor application, a user
10 may be able to "print" a document into a VDE
content container object, applying specific control
information by selecting from amongst a series of
different menu templates for different purposes (for
example, a confidential memo template for internal
15 organization purposes may restrict the ability to
"keep," that is to make an electronic copy of the
memo).

! employ "templates" to ease the process of configuring
20 capabilities of the present invention as they relate to
specific industries or businesses. Templates are
applications or application add-ons under the
present invention. Templates support the efficient
specification and/or manipulation of criteria related
25 to specific content types, distribution approaches,

pricing mechanisms, user interactions with content and/or administrative activities, and/or the like.

5 Given the very large range of capabilities and configurations supported by the present invention, reducing the range of configuration opportunities to a manageable subset particularly appropriate for a given business model allows the full configurable power of the present invention to be easily employed by "typical" users who would be otherwise burdened with complex programming and/or configuration design responsibilities template applications can also help ensure that VDE related processes are secure and optimally bug free by reducing the risks associated with the contribution of independently developed load modules, including unpredictable aspects of code interaction between independent modules and applications, as well as security risks associated with possible presence of viruses in such modules. VDE, through the use of templates, reduces typical user configuration responsibilities to an appropriately focused set of activities including selection of method types (e.g. functionality) through menu choices such as multiple choice, icon selection, and/or prompting for method parameter data (such as identification information, prices, budget limits,

10

15

20

25

dates, periods of time, access rights to specific content, etc.) that supply appropriate and/or necessary data for control information purposes. By limiting the typical (non-programming) user to a limited subset of configuration activities whose general configuration environment (template) has been preset to reflect general requirements corresponding to that user, or a content or other business model can very substantially limit difficulties associated with content containerization (including placing initial control information on content), distribution, client administration, electronic agreement implementation, end-user interaction, and clearinghouse activities, including associated interoperability problems (such as conflicts resulting from security, operating system, and/or certification incompatibilities). Use of appropriate VDE templates can assure users that their activities related to content VDE containerization, contribution of other control information, communications, encryption techniques and/or keys, etc. will be in compliance with specifications for their distributed VDE arrangement. VDE templates constitute preset configurations that can normally be reconfigurable

to allow for new and/or modified templates that reflect adaptation into new industries as they evolve or to reflect the evolution or other change of an existing industry. For example, the template concept may be used to provide individual, overall frameworks for organizations and individuals that create, modify, market, distribute, consume, and/or otherwise use movies, audio recordings and live performances, magazines, telephony based retail sales, catalogs, computer software, information data bases, multimedia, commercial communications, advertisements, market surveys, infomercials, games, CAD/CAM services for numerically controlled machines, and the like. As the context surrounding these templates changes or evolves, template applications provided under the present invention may be modified to meet these changes for broad use, or for more focused activities. A given VDE participant may have a plurality of templates available for different tasks. A party that places content in its initial VDE container may have a variety of different, configurable templates depending on the type of content and/or business model related to the content. An end-user may have different configurable templates that can be applied

5

to different document types (e-mail, secure internal documents, database records, etc.) and/or subsets of users (applying differing general sets of control information to different bodies of users, for example, selecting a list of users who may, under certain preset criteria, use a certain document). Of course, templates may, under certain circumstances have fixed control information and not provide for user selections or parameter data entry.

10

15

20

25

! support plural, different control models regulating the use and/or auditing of either the same specific copy of electronic information content and/or differently regulating different copies (occurrences) of the same electronic information content. Differing models for billing, auditing, and security can be applied to the same piece of electronic information content and such differing sets of control information may employ, for control purposes, the same, or differing, granularities of electronic information control increments. This includes supporting variable control information for budgeting and auditing usage as applied to a variety of predefined increments of electronic information, including employing a variety of different budgets and/or

metering increments for a given electronic
information deliverable for: billing units of measure,
credit limit, security budget limit and security
content metering increments, and/or market
5 surveying and customer profiling content metering
increments. For example, a CD-ROM disk with a
database of scientific articles might be in part billed
according to a formula based on the number of bytes
decrypted, number of articles containing said bytes
10 decrypted, while a security budget might limit the
use of said database to no more than 5% of the
database per month for users on the wide area
network it is installed on.

15 ! provide mechanisms to persistently maintain trusted
content usage and reporting control information
through both a sufficiently secure chain of handling
of content and content control information and
through various forms of usage of such content
20 wherein said persistence of control may survive such
use. Persistence of control includes the ability to
extract information from a VDE container object by
creating a new container whose contents are at least
in part secured and that contains both the extracted
25 content and at least a portion of the control

5 information which control information of the original
container and/or are at least in part produced by
control information of the original container for this
purpose and/or VDE installation control information
stipulates should persist and/or control usage of
content in the newly formed container. Such control
information can continue to manage usage of
container content if the container is "embedded" into
another VDE managed object, such as an object
10 which contains plural embedded VDE containers,
each of which contains content derived (extracted)
from a different source.

! enables users, other value chain participants (such
15 as clearinghouses and government agencies), and/or
user organizations, to specify preferences or
requirements related to their use of electronic
content and/or appliances. Content users, such as
end-user customers using commercially distributed
20 content (games, information resources, software
programs, etc.), can define, if allowed by senior
control information, budgets, and/or other control
information, to manage their own internal use of
content. Uses include, for example, a user setting a
25 limit on the price for electronic documents that the

user is willing to pay without prior express user authorization, and the user establishing the character of metering information he or she is willing to allow to be collected (privacy protection).

5 This includes providing the means for content users to protect the privacy of information derived from their use of a VDE installation and content and/or appliance usage auditing. In particular, VDE can prevent information related to a participant's usage
10 of electronic content from being provided to other parties without the participant's tacit or explicit agreement.

! provide mechanisms that allow control information
15 to "evolve" and be modified according, at least in part, to independently, securely delivered further control information. Said control information may include executable code (e.g., load modules) that has been certified as acceptable (e.g., reliable and
20 trusted) for use with a specific VDE application, class of applications, and/or a VDE distributed arrangement. This modification (evolution) of control information can occur upon content control information (load modules and any associated data)
25 circulating to one or more VDE participants in a

5 pathway of handling of control information, or it may
occur upon control information being received from a
VDE participant. Handlers in a pathway of
handling of content control information, to the extent
each is authorized, can establish, modify, and/or
contribute to, permission, auditing, payment, and
reporting control information related to controlling,
analyzing, paying for, and/or reporting usage of,
electronic content and/or appliances (for example, as
related to usage of VDE controlled property content).
Independently delivered (from an independent
source which is independent except in regards to
certification), at least in part secure, control
information can be employed to securely modify
content control information when content control
information has flowed from one party to another
party in a sequence of VDE content control
information handling. This modification employs,
for example, one or more VDE component assemblies
being securely processed in a VDE secure subsystem.
In an alternate embodiment, control information
may be modified by a senior party through use of
their VDE installation secure sub-system after
receiving submitted, at least in part secured, control
information from a "junior" party, normally in the

form of a VDE administrative object. Control information passing along VDE pathways can represent a mixed control set, in that it may include: control information that persisted through a

5 sequence of control information handlers, other control information that was allowed to be modified, and further control information representing new control information and/or mediating data. Such a control set represents an evolution of control

10 information for disseminated content. In this example the overall content control set for a VDE content container is "evolving" as it securely (e.g. communicated in encrypted form and using authentication and digital signaturing techniques)

15 passes, at least in part, to a new participant's VDE installation where the proposed control information is securely received and handled. The received control information may be integrated (through use of the receiving parties' VDE installation secure

20 sub-system) with in-place control information through a negotiation process involving both control information sets. For example, the modification, within the secure sub-system of a content provider's VDE installation, of content control information for a

25 certain VDE content container may have occurred as

a result of the incorporation of required control information provided by a financial credit provider. Said credit provider may have employed their VDE installation to prepare and securely communicate (directly or indirectly) said required control information to said content provider. Incorporating said required control information enables a content provider to allow the credit provider's credit to be employed by a content end-user to compensate for the end-user's use of VDE controlled content and/or appliances, so long as said end-user has a credit account with said financial credit provider and said credit account has sufficient credit available.

Similarly, control information requiring the payment of taxes and/or the provision of revenue information resulting from electronic commerce activities may be securely received by a content provider. This control information may be received, for example, from a government agency. Content providers might be required by law to incorporate such control information into the control information for commercially distributed content and/or services related to appliance usage. Proposed control information is used to an extent allowed by senior control information and as determined by any

5

negotiation trade-offs that satisfy priorities stipulated by each set (the received set and the proposed set). VDE also accommodates different control schemes specifically applying to different participants (e.g., individual participants and/or participant classes (types)) in a network of VDE content handling participants.

10

! support multiple simultaneous control models for the same content property and/or property portion.

15

This allows, for example, for concurrent business activities which are dependent on electronic commercial product content distribution, such as acquiring detailed market survey information and/or supporting advertising, both of which can increase revenue and result in lower content costs to users and greater value to content providers. Such control

20

information and/or overall control models may be applied, as determined or allowed by control information, in differing manners to different participants in a pathway of content, reporting, payment, and/or related control information handling. VDE supports applying different content

25

control information to the same and/or different content and/or appliance usage related activities,

and/or to different parties in a content and/or
appliance usage model, such that different parties
(or classes of VDE users, for example) are subject to
differing control information managing their use of
5 electronic information content. For example,
differing control models based on the category of a
user as a distributor of a VDE controlled content
object or an end-user of such content may result in
different budgets being applied. Alternatively, for
10 example, a one distributor may have the right to
distribute a different array of properties than
another distributor (from a common content
collection provided, for example, on optical disc). An
individual, and/or a class or other grouping of
15 end-users, may have different costs (for example, a
student, senior citizen, and/or poor citizen user of
content who may be provided with the same or
differing discounts) than a "typical" content user.

20 ! support provider revenue information resulting from
customer use of content and/or appliances, and/or
provider and/or end-user payment of taxes, through
the transfer of credit and/or electronic currency from
said end-user and/or provider to a government
25 agency, might occur "automatically" as a result of

such received control information causing the generation of a VDE content container whose content includes customer content usage information reflecting secure, trusted revenue summary information and/or detailed user transaction listings (level of detail might depend, for example on type or size of transaction—information regarding a bank interest payment to a customer or a transfer of a large (e.g. over \$10,000) might be, by law, automatically reported to the government). Such summary and/or detailed information related to taxable events and/or currency, and/or creditor currency transfer, may be passed along a pathway of reporting and/or payment to the government in a VDE container. Such a container may also be used for other VDE related content usage reporting information.

! support the flowing of content control information through different “branches” of content control information handling so as to accommodate, under the present invention’s preferred embodiment, diverse controlled distributions of VDE controlled content. This allows different parties to employ the same initial electronic content with differing

(perhaps competitive) control strategies. In this instance, a party who first placed control information on content can make certain control assumptions and these assumptions would evolve into more specific and/or extensive control assumptions. These control assumptions can evolve during the branching sequence upon content model participants submitting control information changes, for example, for use in "negotiating" with "in place" content control information. This can result in new or modified content control information and/or it might involve the selection of certain one or more already "in-place" content usage control methods over in-place alternative methods, as well as the submission of relevant control information parameter data. This form of evolution of different control information sets applied to different copies of the same electronic property content and/or appliance results from VDE control information flowing "down" through different branches in an overall pathway of handling and control and being modified differently as it diverges down these different pathway branches. This ability of the present invention to support multiple pathway branches for the flow of both VDE content control

5

information and VDE managed content enables an electronic commerce marketplace which supports diverging, competitive business partnerships, agreements, and evolving overall business models which can employ the same content properties combined, for example, in differing collections of content representing differing at least in part competitive products.

10

15

20

25

enable a user to securely extract, through the use of the secure subsystem at the user's VDE installation, at least a portion of the content included within a VDE content container to produce a new, secure object (content container), such that the extracted information is maintained in a continually secure manner through the extraction process. Formation of the new VDE container containing such extracted content shall result in control information consistent with, or specified by, the source VDE content container, and/or local VDE installation secure subsystem as appropriate, content control information. Relevant control information, such as security and administrative information, derived, at least in part, from the parent (source) object's control information, will normally be automatically inserted

5 into a new VDE content container object containing
extracted VDE content. This process typically occurs
under the control framework of a parent object
and/or VDE installation control information
executing at the user's VDE installation secure
subsystem (with, for example, at least a portion of
this inserted control information being stored
securely in encrypted form in one or more
permissions records). In an alternative embodiment,
10 the derived content control information applied to
extracted content may be in part or whole derived
from, or employ, content control information stored
remotely from the VDE installation that performed
the secure extraction such as at a remote server
15 location. As with the content control information for
most VDE managed content, features of the present
invention allows the content's control information to:

20 (a) "evolve," for example, the extractor of content
may add new control methods and/or modify
control parameter data, such as VDE
application compliant methods, to the extent
allowed by the content's in-place control
information. Such new control information
25 might specify, for example, who may use at

- 5 least a portion of the new object, and/or how
said at least a portion of said extracted content
may be used (e.g. when at least a portion may
be used, or what portion or quantity of
portions may be used);
- 10 (b) allow a user to combine additional content
with at least a portion of said extracted
content, such as material authored by the
extractor and/or content (for example, images,
video, audio, and/or text) extracted from one or
more other VDE container objects for
placement directly into the new container;
- 15 (c) allow a user to securely edit at least a portion
of said content while maintaining said content
in a secure form within said VDE content
container;
- 20 (d) append extracted content to a pre-existing
VDE content container object and attach
associated control information -- in these
cases, user added information may be secured,
e.g., encrypted, in part or as a whole, and may
25 be subject to usage and/or auditing control

information that differs from the those applied to previously in place object content;

5

- (e) preserve VDE control over one or more portions of extracted content after various forms of usage of said portions, for example, maintain content in securely stored form while allowing "temporary" on screen display of content or allowing a software program to be maintained in secure form but transiently decrypt any encrypted executing portion of said program (all, or only a portion, of said program may be encrypted to secure the program).

10

15

Generally, the extraction features of the present invention allow users to aggregate and/or disseminate and/or otherwise use protected electronic content information extracted from content container sources while maintaining secure VDE capabilities thus preserving the rights of providers in said content information after various content usage processes.

20

! support the aggregation of portions of VDE
controlled content, such portions being subject to
differing VDE content container control information,
wherein various of said portions may have been
5 provided by independent, different content providers
from one or more different locations remote to the
user performing the aggregation. Such aggregation,
in the preferred embodiment of the present
invention, may involve preserving at least a portion
10 of the control information (e.g., executable code such
as load modules) for each of various of said portions
by, for example, embedding some or all of such
portions individually as VDE content container
objects within an overall VDE content container
15 and/or embedding some or all of such portions
directly into a VDE content container. In the latter
case, content control information of said content
container may apply differing control information
sets to various of such portions based upon said
20 portions original control information requirements
before aggregation. Each of such embedded VDE
content containers may have its own control
information in the form of one or more permissions
records. Alternatively, a negotiation between control
25 information associated with various aggregated

portions of electronic content, may produce a control information set that would govern some or all of the aggregated content portions. The VDE content control information produced by the negotiation may
5 be uniform (such as having the same load modules and/or component assemblies, and/or it may apply differing such content control information to two or more portions that constitute an aggregation of VDE controlled content such as differing metering,
10 budgeting, billing and/or payment models. For example, content usage payment may be automatically made, either through a clearinghouse, or directly, to different content providers for different portions.

15

enable flexible metering of, or other collection of information related to, use of electronic content and/or electronic appliances. A feature of the present invention enables such flexibility of metering
20 control mechanisms to accommodate a simultaneous, broad array of: (a) different parameters related to electronic information content use; (b) different increment units (bytes, documents, properties, paragraphs, images, etc.) and/or other organizations
25 of such electronic content; and/or (c) different

categories of user and/or VDE installation types, such as client organizations, departments, projects, networks, and/or individual users, etc. This feature of the present invention can be employed for content security, usage analysis (for example, market surveying), and/or compensation based upon the use and/or exposure to VDE managed content. Such metering is a flexible basis for ensuring payment for content royalties, licensing, purchasing, and/or advertising. A feature of the present invention provides for payment means supporting flexible electronic currency and credit mechanisms, including the ability to securely maintain audit trails reflecting information related to use of such currency or credit. VDE supports multiple differing hierarchies of client organization control information wherein an organization client administrator distributes control information specifying the usage rights of departments, users, and/or projects. Likewise, a department (division) network manager can function as a distributor (budgets, access rights, etc.) for department networks, projects, and/or users, etc.

! provide scalable, integratable, standardized control
means for use on electronic appliances ranging from
inexpensive consumer (for example, television
set-top appliances) and professional devices (and
5 hand-held PDAs) to servers, mainframes,
communication switches, etc. The scalable
transaction management/auditing technology of the
present invention will result in more efficient and
reliable interoperability amongst devices functioning
10 in electronic commerce and/or data security
environments. As standardized physical containers
have become essential to the shipping of physical
goods around the world, allowing these physical
containers to universally "fit" unloading equipment,
15 efficiently use truck and train space, and
accommodate known arrays of objects (for example,
boxes) in an efficient manner, so VDE electronic
content containers may, as provided by the present
invention, be able to efficiently move electronic
20 information content (such as commercially published
properties, electronic currency and credit, and
content audit information), and associated content
control information, around the world.
Interoperability is fundamental to efficient electronic
25 commerce. The design of the VDE foundation, VDE

load modules, and VDE containers, are important features that enable the VDE node operating environment to be compatible with a very broad range of electronic appliances. The ability, for example, for control methods based on load modules to execute in very "small" and inexpensive secure sub-system environments, such as environments with very little read/write memory, while also being able to execute in large memory sub-systems that may be used in more expensive electronic appliances, supports consistency across many machines. This consistent VDE operating environment, including its control structures and container architecture, enables the use of standardized VDE content containers across a broad range of device types and host operating environments. Since VDE capabilities can be seamlessly integrated as extensions, additions, and/or modifications to fundamental capabilities of electronic appliances and host operating systems, VDE containers, content control information, and the VDE foundation will be able to work with many device types and these device types will be able to consistently and efficiently interpret and enforce VDE control information. Through this integration users can also

benefit from a transparent interaction with many of
the capabilities of VDE. VDE integration with
software operating on a host electronic appliance
supports a variety of capabilities that would be
5 unavailable or less secure without such integration.
Through integration with one or more device
applications and/or device operating environments,
many capabilities of the present invention can be
presented as inherent capabilities of a given
10 electronic appliance, operating system, or appliance
application. For example, features of the present
invention include: (a) VDE system software to in
part extend and/or modify host operating systems
such that they possess VDE capabilities, such as
15 enabling secure transaction processing and
electronic information storage; (b) one or more
application programs that in part represent tools
associated with VDE operation; and/or (c) code to be
integrated into application programs, wherein such
20 code incorporates references into VDE system
software to integrate VDE capabilities and makes
such applications VDE aware (for example, word
processors, database retrieval applications,
spreadsheets, multimedia presentation authoring
25 tools, film editing software, music editing software

5 such as MIDI applications and the like, robotics control systems such as those associated with CAD/CAM environments and NCM software and the like, electronic mail systems, teleconferencing software, and other data authoring, creating, handling, and/or usage applications including combinations of the above). These one or more features (which may also be implemented in firmware or hardware) may be employed in conjunction with a VDE node secure hardware processing capability, such as a microcontroller(s), microprocessor(s), other CPU(s) or other digital processing logic.

10

15 ! employ audit reconciliation and usage pattern evaluation processes that assess, through certain, normally network based, transaction processing reconciliation and threshold checking activities, whether certain violations of security of a VDE arrangement have occurred. These processes are performed remote to VDE controlled content end-user VDE locations by assessing, for example, purchases, and/or requests, for electronic properties by a given VDE installation. Applications for such

20

25 reconciliation activities include assessing whether

the quantity of remotely delivered VDE controlled content corresponds to the amount of financial credit and/or electronic currency employed for the use of such content. A trusted organization can acquire information from content providers concerning the cost for content provided to a given VDE installation and/or user and compare this cost for content with the credit and/or electronic currency disbursements for that installation and/or user. Inconsistencies in the amount of content delivered versus the amount of disbursement can prove, and/or indicate, depending on the circumstances, whether the local VDE installation has been, at least to some degree, compromised (for example, certain important system security functions, such as breaking encryption for at least some portion of the secure subsystem and/or VDE controlled content by uncovering one or more keys). Determining whether irregular patterns (e.g. unusually high demand) of content usage, or requests for delivery of certain kinds of VDE controlled information during a certain time period by one or more VDE installations and/or users (including, for example, groups of related users whose aggregate pattern of usage is suspicious) may also be useful in determining whether security at

5 such one or more installations, and/or by such one or more users, has been compromised, particularly when used in combination with an assessment of electronic credit and/or currency provided to one or more VDE users and/or installations, by some or all of their credit and/or currency suppliers, compared with the disbursements made by such users and/or installations.

10 ! support security techniques that materially increase the time required to "break" a system's integrity. This includes using a collection of techniques that minimizes the damage resulting from comprising some aspect of the security features of the present inventions.

15

! provide a family of authoring, administrative, reporting, payment, and billing tool user applications that comprise components of the present invention's trusted/secure, universe wide, distributed transaction control and administration system.

20 These components support VDE related: object creation (including placing control information on content), secure object distribution and management (including distribution control information, financial

25

related, and other usage analysis), client internal VDE activities administration and control, security management, user interfaces, payment disbursement, and clearinghouse related functions.

5 These components are designed to support highly secure, uniform, consistent, and standardized: electronic commerce and/or data security pathway(s) of handling, reporting, and/or payment; content control and administration; and human factors (e.g.

10 user interfaces).

! support the operation of a plurality of clearinghouses, including, for example, both financial and user clearinghouse activities, such as

15 those performed by a client administrator in a large organization to assist in the organization's use of a VDE arrangement, including usage information analysis, and control of VDE activities by individuals and groups of employees such as specifying budgets

20 and the character of usage rights available under VDE for certain groups of and/or individual, client personnel, subject to control information series to control information submitted by the client administrator. At a clearinghouse, one or more VDE

25 installations may operate together with a trusted

distributed database environment (which may include concurrent database processing means). A financial clearinghouse normally receives at its location securely delivered content usage information, and user requests (such as requests for further credit, electronic currency, and/or higher credit limit). Reporting of usage information and user requests can be used for supporting electronic currency, billing, payment and credit related activities, and/or for user profile analysis and/or broader market survey analysis and marketing (consolidated) list generation or other information derived, at least in part, from said usage information. this information can be provided to content providers or other parties, through secure, authenticated encrypted communication to the VDE installation secure subsystems. Clearinghouse processing means would normally be connected to specialized I/O means, which may include high speed telecommunication switching means that may be used for secure communications between a clearinghouse and other VDE pathway participants.

securely support electronic currency and credit usage control, storage, and communication at, and

between, VDE installations. VDE further supports automated passing of electronic currency and/or credit information, including payment tokens (such as in the form of electronic currency or credit) or other payment information, through a pathway of payment, which said pathway may or may not be the same as a pathway for content usage information reporting. Such payment may be placed into a VDE container created automatically by a VDE installation in response to control information stipulating the "withdrawal" of credit or electronic currency from an electronic credit or currency account based upon an amount owed resulting from usage of VDE controlled electronic content and/or appliances. Payment credit or currency may then be automatically communicated in protected (at least in part encrypted) form through telecommunication of a VDE container to an appropriate party such as a clearinghouse, provider of original property content or appliance, or an agent for such provider (other than a clearinghouse). Payment information may be packaged in said VDE content container with, or without, related content usage information, such as metering information. An aspect of the present invention further enables certain information

5 regarding currency use to be specified as unavailable
to certain, some, or all VDE parties ("conditionally"
to fully anonymous currency) and/or further can
regulate certain content information, such as
10 currency and/or credit use related information
(and/or other electronic information usage data) to be
available only under certain strict circumstances,
such as a court order (which may itself require
authorization through the use of a court controlled
15 VDE installation that may be required to securely
access "conditionally" anonymous information).
Currency and credit information, under the
preferred embodiment of the present invention, is
treated as administrative content;

15 support fingerprinting (also known as
watermarking) for embedding in content such that
when content protected under the present invention
is released in clear form from a VDE object
20 (displayed, printed, communicated, extracted, and/or
saved), information representing the identification of
the user and/or VDE installation responsible for
transforming the content into clear form is
embedded into the released content. Fingerprinting
25 is useful in providing an ability to identify who

5 extracted information in clear form a VDE container,
or who made a copy of a VDE object or a portion of
its contents. Since the identity of the user and/or
other identifying information may be embedded in
an obscure or generally concealed manner, in VDE
10 container content and/or control information,
potential copyright violators may be deterred from
unauthorized extraction or copying. Fingerprinting
normally is embedded into unencrypted electronic
content or control information, though it can be
15 embedded into encrypted content and later placed in
unencrypted content in a secure VDE installation
sub-system as the encrypted content carrying the
fingerprinting information is decrypted. Electronic
information, such as the content of a VDE container,
20 may be fingerprinted as it leaves a network (such as
Internet) location bound for a receiving party. Such
repository information may be maintained in
unencrypted form prior to communication and be
encrypted as it leaves the repository. Fingerprinting
would preferably take place as the content leaves the
repository, but before the encryption step.
Encrypted repository content can be decrypted, for
example in a secure VDE sub-system, fingerprint
25 information can be inserted, and then the content

can be re-encrypted for transmission. Embedding identification information of the intended recipient user and/or VDE installation into content as it leaves, for example, an Internet repository, would provide important information that would identify or assist in identifying any party that managed to compromise the security of a VDE installation or the delivered content. If a party produces an authorized clear form copy of VDE controlled content, including making unauthorized copies of an authorized clear form copy, fingerprint information would point back to that individual and/or his or her VDE installation. Such hidden information will act as a strong disincentive that should dissuade a substantial portion of potential content "pirates" from stealing other parties electronic information. Fingerprint information identifying a receiving party and/or VDE installation can be embedded into a VDE object before, or during, decryption, replication, or communication of VDE content objects to receivers. Fingerprinting electronic content before it is encrypted for transfer to a customer or other user provides information that can be very useful for identifying who received certain content which may have then been distributed or made available in

unencrypted form. This information would be useful in tracking who may have "broken" the security of a VDE installation and was illegally making certain electronic content available to others.

5 Fingerprinting may provide additional, available information such as time and/or date of the release (for example extraction) of said content information. Locations for inserting fingerprints may be specified by VDE installation and/or content container control

10 information. This information may specify that certain areas and/or precise locations within properties should be used for fingerprinting, such as one or more certain fields of information or information types. Fingerprinting information may

15 be incorporated into a property by modifying in a normally undetectable way color frequency and/or the brightness of certain image pixels, by slightly modifying certain audio signals as to frequency, by modifying font character formation, etc. Fingerprint

20 information, itself, should be encrypted so as to make it particularly difficult for tampered fingerprints to be interpreted as valid. Variations in fingerprint locations for different copies of the same property; "false" fingerprint information; and

25 multiple copies of fingerprint information within a

specific property or other content which copies
employ different fingerprinting techniques such as
information distribution patterns, frequency and/or
brightness manipulation, and encryption related
5 techniques, are features of the present invention for
increasing the difficulty of an unauthorized
individual identifying fingerprint locations and
erasing and/or modifying fingerprint information.

10 ! provide smart object agents that can carry requests,
data, and/or methods, including budgets,
authorizations, credit or currency, and content. For
example, smart objects may travel to and/or from
remote information resource locations and fulfill
15 requests for electronic information content. Smart
objects can, for example, be transmitted to a remote
location to perform a specified database search on
behalf of a user or otherwise "intelligently" search
remote one or more repositories of information for
20 user desired information. After identifying desired
information at one or more remote locations, by for
example, performing one or more database searches,
a smart object may return via communication to the
user in the form of a secure "return object"
25 containing retrieved information. A user may be

charged for the remote retrieving of information, the
returning of information to the user's VDE
installation, and/or the use of such information. In
the latter case, a user may be charged only for the
5 information in the return object that the user
actually uses. Smart objects may have the means to
request use of one or more services and/or resources.
Services include locating other services and/or
resources such as information resources, language or
10 format translation, processing, credit (or additional
credit) authorization, etc. Resources include
reference databases, networks, high powered or
specialized computing resources (the smart object
may carry information to another computer to be
15 efficiently processed and then return the information
to the sending VDE installation), remote object
repositories, etc. Smart objects can make efficient
use of remote resources (e.g. centralized databases,
super computers, etc.) while providing a secure
20 means for charging users based on information
and/or resources actually used.

! support both "translations" of VDE electronic
agreements elements into modern language printed
25 agreement elements (such as English language

agreements) and translations of electronic rights protection/transaction management modern language agreement elements to electronic VDE agreement elements. This feature requires

5 maintaining a library of textual language that corresponds to VDE load modules and/or methods and/or component assemblies. As VDE methods are proposed and/or employed for VDE agreements, a listing of textual terms and conditions can be

10 produced by a VDE user application which, in a preferred embodiment, provides phrases, sentences and/or paragraphs that have been stored and correspond to said methods and/or assemblies. This feature preferably employs artificial intelligence

15 capabilities to analyze and automatically determine, and/or assist one or more users to determine, the proper order and relationship between the library elements corresponding to the chosen methods and/or assemblies so as to compose some or all

20 portions of a legal or descriptive document. One or more users, and/or preferably an attorney (if the document a legal, binding agreement), would review the generated document material upon completion and employ such additional textual information

25 and/or editing as necessary to describe non electronic

transaction elements of the agreement and make any other improvements that may be necessary. These features further support employing modern language tools that allow one or more users to make
5 selections from choices and provide answers to questions and to produce a VDE electronic agreement from such a process. This process can be interactive and the VDE agreement formulation process may employ artificial intelligence expert
10 system technology that learns from responses and, where appropriate and based at least in part on said responses, provides further choices and/or questions which "evolves" the desired VDE electronic agreement.

15 support the use of multiple VDE secure subsystems in a single VDE installation. Various security and/or performance advantages may be realized by employing a distributed VDE design within a single
20 VDE installation. For example, designing a hardware based VDE secure subsystem into an electronic appliance VDE display device, and designing said subsystem's integration with said display device so that it is as close as possible to the
25 point of display, will increase the security for video

materials by making it materially more difficult to “steal” decrypted video information as it moves from outside to inside the video system. Ideally, for example, a VDE secure hardware module would be in the same physical package as the actual display monitor, such as within the packaging of a video monitor or other display device, and such device would be designed, to the extent commercially practical, to be as tamper resistant as reasonable. As another example, embedding a VDE hardware module into an I/O peripheral may have certain advantages from the standpoint of overall system throughput. If multiple VDE instances are employed within the same VDE installation, these instances will ideally share resources to the extent practical, such as VDE instances storing certain control information and content and/or appliance usage information on the same mass storage device and in the same VDE management database.

requiring reporting and payment compliance by employing exhaustion of budgets and time ageing of keys. For example, a VDE commercial arrangement and associated content control information may involve a content provider’s content and the use of

clearinghouse credit for payment for end-user usage
of said content. Control information regarding said
arrangement may be delivered to a user's (of said
content) VDE installation and/or said financial
5 clearinghouse's VDE installation. Said control
information might require said clearinghouse to
prepare and telecommunicate to said content
provider both content usage based information in a
certain form, and content usage payment in the form
10 of electronic credit (such credit might be "owned" by
the provider after receipt and used in lieu of the
availability or adequacy of electronic currency)
and/or electronic currency. This delivery of
information and payment may employ trusted VDE
15 installation secure subsystems to securely, and in
some embodiments, automatically, provide in the
manner specified by said control information, said
usage information and payment content. Features of
the present invention help ensure that a
20 requirement that a clearinghouse report such usage
information and payment content will be observed.
For example, if one participant to a VDE electronic
agreement fails to observe such information
reporting and/or paying obligation, another
25 participant can stop the delinquent party from

5 successfully participating in VDE activities related
to such agreement. For example, if required usage
information and payment was not reported as
specified by content control information, the
"injured" party can fail to provide, through failing to
securely communicate from his VDE installation
secure subsystem, one or more pieces of secure
information necessary for the continuance of one or
more critical processes. For example, failure to
10 report information and/or payment from a
clearinghouse to a content provider (as well as any
security failures or other disturbing irregularities)
can result in the content provider not providing key
and/or budget refresh information to the
15 clearinghouse, which information can be necessary
to authorize use of the clearinghouse's credit for
usage of the provider's content and which the
clearinghouse would communicate to end-user's
during a content usage reporting communication
20 between the clearinghouse and end-user. As another
example, a distributor that failed to make payments
and/or report usage information to a content
provider might find that their budget for creating
permissions records to distribute the content
25 provider's content to users, and/or a security budget

5 limiting one or more other aspect of their use of the
provider's content, are not being refreshed by the
content provider, once exhausted or timed-out (for
example, at a predetermined date). In these and
other cases, the offended party might decide not to
refresh time ageing keys that had "aged out." Such a
use of time aged keys has a similar impact as failing
to refresh budgets or time-aged authorizations.

10 support smart card implementations of the present
invention in the form of portable electronic
appliances, including cards that can be employed as
secure credit, banking, and/or money cards. A
feature of the present invention is the use of portable
15 VDEs as transaction cards at retail and other
establishments, wherein such cards can "dock" with
an establishment terminal that has a VDE secure
sub-system and/or an online connection to a VDE
secure and/or otherwise secure and compatible
20 subsystem, such as a "trusted" financial
clearinghouse (e.g., VISA, Mastercard). The VDE
card and the terminal (and/or online connection) can
securely exchange information related to a
transaction, with credit and/or electronic currency
25 being transferred to a merchant and/or

clearinghouse and transaction information flowing
back to the card. Such a card can be used for
transaction activities of all sorts. A docking station,
such as a PCMCIA connector on an electronic
5 appliance, such as a personal computer, can receive
a consumer's VDE card at home. Such a station/card
combination can be used for on-line transactions in
the same manner as a VDE installation that is
permanently installed in such an electronic
10 appliance. The card can be used as an "electronic
wallet" and contain electronic currency as well as
credit provided by a clearinghouse. The card can act
as a convergence point for financial activities of a
consumer regarding many, if not all, merchant,
15 banking, and on-line financial transactions,
including supporting home banking activities. A
consumer can receive his paycheck and/or
investment earnings and/or "authentic" VDE content
container secured detailed information on such
20 receipts, through on-line connections. A user can
send digital currency to another party with a VDE
arrangement, including giving away such currency.
A VDE card can retain details of transactions in a
highly secure and database organized fashion so that
25 financially related information is both consolidated

and very easily retrieved and/or analyzed. Because of the VDE security, including use of effective encryption, authentication, digital signaturing, and secure database structures, the records contained within a VDE card arrangement may be accepted as valid transaction records for government and/or corporate recordkeeping requirements. In some embodiments of the present invention a VDE card may employ docking station and/or electronic appliance storage means and/or share other VDE arrangement means local to said appliance and/or available across a network, to augment the information storage capacity of the VDE card, by for example, storing dated, and/or archived, backup information. Taxes relating to some or all of an individual's financial activities may be automatically computed based on "authentic" information securely stored and available to said VDE card. Said information may be stored in said card, in said docking station, in an associated electronic appliance, and/or other device operatively attached thereto, and/or remotely, such as at a remote server site. A card's data, e.g. transaction history, can be backed up to an individual's personal computer or other electronic appliance and such an appliance

may have an integrated VDE installation of its own.
A current transaction, recent transactions (for
redundancy), or all or other selected card data may
be backed up to a remote backup repository, such a
5 VDE compatible repository at a financial
clearinghouse, during each or periodic docking for a
financial transaction and/or information
communication such as a user/merchant transaction.
Backing up at least the current transaction during a
10 connection with another party's VDE installation
(for example a VDE installation that is also on a
financial or general purpose electronic network), by
posting transaction information to a remote
clearinghouse and/or bank, can ensure that
15 sufficient backup is conducted to enable complete
reconstruction of VDE card internal information in
the event of a card failure or loss.

! support certification processes that ensure
20 authorized interoperability between various VDE
installations so as to prevent VDE arrangements
and/or installations that unacceptably deviate in
specification protocols from other VDE arrangements
and/or installations from interoperating in a manner
25 that may introduce security (integrity and/or

confidentiality of VDE secured information), process control, and/or software compatibility problems. Certification validates the identity of VDE installations and/or their components, as well as VDE users. Certification data can also serve as information that contributes to determining the decommissioning or other change related to VDE sites.

support the separation of fundamental transaction control processes through the use of event (triggered) based method control mechanisms. These event methods trigger one or more other VDE methods (which are available to a secure VDE sub-system) and are used to carry out VDE managed transaction related processing. These triggered methods include independently (separably) and securely processable component billing management methods, budgeting management methods, metering management methods, and related auditing management processes. As a result of this feature of the present invention, independent triggering of metering, auditing, billing, and budgeting methods, the present invention is able to efficiently, concurrently support multiple financial currencies (e.g. dollars,

marks, yen) and content related budgets, and/or
billing increments as well as very flexible content
distribution models.

5 ! support, complete, modular separation of the control
structures related to (1) content event triggering, (2)
auditing, (3) budgeting (including specifying no right
of use or unlimited right of use), (4) billing, and (5)
user identity (VDE installation, client name,
10 department, network, and/or user, etc.). The
independence of these VDE control structures
provides a flexible system which allows plural
relationships between two or more of these
structures, for example, the ability to associate a
15 financial budget with different event trigger
structures (that are put in place to enable controlling
content based on its logical portions). Without such
separation between these basic VDE capabilities, it
would be more difficult to efficiently maintain
20 separate metering, budgeting, identification, and/or
billing activities which involve the same, differing
(including overlapping), or entirely different,
portions of content for metering, billing, budgeting,
and user identification, for example, paying fees
25 associated with usage of content, performing home

banking, managing advertising services, etc. VDE modular separation of these basic capabilities supports the programming of plural, "arbitrary" relationships between one or differing content portions (and/or portion units) and budgeting, auditing, and/or billing control information. For example, under VDE, a budget limit of \$200 dollars or 300 German Marks a month may be enforced for decryption of a certain database and 2 U.S. Dollars or 3 German Marks may be charged for each record of said database decrypted (depending on user selected currency). Such usage can be metered while an additional audit for user profile purposes can be prepared recording the identity of each filed displayed. Additionally, further metering can be conducted regarding the number of said database bytes that have been decrypted, and a related security budget may prevent the decrypting of more than 5% of the total bytes of said database per year. The user may also, under VDE (if allowed by senior control information), collect audit information reflecting usage of database fields by different individuals and client organization departments and ensure that differing rights of access and differing budgets limiting database usage can be applied to

5 these client individuals and groups. Enabling
content providers and users to practically employ
such diverse sets of user identification, metering,
budgeting, and billing control information results, in
part, from the use of such independent control
capabilities. As a result, VDE can support great
configurability in creation of plural control models
applied to the same electronic property and the same
and/or plural control models applied to differing or
10 entirely different content models (for example, home
banking versus electronic shopping).

Methods, Other Control Information, and VDE Objects

15 VDE control information (e.g., methods) that collectively
control use of VDE managed properties (database, document,
individual commercial product), are either shipped with the
content itself (for example, in a content container) and/or one or
more portions of such control information is shipped to
distributors and/or other users in separably deliverable
20 "administrative objects." A subset of the methods for a property
may in part be delivered with each property while one or more
other subsets of methods can be delivered separately to a user or
otherwise made available for use (such as being available
remotely by telecommunication means). Required methods
25 (methods listed as required for property and/or appliance use)

must be available as specified if VDE controlled content (such as intellectual property distributed within a VDE content container) is to be used. Methods that control content may apply to a plurality of VDE container objects, such as a class or other grouping of such objects. Methods may also be required by certain users or classes of users and/or VDE installations and/or classes of installations for such parties to use one or more specific, or classes of, objects.

10 A feature of VDE provided by the present invention is that certain one or more methods can be specified as required in order for a VDE installation and/or user to be able to use certain and/or all content. For example, a distributor of a certain type of content might be allowed by "senior" participants (by content creators, for example) to require a method which prohibits end-users from electronically saving decrypted content, a provider of credit for VDE transactions might require an audit method that records the time of an electronic purchase, and/or a user might require a method that summarizes usage information for reporting to a clearinghouse (e.g. billing information) in a way that does not convey confidential, personal information regarding detailed usage behavior.

25 A further feature of VDE provided by the present invention is that creators, distributors, and users of content can select from

among a set of predefined methods (if available) to control
container content usage and distribution functions and/or they
may have the right to provide new customized methods to control
at least certain usage functions (such "new" methods may be
5 required to be certified for trustedness and interoperability to the
VDE installation and/or for of a group of VDE applications). As a
result, VDE provides a very high degree of configurability with
respect to how the distribution and other usage of each property
or object (or one or more portions of objects or properties as
10 desired and/or applicable) will be controlled. Each VDE
participant in a VDE pathway of content control information may
set methods for some or all of the content in a VDE container, so
long as such control information does not conflict with senior
control information already in place with respect to:

15

- (1) certain or all VDE managed content,
- (2) certain one or more VDE users and/or groupings of
users,
- 20 (3) certain one or more VDE nodes and/or groupings of
nodes, and/or
- (4) certain one or more VDE applications and/or
25 arrangements.

For example, a content creator's VDE control information for certain content can take precedence over other submitted VDE participant control information and, for example, if allowed by senior control information, a content distributor's control information may itself take precedence over a client administrator's control information, which may take precedence over an end-user's control information. A path of distribution participant's ability to set such electronic content control information can be limited to certain control information (for example, method mediating data such as pricing and/or sales dates) or it may be limited only to the extent that one or more of the participant's proposed control information conflicts with control information set by senior control information submitted previously by participants in a chain of handling of the property, or managed in said participant's VDE secure subsystem.

VDE control information may, in part or in full, (a) represent control information directly put in place by VDE content control information pathway participants, and/or (b) comprise control information put in place by such a participant on behalf of a party who does not directly handle electronic content (or electronic appliance) permissions records information (for example control information inserted by a participant on behalf of a financial clearinghouse or government agency). Such control information methods (and/or load modules and/or

mediating data and/or component assemblies) may also be put in place by either an electronic automated, or a semi-automated and human assisted, control information (control set) negotiating process that assesses whether the use of one or more pieces of submitted control information will be integrated into and/or replace existing control information (and/or chooses between alternative control information based upon interaction with in-place control information) and how such control information may be used.

10

Control information may be provided by a party who does not directly participate in the handling of electronic content (and/or appliance) and/or control information for such content (and/or appliance). Such control information may be provided in secure form using VDE installation secure sub-system managed communications (including, for example, authenticating the deliverer of at least in part encrypted control information) between such not directly participating one or more parties' VDE installation secure subsystems, and a pathway of VDE content control information participant's VDE installation secure subsystem. This control information may relate to, for example, the right to access credit supplied by a financial services provider, the enforcement of regulations or laws enacted by a government agency, or the requirements of a customer of VDE managed content usage information (reflecting usage of content

25

by one or more parties other than such customer) relating to the creation, handling and/or manner of reporting of usage information received by such customer. Such control information may, for example, enforce societal requirements such as laws
5 related to electronic commerce.

VDE content control information may apply differently to different pathway of content and/or control information handling participants. Furthermore, permissions records rights may be
10 added, altered, and/or removed by a VDE participant if they are allowed to take such action. Rights of VDE participants may be defined in relation to specific parties and/or categories of parties and/or other groups of parties in a chain of handling of content and/or content control information (e.g., permissions records).
15 Modifications to control information that may be made by a given, eligible party or parties, may be limited in the number of modifications, and/or degree of modification, they may make.

At least one secure subsystem in electronic appliances of
20 creators, distributors, auditors, clearinghouses, client administrators, and end-users (understanding that two or more of the above classifications may describe a single user) provides a "sufficiently" secure (for the intended applications) environment
for:

25

1. Decrypting properties and control information;
 2. Storing control and metering related information;
 - 5 3. Managing communications;
 4. Processing core control programs, along with associated data, that constitute control information for electronic content and/or appliance rights protection, including the enforcing of preferences and requirements of VDE participants.
- 10

Normally, most usage, audit, reporting, payment, and distribution control methods are themselves at least in part encrypted and are executed by the secure subsystem of a VDE installation. Thus, for example, billing and metering records can be securely generated and updated, and encryption and decryption keys are securely utilized, within a secure subsystem. Since VDE also employs secure (e.g. encrypted and authenticated) communications when passing information between the participant location (nodes) secure subsystems of a VDE arrangement, important components of a VDE electronic agreement can be reliably enforced with sufficient security (sufficiently trusted) for the intended commercial purposes. A VDE electronic agreement for a value chain can be composed, at

15

20

25

least in part, of one or more subagreements between one or more subsets of the value chain participants. These subagreements are comprised of one or more electronic contract "compliance" elements (methods including associated parameter data) that
5 ensure the protection of the rights of VDE participants.

The degree of trustedness of a VDE arrangement will be primarily based on whether hardware SPUs are employed at participant location secure subsystems and the effectiveness of
10 the SPU hardware security architecture, software security techniques when an SPU is emulated in software, and the encryption algorithm(s) and keys that are employed for securing content, control information, communications, and access to VDE node (VDE installation) secure subsystems. Physical facility and
15 user identity authentication security procedures may be used instead of hardware SPUs at certain nodes, such as at an established financial clearinghouse, where such procedures may provide sufficient security for trusted interoperability with a VDE arrangement employing hardware SPUs at user nodes.

20

The updating of property management files at each location of a VDE arrangement, to accommodate new or modified control information, is performed in the VDE secure subsystem and under the control of secure management file updating
25 programs executed by the protected subsystem. Since all secure

communications are at least in part encrypted and the processing
inside the secure subsystem is concealed from outside
observation and interference, the present invention ensures that
content control information can be enforced. As a result, the
5 creator and/or distributor and/or client administrator and/or
other contributor of secure control information for each property
(for example, an end-user restricting the kind of audit
information he or she will allow to be reported and/or a financial
clearinghouse establishing certain criteria for use of its credit for
10 payment for use of distributed content) can be confident that
their contributed and accepted control information will be
enforced (within the security limitations of a given VDE security
implementation design). This control information can determine,
for example:

15

(1) How and/or to whom electronic content can be
provided, for example, how an electronic property
can be distributed;

20

(2) How one or more objects and/or properties, or
portions of an object or property, can be directly
used, such as decrypted, displayed, printed, etc;

25

(3) How payment for usage of such content and/or
content portions may or must be handled; and

- (4) . How audit information about usage information related to at least a portion of a property should be collected, reported, and/or used.

5 Seniority of contributed control information, including resolution of conflicts between content control information submitted by multiple parties, is normally established by:

- 10 (1) the sequence in which control information is put in place by various parties (in place control information normally takes precedence over subsequently submitted control information),
- 15 (2) the specifics of VDE content and/or appliance control information. For example, in-place control information can stipulate which subsequent one or more piece of control from one or more parties or class of parties will take precedence over control information submitted by one or more yet different
- 20 parties and/or classes of parties, and/or
- (3) negotiation between control information sets from plural parties, which negotiation establishes what control information shall constitute the resulting

control information set for a given piece of VDE
managed content and/or VDE installation.

Electronic Agreements and Rights Protection

5 An important feature of VDE is that it can be used to
assure the administration of, and adequacy of security and rights
protection for, electronic agreements implemented through the
use of the present invention. Such agreements may involve one
or more of:

10

(1) creators, publishers, and other distributors, of
electronic information,

(2) financial service (e.g. credit) providers,

15

(3) users of (other than financial service providers)
information arising from content usage such as
content specific demographic information and user
specific descriptive information. Such users may
include market analysts, marketing list compilers for
direct and directed marketing, and government
agencies,

20

(4) end users of content,

25

- 5 (5) infrastructure service and device providers such as telecommunication companies and hardware manufacturers (semiconductor and electronic appliance and/or other computer system manufacturers) who receive compensation based upon the use of their services and/or devices, and
- (6) certain parties described by electronic information.

10 VDE supports commercially secure "extended" value chain electronic agreements. VDE can be configured to support the various underlying agreements between parties that comprise this extended agreement. These agreements can define important electronic commerce considerations including:

- 15
- (1) security,
- (2) content use control, including electronic distribution,
- 20 (3) privacy (regarding, for example, information concerning parties described by medical, credit, tax, personal, and/or of other forms of confidential information),
- 25 (4) management of financial processes, and

- (5) pathways of handling for electronic content, content and/or appliance control information, electronic content and/or appliance usage information and payment and/or credit.

5

VDE agreements may define the electronic commerce relationship of two or more parties of a value chain, but such agreements may, at times, not directly obligate or otherwise directly involve other VDE value chain participants. For example, an electronic agreement between a content creator and a distributor may establish both the price to the distributor for a creator's content (such as for a property distributed in a VDE container object) and the number of copies of this object that this distributor may distribute to end-users over a given period of time. In a second agreement, a value chain end-user may be involved in a three party agreement in which the end-user agrees to certain requirements for using the distributed product such as accepting distributor charges for content use and agreeing to observe the copyright rights of the creator. A third agreement might exist between the distributor and a financial clearinghouse that allows the distributor to employ the clearinghouse's credit for payment for the product if the end-user has a separate (fourth) agreement directly with the clearinghouse extending credit to the end-user. A fifth, evolving agreement may develop between all value chain participants as content control

25

information passes along its chain of handling. This evolving agreement can establish the rights of all parties to content usage information, including, for example, the nature of information to be received by each party and the pathway of handling of content usage information and related procedures. A sixth agreement in this example, may involve all parties to the agreement and establishes certain general assumptions, such as security techniques and degree of trustedness (for example, commercial integrity of the system may require each VDE installation secure subsystem to electronically warrant that their VDE node meets certain interoperability requirements). In the above example, these six agreements could comprise agreements of an extended agreement for this commercial value chain instance.

VDE agreements support evolving ("living") electronic agreement arrangements that can be modified by current and/or new participants through very simple to sophisticated "negotiations" between newly proposed content control information interacting with control information already in place and/or by negotiation between concurrently proposed content control information submitted by a plurality of parties. A given model may be asynchronously and progressively modified over time in accordance with existing senior rules and such modification may be applied to all, to classes of, and/or to specific content, and/or to classes and/or specific users and/or user nodes.

A given piece of content may be subject to different control information at different times or places of handling, depending on the evolution of its content control information (and/or on differing, applicable VDE installation content control information). The evolution of control information can occur during the passing along of one or more VDE control information containing objects, that is control information may be modified at one or more points along a chain of control information handling, so long as such modification is allowed. As a result, VDE managed content may have different control information applied at both different "locations" in a chain of content handling and at similar locations in differing chains of the handling of such content. Such different application of control information may also result from content control information specifying that a certain party or group of parties shall be subject to content control information that differs from another party or group of parties. For example, content control information for a given piece of content may be stipulated as senior information and therefore not changeable, might be put in place by a content creator and might stipulate that national distributors of a given piece of their content may be permitted to make 100,000 copies per calendar quarter, so long as such copies are provided to boni fide end-users, but may pass only a single copy of such content to a local retailers and the control information limits such a retailer to making no more than 1,000 copies per month for retail sales to

end-users. In addition, for example, an end-user of such content might be limited by the same content control information to making three copies of such content, one for each of three different computers he or she uses (one desktop computer at work, one for a desktop computer at home, and one for a portable computer).

Electronic agreements supported by the preferred embodiment of the present invention can vary from very simple to very elaborate. They can support widely diverse information management models that provide for electronic information security, usage administration, and communication and may support:

15

(a) secure electronic distribution of information, for example commercial literary properties,

20

(b) secure electronic information usage monitoring and reporting,

25

(c) secure financial transaction capabilities related to both electronic information and/or appliance usage and other electronic credit and/or currency usage and administration capabilities,

- (d) privacy protection for usage information a user does not wish to release, and
- (e) "living" electronic information content dissemination models that flexibly accommodate:
 - (1) a breadth of participants,
 - (2) one or more pathways (chains) for: the handling of content, content and/or appliance control information, reporting of content and/or appliance usage related information, and/or payment,
 - (3) supporting an evolution of terms and conditions incorporated into content control information, including use of electronic negotiation capabilities,
 - (4) support the combination of multiple pieces of content to form new content aggregations, and
 - (5) multiple concurrent models.

Secure Processing Units

An important part of VDE provided by the present invention is the core secure transaction control arrangement, herein called an SPU (or SPUs), that typically must be present in each user's computer, other electronic appliance, or network. SPUs provide a trusted environment for generating decryption keys, encrypting and decrypting information, managing the secure communication of keys and other information between electronic appliances (i.e. between VDE installations and/or between plural VDE instances within a single VDE installation), securely accumulating and managing audit trail, reporting, and budget information in secure and/or non-secure non-volatile memory, maintaining a secure database of control information management instructions, and providing a secure environment for performing certain other control and administrative functions.

A hardware SPU (rather than a software emulation) within a VDE node is necessary if a highly trusted environment for performing certain VDE activities is required. Such a trusted environment may be created through the use of certain control software, one or more tamper resistant hardware modules such as a semiconductor or semiconductor chipset (including, for example, a tamper resistant hardware electronic appliance peripheral device), for use within, and/or operatively connected to, an electronic appliance. With the present invention, the

trustedness of a hardware SPU can be enhanced by enclosing some or all of its hardware elements within tamper resistant packaging and/or by employing other tamper resisting techniques (e.g. microfusing and/or thin wire detection techniques). A

5 trusted environment of the present invention implemented, in part, through the use of tamper resistant semiconductor design, contains control logic, such as a microprocessor, that securely executes VDE processes.

10 A VDE node's hardware SPU is a core component of a VDE secure subsystem and may employ some or all of an electronic appliance's primary control logic, such as a microcontroller, microcomputer or other CPU arrangement. This primary control logic may be otherwise employed for non VDE purposes such as

15 the control of some or all of an electronic appliance's non-VDE functions. When operating in a hardware SPU mode, said primary control logic must be sufficiently secure so as to protect and conceal important VDE processes. For example, a hardware SPU may employ a host electronic appliance microcomputer

20 operating in protected mode while performing VDE related activities, thus allowing portions of VDE processes to execute with a certain degree of security. This alternate embodiment is in contrast to the preferred embodiment wherein a trusted environment is created using a combination of one or more

25 tamper resistant semiconductors that are not part of said

primary control logic. In either embodiment, certain control information (software and parameter data) must be securely maintained within the SPU, and further control information can be stored externally and securely (e.g. in encrypted and tagged form) and loaded into said hardware SPU when needed. In many cases, and in particular with microcomputers, the preferred embodiment approach of employing special purpose secure hardware for executing said VDE processes, rather than using said primary control logic, may be more secure and efficient. The level of security and tamper resistance required for trusted SPU hardware processes depends on the commercial requirements of particular markets or market niches, and may vary widely.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features and advantages provided by the present invention(s) may be better and more completely understood by referring to the following detailed description of presently preferred example embodiments in connection with the drawings, of which:

FIGURE 1 illustrates an example of a "Virtual Distribution Environment" provided in accordance with a preferred example/embodiment of this invention;

FIGURE 1A is a more detailed illustration of an example of the "Information Utility" shown in FIGURE 1;

FIGURE 2 illustrates an example of a chain of handling and control;

FIGURE 2A illustrates one example of how rules and control information may persist from one participant to another in the Figure 2 chain of handling and control;

FIGURE 3 shows one example of different control information that may be provided;

5 FIGURE 4 illustrates examples of some different types of rules and/or control information;

FIGURES 5A and 5B show an example of an "object";

10 FIGURE 6 shows an example of a Secure Processing Unit ("SPU");

FIGURE 7 shows an example of an electronic appliance;

15 FIGURE 8 is a more detailed block diagram of an example of the electronic appliance shown in FIGURE 7;

FIGURE 9 is a detailed view of an example of the Secure Processing Unit (SPU) shown in FIGURES 6 and 8;

20 FIGURE 10 shows an example of a "Rights Operating System" ("ROS") architecture provided by the Virtual Distribution Environment;

FIGURES 11A-11C show examples of functional relationship(s) between applications and the Rights Operating System;

5 FIGURES 11D-11J show examples of “components” and “component assemblies”;

FIGURE 12 is a more detailed diagram of an example of the Rights Operating System shown in FIGURE 10;

10

FIGURE 12A shows an example of how “objects” can be created;

FIGURE 13 is a detailed block diagram of an example the software architecture for a “protected processing environment” shown in FIGURE 12;

15

FIGURES 14A-14C are examples of SPU memory maps provided by the protected processing environment shown in FIGURE 13;

20

FIGURE 15 illustrates an example of how the channel services manager and load module execution manager of FIGURE 13 can support a channel;

5 FIGURE 15A is an example of a channel header and channel detail records shown in FIGURE 15;

 FIGURE 15B is a flowchart of an example of program control steps that may be performed by the FIGURE 13 protected
10 processing environment to create a channel;

 FIGURE 16 is a block diagram of an example of a secure data base structure;

15 FIGURE 17 is an illustration of an example of a logical object structure;

 FIGURE 18 shows an example of a stationary object structure;

20

 FIGURE 19 shows an example of a traveling object structure;

FIGURE 20 shows an example of a content object structure;

5 FIGURE 21 shows an example of an administrative object structure;

FIGURE 22 shows an example of a method core structure;

10 FIGURE 23 shows an example of a load module structure;

FIGURE 24 shows an example of a User Data Element (UDE) and/or Method Data Element (MDE) structure;

15 FIGURES 25A-25C show examples of "map meters";

FIGURE 26 shows an example of a permissions record (PERC) structure;

20 FIGURES 26A and 26B together show a more detailed example of a permissions record structure;

FIGURE 27 shows an example of a shipping table structure;

5 FIGURE 28 shows an example of a receiving table structure;

FIGURE 29 shows an example of an administrative event log structure;

10 FIGURE 30 shows an example inter-relationship between and use of the object registration table, subject table and user rights table shown in the FIGURE 16 secure database;

15 FIGURE 31 is a more detailed example of an object registration table shown in FIGURE 16;

FIGURE 32 is a more detailed example of subject table shown in FIGURE 16;

20 FIGURE 33 is a more detailed example of a user rights table shown in FIGURE 16;

FIGURE 34 shows a specific example of how a site record table and group record table may track portions of the secure database shown in FIGURE 16;

5 FIGURE 34A is an example of a FIGURE 34 site record table structure;

FIGURE 34B is an example of a FIGURE 34 group record table structure;

10

FIGURE 35 shows an example of a process for updating the secure database;

15 FIGURE 36 shows an example of how new elements may be inserted into the FIGURE 16 secure data base;

FIGURE 37 shows an example of how an element of the secure database may be accessed;

20 FIGURE 38 is a flowchart example of how to protect a secure database element;

FIGURE 39 is a flowchart example of how to back up a
secure database;

5 FIGURE 40 is a flowchart example of how to recover a
secure database from a backup;

10 FIGURES 41A-41D are a set of examples showing how a
“chain of handling and control” may be enabled using “reciprocal
methods”;

FIGURES 42A-42D show an example of a “reciprocal”
BUDGET method;

15 FIGURES 43A-43D show an example of a “reciprocal”
REGISTER method;

FIGURES 44A-44C show an example of a “reciprocal”
AUDIT method;

20 FIGURES 45-48 show examples of several methods being
used together to control release of content or other information;

FIGURES 49, 49A-49F show an example OPEN method;

FIGURES 50, 50A-50F show an example of a READ
method;

5

FIGURES 51, 51A-51F show an example of a WRITE
method;

FIGURE 52 shows an example of a CLOSE method;

10

FIGURES 53A-53B show an example of an EVENT
method;

FIGURE 53C shows an example of a BILLING method;

15

FIGURE 54 shows an example of an ACCESS method;

FIGURES 55A-55B show examples of DECRYPT and
ENCRYPT methods;

20

FIGURE 56 shows an example of a CONTENT method;

FIGURES 57A and 57B show examples of EXTRACT and
EMBED methods;

5

FIGURE 58A shows an example of an OBSCURE method;

FIGURES 58B, 58C show examples of a FINGERPRINT
method;

10

FIGURE 59 shows an example of a DESTROY method;

FIGURE 60 shows an example of a PANIC method;

FIGURE 61 shows an example of a METER method;

15

FIGURE 62 shows an example of a key "convolution"
process;

20

FIGURE 63 shows an example of how different keys may
be generated using a key convolution process to determine a
"true" key;

FIGURES 64 and 65 show an example of how protected processing environment keys may be initialized;

5 FIGURES 66 and 67 show example processes for decrypting information contained within stationary and traveling objects, respectively;

10 FIGURE 68 shows an example of how a protected processing environment may be initialized;

FIGURE 69 shows an example of how firmware may be downloaded into a protected processing environment;

15 FIGURE 70 shows an example of multiple VDE electronic appliances connected together with a network or other communications means;

20 FIGURE 71 shows an example of a portable VDE electronic appliance;

FIGURES 72A-72D show examples of "pop-up" displays that may be generated by the user notification and exception interface;

5 FIGURE 73 shows an example of a "smart object";

FIGURE 74 shows an example of a process using "smart objects";

10 FIGURES 75A-75D show examples of data structures used for electronic negotiation;

FIGURES 75E-75F show example structures relating to an electronic agreement;

15

FIGURES 76A-76B show examples of electronic negotiation processes;

20

FIGURE 77 shows a further example of a chain of handling and control;

FIGURE 78 shows an example of a VDE "repository";

FIGURES 79-83 show an example illustrating a chain of handling and control to evolve and transform VDE managed content and control information;

5 FIGURE 84 shows a further example of a chain of handling and control involving several categories of VDE participants;

FIGURE 85 shows a further example of a chain of distribution and handling within an organization;

10

Figures 86 and 86A show a further example of a chain of handling and control; and

15 Figure 87 shows an example of a virtual silicon container model.

MORE DETAILED DESCRIPTION

Figures 1-7 and the discussion below provides an overview of some aspects of features provided by this invention. Following this overview is a more technical "detail description" of example embodiments in accordance with the invention.

Overview

Figure 1 shows a "Virtual Distribution Environment" ("VDE") 100 that may be provided in accordance with this invention. In Figure 1, an information utility 200 connects to communications means 202 such as telephone or cable TV lines for example. Telephone or cable TV lines 202 may be part of an "electronic highway" that carries electronic information from place to place. Lines 202 connect information utility 200 to other people

such as for example a consumer 208, an office 210, a video
production studio 204, and a publishing house 214. Each of the
people connected to information utility 200 may be called a "VDE
participant" because they can participate in transactions
5 occurring within the virtual distribution environment 100.

Almost any sort of transaction you can think of can be
supported by virtual distribution environment 100. A few of
many examples of transactions that can be supported by virtual
10 distribution environment 100 include:

- C home banking and electronic payments;
- C electronic legal contracts;
- C distribution of "content" such as electronic printed matter,
video, audio, images and computer programs; and
- 15 C secure communication of private information such as
medical records and financial information.

Virtual distribution environment 100 is "virtual" because it
does not require many of the physical "things" that used to be
20 necessary to protect rights, ensure reliable and predictable
distribution, and ensure proper compensation to content creators
and distributors. For example, in the past, information was

distributed on records or disks that were difficult to copy. In the past, private or secret content was distributed in sealed envelopes or locked briefcases delivered by courier. To ensure appropriate compensation, consumers received goods and services only after they handed cash over to a seller. Although information utility 200 may deliver information by transferring physical "things" such as electronic storage media, the virtual distribution environment 100 facilitates a completely electronic "chain of handling and control."

10

VDE Flexibility Supports Transactions

Information utility 200 flexibly supports many different kinds of information transactions. Different VDE participants may define and/or participate in different parts of a transaction. Information utility 200 may assist with delivering information about a transaction, or it may be one of the transaction participants.

15

20

For example, the video production studio 204 in the upper right-hand corner of Figure 1 may create video/television programs. Video production studio 204 may send these programs over lines 202, or may use other paths such as satellite

link 205 and CD ROM delivery service 216. Video production studio 204 can send the programs directly to consumers 206, 208, 210, or it can send the programs to information utility 200 which may store and later send them to the consumers, for example.

5 Consumers 206, 208, 210 are each capable of receiving and using the programs created by video production studio 204—assuming, that is, that the video production studio or information utility 200 has arranged for these consumers to have appropriate “rules and controls” (control information) that give the consumers rights to
10 use the programs.

Even if a consumer has a copy of a video program, she cannot watch or copy the program unless she has “rules and controls” that authorize use of the program. She can use the
15 program only as permitted by the “rules and controls.”

For example, video production studio 204 might release a half-hour exercise video in the hope that as many viewers as possible will view it. The video production studio 204 wishes to
20 receive \$2.00 per viewing. Video production studio 204 may, through information utility 200, make the exercise video available in “protected” form to all consumers 206, 208, 210.

Video production studio 204 may also provide "rules and controls" for the video. These "rules and controls" may specify for example:

5 (1) any consumer who has good credit of at least \$2.00 based on a credit account with independent financial provider 212 (such as Mastercard or VISA) may watch the video,

10 (2) virtual distribution environment 100 will "meter" each time a consumer watches the video, and report usage to video production studio 204 from time to time, and

15 (3) financial provider 212 may electronically collect payment (\$2.00) from the credit account of each consumer who watches the video, and transfer these payments to the video production studio 204.

20 Information utility 200 allows even a small video production studio to market videos to consumers and receive compensation for its efforts. Moreover, the videos can, with appropriate payment to the video production studio, be made

available to other video publishers who may add value and/or act as repackagers or redistributors.

5 Figure 1 also shows a publishing house 214. Publishing house 214 may act as a distributor for an author 206. The publishing house 214 may distribute rights to use "content" (such as computer software, electronic newspapers, the video produced by publishing house 214, audio, or any other data) to consumers such as office 210. The use rights may be defined by "rules and controls" distributed by publishing house 216. Publishing house 10 216 may distribute these "rules and controls" with the content, but this is not necessary. Because the content can be used only by consumers that have the appropriate "rules and controls," content and its associated "rules and controls" may be distributed 15 at different times, in different ways, by different VDE participants. The ability of VDE to securely distribute and enforce "rules and controls" separately from the content they apply to provides great advantages.

20 Use rights distributed by publishing house 214 may, for example, permit office 210 to make and distribute copies of the content to its employees. Office 210 may act as a redistributor by

extending a "chain of handling and control" to its employees. The office 210 may add or modify "rules and controls" (consistent with the "rules and controls" it receives from publishing house 214) to provide office-internal control information and mechanisms. For
5 example, office 210 may set a maximum usage budget for each individual user and/or group within the office, or it may permit only specified employees and/or groups to access certain information.

10 Figure 1 also shows an information delivery service 216 delivering electronic storage media such as "CD ROM" disks to consumers 206. Even though the electronic storage media themselves are not delivered electronically by information utility
200 over lines 202, they are still part of the virtual distribution
15 environment 100. The electronic storage media may be used to distribute content, "rules and controls," or other information.

Example of What's Inside Information Utility 200

"Information utility" 200 in Figure 1 can be a collection of
20 participants that may act as distributors, financial clearinghouses, and administrators. Figure 1A shows an example of what may be inside one example of information utility

200. Information utility participants 200a-200g could each be an independent organization/business. There can be any number of each of participants 200a-200g. In this example, electronic “switch” 200a connects internal parts of information utility 200 to each other and to outside participants, and may also connect outside participants to one another.

Information utility 200 may include a “transaction processor” 200b that processes transactions (to transfer electronic funds, for example) based on requests from participants and/or report receiver 200e. It may also include a “usage analyst” 200c that analyzes reported usage information. A “report creator” 200d may create reports based on usage for example, and may provide these reports to outside participants and/or to participants within information utility 200. A “report receiver” 200e may receive reports such as usage reports from content users. A “permissioning agent” 200f may distribute “rules and controls” granting usage or distribution permissions based on a profile of a consumer’s credit worthiness, for example. An administrator 200h may provide information that keeps the virtual distribution environment 100 operating properly. A

content and message storage 200g may store information for use by participants within or outside of information utility 200.

5 **Example of Distributing Content” Using A Chain of Handling and Control”**

As explained above, virtual distribution environment 100 can be used to manage almost any sort of transaction. One type of important transaction that virtual distribution environment 100 may be used to manage is the distribution or communication of “content” or other important information. Figure 2 more abstractly shows a “model” of how the Figure 1 virtual distribution environment 100 may be used to provide a “chain of handling and control” for distributing content. Each of the blocks in Figure 2 may correspond to one or more of the VDE participants shown in Figure 1.

In the Figure 2 example, a VDE content creator 102 creates “content.” The content creator 102 may also specify “rules and controls” for distributing the content. These distribution-related “rules and controls” can specify who has permission to distribute the rights to use content, and how many users are allowed to use the content.

Arrow 104 shows the content creator 102 sending the "rules and controls" associated with the content to a VDE rights distributor 106 ("distributor") over an electronic highway 108 (or by some other path such as an optical disk sent by a delivery service such as U. S. mail). The content can be distributed over the same or different path used to send the "rules and controls." The distributor 106 generates her own "rules and controls" that relate to usage of the content. The usage-related "rules and controls" may, for example, specify what a user can and can't do with the content and how much it costs to use the content. These usage-related "rules and controls" must be consistent with the "rules and controls" specified by content creator 102.

Arrow 110 shows the distributor 106 distributing rights to use the content by sending the content's "rules and controls" to a content user 112 such as a consumer. The content user 112 uses the content in accordance with the usage-related "rules and controls."

In this Figure 2 example, information relating to content use is, as shown by arrow 114, reported to a financial clearinghouse 116. Based on this "reporting," the financial

clearinghouse 116 may generate a bill and send it to the content user 112 over a "reports and payments" network 118. Arrow 120 shows the content user 112 providing payments for content usage to the financial clearinghouse 116. Based on the reports and payments it receives, the financial clearinghouse 116 may provide reports and/or payments to the distributor 106. The distributor 106 may, as shown by arrow 122, provide reports and/or payments to the content creator 102. The clearinghouse 116 may provide reports and payments directly to the creator 102. Reporting and/or payments may be done differently. For example, clearinghouse 116 may directly or through an agent, provide reports and/or payments to each of VDE content creators 102, and rights distributor 106, as well as reports to content user 112.

15

The distributor 106 and the content creator 102 may be the same person, or they may be different people. For example, a musical performing group may act as both content creator 102 and distributor 106 by creating and distributing its own musical recordings. As another example, a publishing house may act as a distributor 106 to distribute rights to use works created by an author content creator 102. Content creators 102 may use a

20

distributor 106 to efficiently manage the financial end of content distribution.

5 The "financial clearinghouse" 116 shown in Figure 2 may also be a "VDE administrator." Financial clearinghouse 116 in its VDE administrator role sends "administrative" information to the VDE participants. This administrative information helps to keep the virtual distribution environment 100 operating properly. The "VDE administrator" and financial clearinghouse
10 roles may be performed by different people or companies, and there can be more than one of each.

More about Rules and Controls"

15 The virtual distribution environment 100 prevents use of protected information except as permitted by the "rules and controls" (control information). For example, the "rules and controls" shown in Figure 2 may grant specific individuals or classes of content users 112 "permission" to use certain content. They may specify what kinds of content usage are permitted, and
20 what kinds are not. They may specify how content usage is to be paid for and how much it costs. As another example, "rules and

controls” may require content usage information to be reported back to the distributor 106 and/or content creator 102.

5 Every VDE participant in “chain of handling and control”
is normally subject to “rules and controls.” “Rules and controls”
define the respective rights and obligations of each of the various
VDE participants. “Rules and controls” provide information and
mechanisms that may establish interdependencies and
relationships between the participants. “Rules and controls” are
10 flexible, and permit “virtual distribution environment” 100 to
support most “traditional” business transactions. For example:
C “Rules and controls” may specify which financial
clearinghouse(s) 116 may process payments,
C “Rules and controls” may specify which participant(s)
15 receive what kind of usage report, and
C “Rules and controls” may specify that certain information
is revealed to certain participants, and that other
information is kept secret from them.

20 “Rules and controls” may self limit if and how they may be
changed. Often, “rules and controls” specified by one VDE
participant cannot be changed by another VDE participant. For

example, a content user 112 generally can't change "rules and controls" specified by a distributor 106 that require the user to pay for content usage at a certain rate. "Rules and controls" may "persist" as they pass through a "chain of handling and control," and may be "inherited" as they are passed down from one VDE participant to the next.

Depending upon their needs, VDE participants can specify that their "rules and controls" can be changed under conditions specified by the same or other "rules and controls." For example, "rules and controls" specified by the content creator 102 may permit the distributor 106 to "mark up" the usage price just as retail stores "mark up" the wholesale price of goods. Figure 2A shows an example in which certain "rules and controls" persist unchanged from content creator 102 to content user 112; other "rules and controls" are modified or deleted by distributor 106; and still other "rules and controls" are added by the distributor.

"Rules and controls" can be used to protect the content user's privacy by limiting the information that is reported to other VDE participants. As one example, "rules and controls" can cause content usage information to be reported anonymously

without revealing content user identity, or it can reveal only certain information to certain participants (for example, information derived from usage) with appropriate permission, if required. This ability to securely control what information is revealed and what VDE participant(s) it is revealed to allows the privacy rights of all VDE participants to be protected.

Rules and Contents" Can Be Separately Delivered

As mentioned above, virtual distribution environment 100 "associates" content with corresponding "rules and controls," and prevents the content from being used or accessed unless a set of corresponding "rules and controls" is available. The distributor 106 doesn't need to deliver content to control the content's distribution. The preferred embodiment can securely protect content by protecting corresponding, usage enabling "rules and controls" against unauthorized distribution and use.

In some examples, "rules and controls" may travel with the content they apply to. Virtual distribution environment 100 also allows "rules and controls" to be delivered separately from content. Since no one can use or access protected content without "permission" from corresponding "rules and controls," the

distributor 106 can control use of content that has already been (or will in the future be) delivered. "Rules and controls" may be delivered over a path different from the one used for content delivery. "Rules and controls" may also be delivered at some
5 other time. The content creator 102 might deliver content to content user 112 over the electronic highway 108, or could make the content available to anyone on the highway. Content may be used at the time it is delivered, or it may be stored for later use or reuse.

10

The virtual distribution environment 100 also allows payment and reporting means to be delivered separately. For example, the content user 112 may have a virtual "credit card" that extends credit (up to a certain limit) to pay for usage of any
15 content. A "credit transaction" can take place at the user's site without requiring any "online" connection or further authorization. This invention can be used to help securely protect the virtual "credit card" against unauthorized use.

20 **Rules and Contents" Define Processes**

Figure 3 shows an example of an overall process based on "rules and controls." It includes an "events" process 402, a meter

process 404, a billing process 406, and a budget process 408. Not all of the processes shown in Figure 3 will be used for every set of "rules and controls."

5 The "events process" 402 detects things that happen ("events") and determines which of those "events" need action by the other "processes." The "events" may include, for example, a request to use content or generate a usage permission. Some events may need additional processing, and others may not.

10 Whether an "event" needs more processing depends on the "rules and controls" corresponding to the content. For example, a user who lacks permission will not have her request satisfied ("No Go"). As another example, each user request to turn to a new page of an electronic book may be satisfied ("Go"), but it may not

15 be necessary to meter, bill or budget those requests. A user who has purchased a copy of a novel may be permitted to open and read the novel as many times as she wants to without any further metering, billing or budgeting. In this simple example, the "event process" 402 may request metering, billing and/or

20 budgeting processes the first time the user asks to open the protected novel (so the purchase price can be charged to the user), and treat all later requests to open the same novel as

“insignificant events.” Other content (for example, searching an electronic telephone directory) may require the user to pay a fee for each access.

5 “Meter” process 404 keeps track of events, and may report usage to distributor 106 and/or other appropriate VDE participant(s). Figure 4 shows that process 404 can be based on a number of different factors such as:

- (a) type of usage to charge for,
- 10 (b) what kind of unit to base charges on,
- (c) how much to charge per unit,
- (d) when to report, and
- (e) how to pay.

These factors may be specified by the “rules and controls” that
15 control the meter process.

Billing process 406 determines how much to charge for events. It records and reports payment information.

20 Budget process 408 limits how much content usage is permitted. For example, budget process 408 may limit the number of times content may be accessed or copied, or it may

limit the number of pages or other amount of content that can be used based on, for example, the number of dollars available in a credit account. Budget process 408 records and reports financial and other transaction information associated with such limits.

5

Content may be supplied to the user once these processes have been successfully performed.

Containers and Objects*

10

Figure 5A shows how the virtual distribution environment 100, in a preferred embodiment, may package information elements (content) into a "container" 302 so the information can't be accessed except as provided by its "rules and controls."

Normally, the container 302 is electronic rather than physical.

15

Electronic container 302 in one example comprises "digital" information having a well defined structure. Container 302 and its contents can be called an "object 300."

20

The Figure 5A example shows items "within" and enclosed by container 302. However, container 302 may "contain" items without those items actually being stored within the container. For example, the container 302 may reference items that are

available elsewhere such as in other containers at remote sites. Container 302 may reference items available at different times or only during limited times. Some items may be too large to store within container 302. Items may, for example, be delivered to the user in the form of a "live feed" of video at a certain time. Even then, the container 302 "contains" the live feed (by reference) in this example.

Container 302 may contain information content 304 in electronic (such as "digital") form. Information content 304 could be the text of a novel, a picture, sound such as a musical performance or a reading, a movie or other video, computer software, or just about any other kind of electronic information you can think of. Other types of "objects" 300 (such as "administrative objects") may contain "administrative" or other information instead of or in addition to information content 304.

In the Figure 5A example, container 302 may also contain "rules and controls" in the form of:

- (a) a "permissions record" 808;
- (b) "budgets" 308; and
- (c) "other methods" 1000.

Figure 5B gives some additional detail about permissions record 808, budgets 308 and other methods 1000. The “permissions record” 808 specifies the rights associated with the object 300 such as, for example, who can open the container 302, who can use the object’s contents, who can distribute the object, and what other control mechanisms must be active. For example, permissions record 808 may specify a user’s rights to use, distribute and/or administer the container 302 and its content. Permissions record 808 may also specify requirements to be applied by the budgets 308 and “other methods” 1000. Permissions record 808 may also contain security related information such as scrambling and descrambling “keys.”

“Budgets” 308 shown in Figure 5B are a special type of “method” 1000 that may specify, among other things, limitations on usage of information content 304, and how usage will be paid for. Budgets 308 can specify, for example, how much of the total information content 304 can be used and/or copied. The methods 310 may prevent use of more than the amount specified by a specific budget.

“Other methods” 1000 define basic operations used by “rules and controls.” Such “methods” 1000 may include, for example, how usage is to be “metered,” if and how content 304 and other information is to be scrambled and descrambled, and other processes associated with handling and controlling information content 304. For example, methods 1000 may record the identity of anyone who opens the electronic container 302, and can also control how information content is to be charged based on “metering.” Methods 1000 may apply to one or several different information contents 304 and associated containers 302, as well as to all or specific portions of information content 304.

Secure Processing Unit (SPU)

The “VDE participants” may each have an “electronic appliance.” The appliance may be or contain a computer. The appliances may communicate over the electronic highway 108. Figure 6 shows a secure processing unit (“SPU”) 500 portion of the “electronic appliance” used in this example by each VDE participant. SPU 500 processes information in a secure processing environment 503, and stores important information securely. SPU 500 may be emulated by software operating in a host electronic appliance.

SPU 500 is enclosed within and protected by a "tamper resistant security barrier" 502. Security barrier 502 separates the secure environment 503 from the rest of the world. It prevents information and processes within the secure environment 503 from being observed, interfered with and leaving except under appropriate secure conditions. Barrier 502 also controls external access to secure resources, processes and information within SPU 500. In one example, tamper resistant security barrier 502 is formed by security features such as "encryption," and hardware that detects tampering and/or destroys sensitive information within secure environment 503 when tampering is detected.

SPU 500 in this example is an integrated circuit ("IC") "chip" 504 including "hardware" 506 and "firmware" 508. SPU 500 connects to the rest of the electronic appliance through an "appliance link" 510. SPU "firmware" 508 in this example is "software" such as a "computer program(s)" "embedded" within chip 504. Firmware 508 makes the hardware 506 work. Hardware 506 preferably contains a processor to perform instructions specified by firmware 508. "Hardware" 506 also contains long-term and short-term memories to store information

securely so it can't be tampered with. SPU 500 may also have a protected clock/calendar used for timing events. The SPU hardware 506 in this example may include special purpose electronic circuits that are specially designed to perform certain processes (such as "encryption" and "decryption") rapidly and efficiently.

The particular context in which SPU 500 is being used will determine how much processing capabilities SPU 500 should have. SPU hardware 506, in this example, provides at least enough processing capabilities to support the secure parts of processes shown in Figure 3. In some contexts, the functions of SPU 500 may be increased so the SPU can perform all the electronic appliance processing, and can be incorporated into a general purpose processor. In other contexts, SPU 500 may work alongside a general purpose processor, and therefore only needs to have enough processing capabilities to handle secure processes.

VDE Electronic Appliance and Rights Operating System"

Figure 7 shows an example of an electronic appliance 600 including SPU 500. Electronic appliance 600 may be practically any kind of electrical or electronic device, such as:

5

- C a computer
- C a T.V. "set top" control box
- C a pager
- C a telephone
- 10 C a sound system
- C a video reproduction system
- C a video game player
- C a "smart" credit card

15

Electronic appliance 600 in this example may include a keyboard or keypad 612, a voice recognizer 613, and a display 614. A human user can input commands through keyboard 612 and/or voice recognizer 613, and may view information on display 614. Appliance 600 may communicate with the outside world through
20 any of the connections/devices normally used within an electronic appliance. The connections/devices shown along the bottom of the drawing are examples:

- a "modem" 618 or other telecommunications link;
- a CD ROM disk 620 or other storage medium or device;
- a printer 622;
- broadcast reception 624;
- 5 a document scanner 626; and
- a "cable" 628 connecting the appliance with a "network."

Virtual distribution environment 100 provides a "rights operating system" 602 that manages appliance 600 and SPU 500
10 by controlling their hardware resources. The operating system 602 may also support at least one "application" 608. Generally, "application" 608 is hardware and/or software specific to the context of appliance 600. For example, if appliance 600 is a personal computer, then "application" 608 could be a program
15 loaded by the user, for instance, a word processor, a communications system or a sound recorder. If appliance 600 is a television controller box, then application 608 might be hardware or software that allows a user to order videos on demand and perform other functions such as fast forward and rewind. In this
20 example, operating system 602 provides a standardized, well defined, generalized "interface" that could support and work with many different "applications" 608.

Operating system 602 in this example provides “rights and auditing operating system functions” 604 and “other operating system functions” 606. The “rights and auditing operating system functions” 604 securely handle tasks that relate to virtual
5 distribution environment 100. SPU 500 provides or supports many of the security functions of the “rights and auditing operating system functions” 402. The “other operating system functions” 606 handle general appliance functions. Overall operating system 602 may be designed from the beginning to
10 include the “rights and auditing operating system functions” 604 plus the “other operating system functions” 606, or the “rights and auditing operating system functions” may be an add-on to a preexisting operating system providing the “other operating system functions.”

15

“Rights operating system” 602 in this example can work with many different types of appliances 600. For example, it can work with large mainframe computers, “minicomputers” and “microcomputers” such as personal computers and portable
20 computing devices. It can also work in control boxes on the top of television sets, small portable “pagers,” desktop radios, stereo sound systems, telephones, telephone switches, or any other

electronic appliance. This ability to work on big appliances as well as little appliances is called "scalable." A "scalable" operating system 602 means that there can be a standardized interface across many different appliances performing a wide
5 variety of tasks.

The "rights operating system functions" 604 are "services-based" in this example. For example, "rights operating system functions" 604 handle summary requests from application 608
10 rather than requiring the application to always make more detailed "subrequests" or otherwise get involved with the underlying complexities involved in satisfying a summary request. For example, application 608 may simply ask to read specified information; "rights operating system functions" 604
15 can then decide whether the desired information is VDE-protected content and, if it is, perform processes needed to make the information available. This feature is called "transparency."
"Transparency" makes tasks easy for the application 608.
"Rights operating system functions" 604 can support applications
20 608 that "know" nothing about virtual distribution environment 100. Applications 608 that are "aware" of virtual distribution

environment 100 may be able to make more detailed use of virtual distribution environment 100.

5 In this example, "rights operating system functions" 604 are "event driven". Rather than repeatedly examining the state of electronic appliance 600 to determine whether a condition has arisen, the "rights operating system functions" 604 may respond directly to "events" or "happenings" within appliance 600.

10 In this example, some of the services performed by "rights operating system functions" 604 may be extended based on additional "components" delivered to operating system 602. "Rights operating system functions" 604 can collect together and use "components" sent by different participants at different
15 times. The "components" help to make the operating system 602 "scalable." Some components can change how services work on little appliances versus how they work on big appliances (e.g., multi-user). Other components are designed to work with specific applications or classes of applications (e.g., some types of meters
20 and some types of budgets).

Electronic Appliance 600

An electronic appliance 600 provided by the preferred embodiment may, for example, be any electronic apparatus that contains one or more microprocessors and/or microcontrollers and/or other devices which perform logical and/or mathematical calculations. This may include computers; computer terminals; 5 device controllers for use with computers; peripheral devices for use with computers; digital display devices; televisions; video and audio/video projection systems; channel selectors and/or decoders for use with broadcast and/or cable transmissions; remote control 10 devices; video and/or audio recorders; media players including compact disc players, videodisc players and tape players; audio and/or video amplifiers; virtual reality machines; electronic game players; multimedia players; radios; telephones; videophones; facsimile machines; robots; numerically controlled machines 15 including machine tools and the like; and other devices containing one or more microcomputers and/or microcontrollers and/or other CPUs, including those not yet in existence.

Figure 8 shows an example of an electronic appliance 600. 20 This example of electronic appliance 600 includes a system bus 653. In this example, one or more conventional general purpose central processing units ("CPUs") 654 are connected to bus 653.

Bus 653 connects CPU(s) 654 to RAM 656, ROM 658, and I/O controller 660. One or more SPUs 500 may also be connected to system bus 653. System bus 653 may permit SPU(s) 500 to communicate with CPU(s) 654, and also may allow both the
5 CPU(s) and the SPU(s) to communicate (e.g., over shared address and data lines) with RAM 656, ROM 658 and I/O controller 660. A power supply 659 may provide power to SPU 500, CPU 654 and the other system components shown.

10 In the example shown, I/O controller 660 is connected to secondary storage device 652, a keyboard/display 612,614, a communications controller 666, and a backup storage device 668. Backup storage device 668 may, for example, store information on mass media such as a tape 670, a floppy disk, a removable
15 memory card, etc. Communications controller 666 may allow electronic appliance 600 to communicate with other electronic appliances via network 672 or other telecommunications links. Different electronic appliances 600 may interoperate even if they use different CPUs and different instances of ROS 602, so long as
20 they typically use compatible communication protocols and/or security methods. In this example, I/O controller 660 permits CPU 654 and SPU 500 to read from and write to secondary

storage 662, keyboard/display 612, 614, communications controller 666, and backup storage device 668.

5 Secondary storage 662 may comprise the same one or more non-secure secondary storage devices (such as a magnetic disk and a CD-ROM drive as one example) that electronic appliance 600 uses for general secondary storage functions. In some implementations, part or all of secondary storage 652 may comprise a secondary storage device(s) that is physically enclosed
10 within a secure enclosure. However, since it may not be practical or cost-effective to physically secure secondary storage 652 in many implementations, secondary storage 652 may be used to store information in a secure manner by encrypting information before storing it in secondary storage 652. If information is
15 encrypted before it is stored, physical access to secondary storage 652 or its contents does not readily reveal or compromise the information.

20 Secondary storage 652 in this example stores code and data used by CPU 654 and/or SPU 500 to control the overall operation of electronic appliance 600. For example, Figure 8 shows that "Rights Operating System" ("ROS") 602 (including a

portion 604 of ROS that provides VDE functions and a portion
606 that provides other OS functions) shown in Figure 7 may be
stored on secondary storage 652. Secondary storage 652 may
also store one or more VDE objects 300. Figure 8 also shows that
5 the secure files 610 shown in Figure 7 may be stored on
secondary storage 652 in the form of a "secure database" or
management file system 610. This secure database 610 may
store and organize information used by ROS 602 to perform VDE
functions 604. Thus, the code that is executed to perform VDE
10 and other OS functions 604, 606, and secure files 610 (as well as
VDE objects 300) associated with those functions may be stored
in secondary storage 652. Secondary storage 652 may also store
"other information" 673 such as, for example, information used by
other operating system functions 606 for task management, non-
15 VDE files, etc. Portions of the elements indicated in secondary
storage 652 may also be stored in ROM 658, so long as those
elements do not require changes (except when ROM 658 is
replaced). Portions of ROS 602 in particular may desirably be
included in ROM 658 (e.g., "bootstrap" routines, POST routines,
20 etc. for use in establishing an operating environment for
electronic appliance 600 when power is applied).

Figure 8 shows that secondary storage 652 may also be used to store code ("application programs") providing user application(s) 608 shown in Figure 7. Figure 8 shows that there may be two general types of application programs 608: "VDE aware" applications 608a, and Non-VDE aware applications 608b. VDE aware applications 608a may have been at least in part designed specifically with VDE 100 in mind to access and take detailed advantage of VDE functions 604. Because of the "transparency" features of ROS 602, non-VDE aware applications 608b (e.g., applications not specifically designed for VDE 100) can also access and take advantage of VDE functions 604.

SECURE PROCESSING UNIT 500

Each VDE node or other electronic appliance 600 in the preferred embodiment may include one or more SPUs 500. SPUs 500 may be used to perform all secure processing for VDE 100. For example, SPU 500 is used for decrypting (or otherwise unsecuring) VDE protected objects 300. It is also used for managing encrypted and/or otherwise secured communication (such as by employing authentication and/or error-correction validation of information). SPU 500 may also perform secure data management processes including governing usage of,

auditing of, and where appropriate, payment for VDE objects 300
(through the use of prepayments, credits, real-time electronic
debits from bank accounts and/or VDE node currency token
deposit accounts). SPU 500 may perform other transactions
5 related to such VDE objects 300.

SPU Physical Packaging and Security Barrier 502

As shown Figure 6, in the preferred embodiment, an SPU
500 may be implemented as a single integrated circuit "chip" 505
10 to provide a secure processing environment in which confidential
and/or commercially valuable information can be safely
processed, encrypted and/or decrypted. IC chip 505 may, for
example, comprise a small semiconductor "die" about the size of a
thumbnail. This semiconductor die may include semiconductor
15 and metal conductive pathways. These pathways define the
circuitry, and thus the functionality, of SPU 500. Some of these
pathways are electrically connected to the external "pins" 504 of
the chip 505.

20 As shown in Figures 6 and 9, SPU 500 may be surrounded
by a tamper-resistant hardware security barrier 502. Part of this
security barrier 502 is formed by a plastic or other package in

which an SPU "die" is encased. Because the processing occurring within, and information stored by, SPU 500 are not easily accessible to the outside world, they are relatively secure from unauthorized access and tampering. All signals cross barrier 502 through a secure, controlled path provided by BIU 530 that restricts the outside world's access to the internal components within SPU 500. This secure, controlled path resists attempts from the outside world to access secret information and resources within SPU 500.

10

It is possible to remove the plastic package of an IC chip and gain access to the "die." It is also possible to analyze and "reverse engineer" the "die" itself (e.g., using various types of logic analyzers and microprobes to collect and analyze signals on the die while the circuitry is operating, using acid etching or other techniques to remove semiconductor layers to expose other layers, viewing and photographing the die using an electron microscope, etc.) Although no system or circuit is absolutely impervious to such attacks, SPU barrier 502 may include additional hardware protections that make successful attacks exceedingly costly and time consuming. For example, ion implantation and/or other fabrication techniques may be used to

20

make it very difficult to visually discern SPU die conductive pathways, and SPU internal circuitry may be fabricated in such a way that it "self-destructs" when exposed to air and/or light. SPU 500 may store secret information in internal memory that loses its contents when power is lost. Circuitry may be incorporated within SPU 500 that detects microprobing or other tampering, and self-destructs (or destroys other parts of the SPU) when tampering is detected. These and other hardware-based physical security techniques contribute to tamper-resistant hardware security barrier 502.

To increase the security of security barrier 502 even further, it is possible to encase or include SPU 500 in one or more further physical enclosures such as, for example: epoxy or other "potting compound"; further module enclosures including additional self-destruct, self-disabling or other features activated when tampering is detected; further modules providing additional security protections such as requiring password or other authentication to operate; and the like. In addition, further layers of metal may be added to the die to complicate acid etching, micro probing, and the like; circuitry designed to "zeroize" memory may be included as an aspect of self-destruct

processes; the plastic package itself may be designed to resist chemical as well as physical "attacks"; and memories internal to SPU 500 may have specialized addressing and refresh circuitry that "shuffles" the location of bits to complicate efforts to electrically determine the value of memory locations. These and other techniques may contribute to the security of barrier 502.

In some electronic appliances 600, SPU 500 may be integrated together with the device microcontroller or equivalent or with a device I/O or communications microcontroller into a common chip (or chip set) 505. For example, in one preferred embodiment, SPU 500 may be integrated together with one or more other CPU(s) (e.g., a CPU 654 of an electronic appliance) in a single component or package. The other CPU(s) 654 may be any centrally controlling logic arrangement, such as for example, a microprocessor, other microcontroller, and/or array or other parallel processor. This integrated configuration may result in lower overall cost, smaller overall size, and potentially faster interaction between an SPU 500 and a CPU 654. Integration may also provide wider distribution if an integrated SPU/CPU component is a standard feature of a widely distributed microprocessor line. Merging an SPU 500 into a main CPU 654

of an electronic appliance 600 (or into another appliance or
appliance peripheral microcomputer or other microcontroller)
may substantially reduce the overhead cost of implementing VDE
100. Integration considerations may include cost of
5 implementation, cost of manufacture, desired degree of security,
and value of compactness.

SPU 500 may also be integrated with devices other than
CPUs. For example, for video and multimedia applications, some
10 performance and/or security advantages (depending on overall
design) could result from integrating an SPU 500 into a video
controller chip or chipset. SPU 500 can also be integrated
directly into a network communications chip or chipset or the
like. Certain performance advantages in high speed
15 communications applications may also result from integrating an
SPU 500 with a modem chip or chipset. This may facilitate
incorporation of an SPU 500 into communication appliances such
as stand-alone fax machines. SPU 500 may also be integrated
into other peripheral devices, such as CD-ROM devices, set-top
20 cable devices, game devices, and a wide variety of other electronic
appliances that use, allow access to, perform transactions related
to, or consume, distributed information.

SPU 500 Internal Architecture

Figure 9 is a detailed diagram of the internal structure within an example of SPU 500. SPU 500 in this example includes a single microprocessor 520 and a limited amount of memory configured as ROM 532 and RAM 534. In more detail, this example of SPU 500 includes microprocessor 520, an encrypt/decrypt engine 522, a DMA controller 526, a real-time clock 528, a bus interface unit ("BIU") 530, a read only memory (ROM) 532, a random access memory (RAM) 534, and a memory management unit ("MMU") 540. DMA controller 526 and MMU 540 are optional, but the performance of SPU 500 may suffer if they are not present. SPU 500 may also include an optional pattern matching engine 524, an optional random number generator 542, an optional arithmetic accelerator circuit 544, and optional compression/decompression circuit 546. A shared address/data bus arrangement 536 may transfer information between these various components under control of microprocessor 520 and/or DMA controller 526. Additional or alternate dedicated paths 538 may connect microprocessor 520 to the other components (e.g., encrypt/decrypt engine 522 via line 538a, real-time clock 528 via line 538b, bus interface unit 530 via

line 538c, DMA controller via line 538d, and memory management unit (MMU) 540 via line 538e).

5 The following section discusses each of these SPU components in more detail.

Microprocessor 520

10 Microprocessor 520 is the "brain" of SPU 500. In this example, it executes a sequence of steps specified by code stored (at least temporarily) within ROM 532 and/or RAM 534.

15 Microprocessor 520 in the preferred embodiment comprises a dedicated central processing arrangement (e.g., a RISC and/or CISC processor unit, a microcontroller, and/or other central processing means or, less desirably in most applications, process specific dedicated control logic) for executing instructions stored in the ROM 532 and/or other memory. Microprocessor 520 may be separate elements of a circuitry layout, or may be separate packages within a secure SPU 500.

20 In the preferred embodiment, microprocessor 520 normally handles the most security sensitive aspects of the operation of electronic appliance 600. For example, microprocessor 520 may

manage VDE decrypting, encrypting, certain content and/or appliance usage control information, keeping track of usage of VDE secured content, and other VDE usage control related functions.

5

Stored in each SPU 500 and/or electronic appliance secondary memory 652 may be, for example, an instance of ROS 602 software, application programs 608, objects 300 containing VDE controlled property content and related information, and management database 610 that stores both information associated with objects and VDE control information. ROS 602 includes software intended for execution by SPU microprocessor 520 for, in part, controlling usage of VDE related objects 300 by electronic appliance 600. As will be explained, these SPU programs include "load modules" for performing basic control functions. These various programs and associated data are executed and manipulated primarily by microprocessor 520.

10

15

Real Time Clock (RTC) 528

In the preferred embodiment, SPU 500 includes a real time clock circuit ("RTC") 528 that serves as a reliable, tamper resistant time base for the SPU. RTC 528 keeps track of time of

20

day and date (e.g., month, day and year) in the preferred embodiment, and thus may comprise a combination calendar and clock. A reliable time base is important for implementing time based usage metering methods, "time aged decryption keys," and other time based SPU functions.

The RTC 528 must receive power in order to operate. Optimally, the RTC 528 power source could comprise a small battery located within SPU 500 or other secure enclosure. However, the RTC 528 may employ a power source such as an externally located battery that is external to the SPU 500. Such an externally located battery may provide relatively uninterrupted power to RTC 528, and may also maintain as non-volatile at least a portion of the otherwise volatile RAM 534 within SPU 500.

In one implementation, electronic appliance power supply 659 is also used to power SPU 500. Using any external power supply as the only power source for RTC 528 may significantly reduce the usefulness of time based security techniques unless, at minimum, SPU 500 recognizes any interruption (or any material interruption) of the supply of external power, records such

interruption, and responds as may be appropriate such as disabling the ability of the SPU 500 to perform certain or all VDE processes. Recognizing a power interruption may, for example, be accomplished by employing a circuit which is activated by power failure. The power failure sensing circuit may power another circuit that includes associated logic for recording one or more power fail events. Capacitor discharge circuitry may provide the necessary temporary power to operate this logic. In addition or alternatively, SPU 500 may from time to time compare an output of RTC 528 to a clock output of a host electronic appliance 600, if available. In the event a discrepancy is detected, SPU 500 may respond as appropriate, including recording the discrepancy and/or disabling at least some portion of processes performed by SPU 500 under at least some circumstances.

If a power failure and/or RTC 528 discrepancy and/or other event indicates the possibility of tampering, SPU 500 may automatically destroy, or render inaccessible without privileged intervention, one or more portions of sensitive information it stores, such as execution related information and/or encryption key related information. To provide further SPU operation, such

destroyed information would have to be replaced by a VDE clearinghouse, administrator and/or distributor, as may be appropriate. This may be achieved by remotely downloading update and/or replacement data and/or code. In the event of a
5 disabling and/or destruction of processes and/or information as described above, the electronic appliance 600 may require a secure VDE communication with an administrator, clearinghouse, and/or distributor as appropriate in order to reinitialize the RTC 528. Some or all secure SPU 500 processes
10 may not operate until then.

It may be desirable to provide a mechanism for setting and/or synchronizing RTC 528. In the preferred embodiment, when communication occurs between VDE electronic appliance 600 and another VDE appliance, an output of RTC 528 may be
15 compared to a controlled RTC 528 output time under control of the party authorized to be "senior" and controlling. In the event of a discrepancy, appropriate action may be taken, including resetting the RTC 528 of the "junior" controlled participant in the
communication.

20

SPU Encrypt/Decrypt Engine 522

In the preferred embodiment, SPU encrypt/decrypt engine 522 provides special purpose hardware (e.g., a hardware state machine) for rapidly and efficiently encrypting and/or decrypting data. In some implementations, the encrypt/decrypt functions
5 may be performed instead by microprocessor 520 under software control, but providing special purpose encrypt/decrypt hardware engine 522 will, in general, provide increased performance. Microprocessor 520 may, if desired, comprise a combination of processor circuitry and dedicated encryption/decryption logic that
10 may be integrated together in the same circuitry layout so as to, for example, optimally share one or more circuit elements.

Generally, it is preferable that a computationally efficient but highly secure "bulk" encryption/decryption technique should
15 be used to protect most of the data and objects handled by SPU 500. It is preferable that an extremely secure encryption/decryption technique be used as an aspect of authenticating the identity of electronic appliances 600 that are establishing a communication channel and securing any
20 transferred permission, method, and administrative information. In the preferred embodiment, the encrypt/decrypt engine 522 includes both a symmetric key encryption/decryption circuit (e.g.

DES, Skipjack/Clipper, IDEA, RC-2, RC-4, etc.) and an
antisymmetric (asymmetric) or Public Key ("PK")
encryption/decryption circuit. The public/private key
encryption/decryption circuit is used principally as an aspect of
5 secure communications between an SPU 500 and VDE
administrators, or other electronic appliances 600, that is
between VDE secure subsystems. A symmetric
encryption/decryption circuit may be used for "bulk" encrypting
and decrypting most data stored in secondary storage 662 of
10 electronic appliance 600 in which SPU 500 resides. The
symmetric key encryption/decryption circuit may also be used for
encrypting and decrypting content stored within VDE objects
300.

15 DES or public/private key methods may be used for all
encryption functions. In alternate embodiments, encryption and
decryption methods other than the DES and public/private key
methods could be used for the various encryption related
functions. For instance, other types of symmetric
20 encryption/decryption techniques in which the same key is used
for encryption and decryption could be used in place of DES
encryption and decryption. The preferred embodiment can

support a plurality of decryption/encryption techniques using multiple dedicated circuits within encrypt/decrypt engine 522 and/or the processing arrangement within SPU 500.

5 **Pattern Matching Engine 524**

Optional pattern matching engine 524 may provide special purpose hardware for performing pattern matching functions.

One of the functions SPU 500 may perform is to validate/authenticate VDE objects 300 and other items.

10 Validation/authentication often involves comparing long data strings to determine whether they compare in a predetermined way. In addition, certain forms of usage (such as logical and/or physical (contiguous) relatedness of accessed elements) may require searching potentially long strings of data for certain bit
15 patterns or other significant pattern related metrics. Although pattern matching can be performed by SPU microprocessor 520 under software control, providing special purpose hardware pattern matching engine 524 may speed up the pattern matching process.

20

Compression/Decompression Engine 546

An optional compression/decompression engine 546 may be provided within an SPU 500 to, for example, compress and/or decompress content stored in, or released from, VDE objects 300. Compression/decompression engine 546 may implement one or
5 more compression algorithms using hardware circuitry to improve the performance of compression/decompression operations that would otherwise be performed by software operating on microprocessor 520, or outside SPU 500.

Decompression is important in the release of data such as video
10 and audio that is usually compressed before distribution and whose decompression speed is important. In some cases, information that is useful for usage monitoring purposes (such as record separators or other delimiters) is "hidden" under a compression layer that must be removed before this information
15 can be detected and used inside SPU 500.

Random Number Generator 542

Optional random number generator 542 may provide specialized hardware circuitry for generating random values (e.g.,
20 from inherently unpredictable physical processes such as quantum noise). Such random values are particularly useful for constructing encryption keys or unique identifiers, and for

initializing the generation of pseudo-random sequences. Random number generator 542 may produce values of any convenient length, including as small as a single bit per use. A random number of arbitrary size may be constructed by concatenating values produced by random number generator 542. A cryptographically strong pseudo-random sequence may be generated from a random key and seed generated with random number generator 542 and repeated encryption either with the encrypt/decrypt engine 522 or cryptographic algorithms in SPU 500. Such sequences may be used, for example, in private headers to frustrate efforts to determine an encryption key through cryptoanalysis.

Arithmetic Accelerator 544

An optional arithmetic accelerator 544 may be provided within an SPU 500 in the form of hardware circuitry that can rapidly perform mathematical calculations such as multiplication and exponentiation involving large numbers. These calculations can, for example, be requested by microprocessor 520 or encrypt/decrypt engine 522, to assist in the computations required for certain asymmetric encryption/decryption operations. Such arithmetic accelerators are well-known to those

skilled in the art. In some implementations, a separate arithmetic accelerator 544 may be omitted and any necessary calculations may be performed by microprocessor 520 under software control.

5

DMA Controller 526

DMA controller 526 controls information transfers over address/data bus 536 without requiring microprocessor 520 to process each individual data transfer. Typically, microprocessor 520 may write to DMA controller 526 target and destination addresses and the number of bytes to transfer, and DMA controller 526 may then automatically transfer a block of data between components of SPU 500 (e.g., from ROM 532 to RAM 534, between encrypt/decrypt engine 522 and RAM 534, between bus interface unit 530 and RAM 534, etc.). DMA controller 526 may have multiple channels to handle multiple transfers simultaneously. In some implementations, a separate DMA controller 526 may be omitted, and any necessary data movements may be performed by microprocessor 520 under software control.

10

15

20

Bus Interface Unit (BIU) 530

Bus interface unit (BIU) 530 communicates information between SPU 500 and the outside world across the security barrier 502. BIU 530 shown in Figure 9 plus appropriate driver software may comprise the "appliance link" 510 shown in Figure 6. Bus interface unit 530 may be modelled after a USART or PCI bus interface in the preferred embodiment. In this example, BIU 530 connects SPU 500 to electronic appliance system bus 653 shown in Figure 8. BIU 530 is designed to prevent unauthorized access to internal components within SPU 500 and their contents. It does this by only allowing signals associated with an SPU 500 to be processed by control programs running on microprocessor 520 and not supporting direct access to the internal elements of an SPU 500.

15 **Memory Management Unit 540**

Memory Management Unit (MMU) 540, if present, provides hardware support for memory management and virtual memory management functions. It may also provide heightened security by enforcing hardware compartmentalization of the secure execution space (e.g., to prevent a less trusted task from modifying a more trusted task). More details are provided below

in connection with a discussion of the architecture of a Secure Processing Environment ("SPE") 503 supported by SPU 500.

MMU 540 may also provide hardware-level support
5 functions related to memory management such as, for example,
address mapping.

SPU Memory Architecture

In the preferred embodiment, SPU 500 uses three general
10 kinds of memory:

- (1) internal ROM 532;
- (2) internal RAM 534; and
- (3) external memory (typically RAM and/or disk supplied
by a host electronic appliance).

15

The internal ROM 532 and RAM 534 within SPU 500
provide a secure operating environment and execution space.
Because of cost limitations, chip fabrication size, complexity and
other limitations, it may not be possible to provide sufficient
20 memory within SPU 500 to store all information that an SPU
needs to process in a secure manner. Due to the practical limits
on the amount of ROM 532 and RAM 534 that may be included

within SPU 500, SPU 500 may store information in memory external to it, and move this information into and out of its secure internal memory space on an as needed basis. In these cases, secure processing steps performed by an SPU typically

5 must be segmented into small, securely packaged elements that may be "paged in" and "paged out" of the limited available internal memory space. Memory external to an SPU 500 may not be secure. Since the external memory may not be secure, SPU 500 may encrypt and cryptographically seal code and other

10 information before storing it in external memory. Similarly, SPU 500 must typically decrypt code and other information obtained from external memory in encrypted form before processing (e.g., executing) based on it. In the preferred embodiment, there are two general approaches used to address potential memory

15 limitations in a SPU 500. In the first case, the small, securely packaged elements represent information contained in secure database 610. In the second case, such elements may represent protected (e.g., encrypted) virtual memory pages. Although virtual memory pages may correspond to information elements

20 stored in secure database 610, this is not required in this example of a SPU memory architecture.

The following is a more detailed discussion of each of these three SPU memory resources.

SPU Internal ROM

5 SPU 500 read only memory (ROM) 532 or comparable purpose device provides secure internal non-volatile storage for certain programs and other information. For example, ROM 532 may store "kernel" programs such as SPU control firmware 508 and, if desired, encryption key information and certain
10 fundamental "load modules." The "kernel" programs, load module information, and encryption key information enable the control of certain basic functions of the SPU 500. Those components that are at least in part dependent on device configuration (e.g., POST, memory allocation, and a dispatcher)
15 may be loaded in ROM 532 along with additional load modules that have been determined to be required for specific installations or applications.

In the preferred embodiment, ROM 532 may comprise a
20 combination of a masked ROM 532a and an EEPROM and/or equivalent "flash" memory 532b. EEPROM or flash memory 532b is used to store items that need to be updated and/or

initialized, such as for example, certain encryption keys. An
additional benefit of providing EEPROM and/or flash memory
532b is the ability to optimize any load modules and library
functions persistently stored within SPU 500 based on typical
5 usage at a specific site. Although these items could also be
stored in NVRAM 534b, EEPROM and/or flash memory 532b
may be more cost effective.

Masked ROM 532a may cost less than flash and/or
10 EEPROM 532b, and can be used to store permanent portions of
SPU software/firmware. Such permanent portions may include,
for example, code that interfaces to hardware elements such as
the RTC 528, encryption/decryption engine 522, interrupt
handlers, key generators, etc. Some of the operating system,
15 library calls, libraries, and many of the core services provided by
SPU 500 may also be in masked ROM 532a. In addition, some of
the more commonly used executables are also good candidates for
inclusion in masked ROM 532a. Items that need to be updated or
that need to disappear when power is removed from SPU 500
20 should not be stored in masked ROM 532a.

Under some circumstances, RAM 534a and/or NVRAM 534b (NVRAM 534b may, for example, be constantly powered conventional RAM) may perform at least part of the role of ROM 532.

5

SPU Internal RAM

SPU 500 general purpose RAM 534 provides, among other things, secure execution space for secure processes. In the preferred embodiment, RAM 534 is comprised of different types of RAM such as a combination of high-speed RAM 534a and an NVRAM ("non-volatile RAM") 534b. RAM 534a may be volatile, while NVRAM 534b is preferably battery backed or otherwise arranged so as to be non-volatile (i.e., it does not lose its contents when power is turned off).

10
15

High-speed RAM 534a stores active code to be executed and associated data structures.

20

NVRAM 534b preferably contains certain keys and summary values that are preloaded as part of an initialization process in which SPU 500 communicates with a VDE administrator, and may also store changeable or changing

information associated with the operation of SPU 500. For security reasons, certain highly sensitive information (e.g., certain load modules and certain encryption key related information such as internally generated private keys) needs to be loaded into or generated internally by SPU 500 from time to time but, once loaded or generated internally, should never leave the SPU. In this preferred embodiment, the SPU 500 non-volatile random access memory (NVRAM) 534b may be used for securely storing such highly sensitive information. NVRAM 534b is also used by SPU 500 to store data that may change frequently but which preferably should not be lost in a power down or power fail mode.

NVRAM 534b is preferably a flash memory array, but may in addition or alternatively be electrically erasable programmable read only memory (EEPROM), static RAM (SRAM), bubble memory, three dimensional holographic or other electro-optical memory, or the like, or any other writable (e.g., randomly accessible) non-volatile memory of sufficient speed and cost-effectiveness.

SPU External Memory

The SPU 500 can store certain information on memory devices external to the SPU. If available, electronic appliance 600 memory can also be used to support any device external portions of SPU 500 software. Certain advantages may be gained by
5 allowing the SPU 500 to use external memory. As one example, memory internal to SPU 500 may be reduced in size by using non-volatile read/write memory in the host electronic appliance 600 such as a non-volatile portion of RAM 656 and/or ROM 658.

10 Such external memory may be used to store SPU programs, data and/or other information. For example, a VDE control program may be, at least in part, loaded into the memory and communicated to and decrypted within SPU 500 prior to execution. Such control programs may be re-encrypted and
15 communicated back to external memory where they may be stored for later execution by SPU 500. "Kernel" programs and/or some or all of the non-kernel "load modules" may be stored by SPU 500 in memory external to it. Since a secure database 610 may be relatively large, SPU 500 can store some or all of secure
20 database 610 in external memory and call portions into the SPU 500 as needed.

As mentioned above, memory external to SPU 500 may not be secure. Therefore, when security is required, SPU 500 must encrypt secure information before writing it to external memory, and decrypt secure information read from external memory
5 before using it. Inasmuch as the encryption layer relies on secure processes and information (e.g., encryption algorithms and keys) present within SPU 500, the encryption layer effectively "extends" the SPU security barrier 502 to protect information the SPU 500 stores in memory external to it.

10

SPU 500 can use a wide variety of different types of external memory. For example, external memory may comprise electronic appliance secondary storage 652 such as a disk; external EEPROM or flash memory 658; and/or external RAM
15 656. External RAM 656 may comprise an external nonvolatile (e.g. constantly powered) RAM and/or cache RAM.

Using external RAM local to SPU 500 can significantly improve access times to information stored externally to an SPU.
20 For example, external RAM may be used:
C to buffer memory image pages and data structures prior to their storage in flash memory or on an external hard disk

(assuming transfer to flash or hard disk can occur in significant power or system failure cases);

C provide encryption and decryption buffers for data being released from VDE objects 300.

5 C to cache "swap blocks" and VDE data structures currently in use as an aspect of providing a secure virtual memory environment for SPU 500.

C to cache other information in order to, for example, reduce frequency of access by an SPU to secondary storage 652 and/or for other reasons.

10

Dual ported external RAM can be particularly effective in improving SPU 500 performance, since it can decrease the data movement overhead of the SPU bus interface unit 530 and SPU microprocessor 520.

15

Using external flash memory local to SPU 500 can be used to significantly improve access times to virtually all data structures. Since most available flash storage devices have limited write lifetimes, flash storage needs to take into account the number of writes that will occur during the lifetime of the flash memory. Hence, flash storage of frequently written temporary items is not recommended. If external RAM is non-

20

volatile, then transfer to flash (or hard disk) may not be necessary.

5 External memory used by SPU 500 may include two categories:

- C external memory dedicated to SPU 500, and
- C memory shared with electronic appliance 600.

10 For some VDE implementations, sharing memory (e.g., electronic appliance RAM 656, ROM 658 and/or secondary storage 652) with CPU 654 or other elements of an electronic appliance 600 may be the most cost effective way to store VDE secure database management files 610 and information that needs to be stored external to SPU 500. A host system hard disk secondary memory 652 used for general purpose file storage can, 15 for example, also be used to store VDE management files 610. SPU 500 may be given exclusive access to the external memory (e.g., over a local bus high speed connection provided by BIU 530). Both dedicated and shared external memory may be 20 provided.

* * * * *

The hardware configuration of an example of electronic appliance 600 has been described above. The following section describes an example of the software architecture of electronic appliance 600 provided by the preferred embodiment, including the structure and operation of preferred embodiment "Rights Operating System" ("ROS") 602.

Rights Operating System 602

Rights Operating System ("ROS") 602 in the preferred embodiment is a compact, secure, event-driven, services-based, "component" oriented, distributed multiprocessing operating system environment that integrates VDE information security control information, components and protocols with traditional operating system concepts. Like traditional operating systems, ROS 602 provided by the preferred embodiment is a piece of software that manages hardware resources of a computer system and extends management functions to input and/or output devices, including communications devices. Also like traditional operating systems, preferred embodiment ROS 602 provides a coherent set of basic functions and abstraction layers for hiding the differences between, and many of the detailed complexities of, particular hardware implementations. In addition to these

characteristics found in many or most operating systems, ROS 602 provides secure VDE transaction management and other advantageous features not found in other operating systems. The following is a non-exhaustive list of some of the advantageous features provided by ROS 602 in the preferred embodiment:

- Standardized interface provides coherent set of basic functions
- C simplifies programming
 - C the same application can run on many different platforms
- 10 Event driven
- C eases functional decomposition
 - C extendible
 - C accommodates state transition and/or process oriented events
- 15 C simplifies task management
- C simplifies inter-process communications
- Services based
- C allows simplified and transparent scalability
 - C simplifies multiprocessor support
- 20 C hides machine dependencies
- C eases network management and support
- Component Based Architecture

- C processing based on independently deliverable secure components
- C component model of processing control allows different sequential steps that are reconfigurable based on requirements
- 5 C components can be added, deleted or modified (subject to permissioning)
- C full control information over pre-defined and user-defined application events
- 10 C events can be individually controlled with independent executables

Secure

- C secure communications
- C secure control functions
- 15 C secure virtual memory management
- C information control structures protected from exposure
- C data elements are validated, correlated and access controlled
- C components are encrypted and validated independently
- 20 C components are tightly correlated to prevent unauthorized use of elements

- C control structures and secured executables are validated prior to use to protect against tampering
- C integrates security considerations at the I/O level
- C provides on-the-fly decryption of information at release
5 time
- C enables a secure commercial transaction network
- C flexible key management features

Scalaeble

- C highly scalaeble across many different platforms
- 10 C supports concurrent processing in a multiprocessor environment
- C supports multiple cooperating processors
- C any number of host or security processors can be supported
- C control structures and kernel are easily portable to various
15 host platforms and to different processors within a target platform without recompilation
- C supports remote processing
- C Remote Procedure Calls may be used for internal OS communications

20 Highly Integratable

- C can be highly integrated with host platforms as an additional operating system layer

- C permits non-secure storage of secured components and information using an OS layer "on top of" traditional OS platforms
- 5 C can be seamlessly integrated with a host operating system to provide a common usage paradigm for transaction management and content access
- C integration may take many forms: operating system layers for desktops (e.g., DOS, Windows, Macintosh); device drivers and operating system interfaces for network services (e.g., Unix and Netware); and dedicated component drivers for "low end" set tops are a few of many examples
- 10 C can be integrated in traditional and real time operating systems

Distributed

- 15 C provides distribution of control information and reciprocal control information and mechanisms
- C supports conditional execution of controlled processes within any VDE node in a distributed, asynchronous arrangement
- 20 C controlled delegation of rights in a distributed environment
- C supports chains of handling and control

- C management environment for distributed, occasionally connected but otherwise asynchronous networked database
- C real time and time independent data management
- C supports "agent" processes

5 Transparent

- C can be seamlessly integrated into existing operating systems
- C can support applications not specifically written to use it

Network friendly

- 10 C internal OS structures may use RPCs to distribute processing
- C subnets may seamlessly operate as a single node or independently

15 **General Background Regarding Operating Systems**

An "operating system" provides a control mechanism for organizing computer system resources that allows programmers to create applications for computer systems more easily. An operating system does this by providing commonly used functions, and by helping to ensure compatibility between different computer hardware and architectures (which may, for example, be manufactured by different vendors). Operating

20

systems also enable computer "peripheral device" manufacturers to far more easily supply compatible equipment to computer manufacturers and users.

5 Computer systems are usually made up of several different hardware components. These hardware components include, for example:

a central processing unit (CPU) for executing instructions;

10 an array of main memory cells (e.g., "RAM" or "ROM") for storing instructions for execution and data acted upon or parameterizing those instructions; and

15 one or more secondary storage devices (e.g., hard disk drive, floppy disk drive, CD-ROM drive, tape reader, card reader, or "flash" memory) organized to reflect named elements (a "file system") for storing images of main memory cells.

20 Most computer systems also include input/output devices such as keyboards, mice, video systems, printers, scanners and communications devices.

To organize the CPU's execution capabilities with available RAM, ROM and secondary storage devices, and to provide commonly used functions for use by programmers, a piece of software called an "operating system" is usually included with the other components. Typically, this piece of software is designed to begin executing after power is applied to the computer system and hardware diagnostics are completed. Thereafter, all use of the CPU, main memory and secondary memory devices is normally managed by this "operating system" software. Most computer operating systems also typically include a mechanism for extending their management functions to I/O and other peripheral devices, including commonly used functions associated with these devices.

By managing the CPU, memory and peripheral devices through the operating system, a coherent set of basic functions and abstraction layers for hiding hardware details allows programmers to more easily create sophisticated applications. In addition, managing the computer's hardware resources with an operating system allows many differences in design and equipment requirements between different manufacturers to be hidden. Furthermore, applications can be more easily shared

with other computer users who have the same operating system, with significantly less work to support different manufacturers' base hardware and peripheral devices.

5 **ROS 602 is an Operating System Providing Significant Advantages**

ROS 602 is an "operating system." It manages the resources of electronic appliance 600, and provides a commonly used set of functions for programmers writing applications 608 for the electronic appliance. ROS 602 in the preferred
10 embodiment manages the hardware (e.g., CPU(s), memory(ies), secure RTC(s), and encrypt/decrypt engines) within SPU 500. ROS may also manage the hardware (e.g., CPU(s) and memory(ies)) within one or more general purpose processors
15 within electronic appliance 600. ROS 602 also manages other electronic appliance hardware resources, such as peripheral devices attached to an electronic appliance. For example, referring to Figure 7, ROS 602 may manage keyboard 612, display 614, modem 618, disk drive 620, printer 622, scanner 624.
20 ROS 602 may also manage secure database 610 and a storage device (e.g., "secondary storage" 652) used to store secure database 610.

ROS 602 supports multiple processors. ROS 602 in the preferred embodiment supports any number of local and/or remote processors. Supported processors may include at least two types: one or more electronic appliance processors 654, 5 and/or one or more SPUs 500. A host processor CPU 654 may provide storage, database, and communications services. SPU 500 may provide cryptographic and secured process execution services. Diverse control and execution structures supported by ROS 602 may require that processing of control information occur 10 within a controllable execution space -- this controllable execution space may be provided by SPU 500. Additional host and/or SPU processors may increase efficiencies and/or capabilities. ROS 602 may access, coordinate and/or manage further processors remote to an electronic appliance 600 (e.g., via 15 network or other communications link) to provide additional processor resources and/or capabilities.

ROS 602 is services based. The ROS services provided using a host processor 654 and/or a secure processor (SPU 500) 20 are linked in the preferred embodiment using a "Remote Procedure Call" ("RPC") internal processing request structure. Cooperating processors may request interprocess services using a

RPC mechanism, which is minimally time dependent and can be distributed over cooperating processors on a network of hosts. The multi-processor architecture provided by ROS 602 is easily extensible to support any number of host or security processors.

5 This extensibility supports high levels of scalability. Services also allow functions to be implemented differently on different equipment. For example, a small appliance that typically has low levels of usage by one user may implement a database service using very different techniques than a very large appliance with

10 high levels of usage by many users. This is another aspect of scalability.

ROS 602 provides a distributed processing environment.

For example, it permits information and control structures to

15 automatically, securely pass between sites as required to fulfill a user's requests. Communications between VDE nodes under the distributed processing features of ROS 602 may include interprocess service requests as discussed above. ROS 602 supports conditional and/or state dependent execution of

20 controlled processors within any VDE node. The location that the process executes and the control structures used may be

locally resident, remotely accessible, or carried along by the process to support execution on a remote system.

ROS 602 provides distribution of control information, including for example the distribution of control structures required to permit "agents" to operate in remote environments. Thus, ROS 602 provides facilities for passing execution and/or information control as part of emerging requirements for "agent" processes.

If desired, ROS 602 may independently distribute control information over very low bandwidth connections that may or may not be "real time" connections. ROS 602 provided by the preferred embodiment is "network friendly," and can be implemented with any level of networking protocol. Some examples include e-mail and direct connection at approximately "Layer 5" of the ISO model.

The ROS 602 distribution process (and the associated auditing of distributed information) is a controlled event that itself uses such control structures. This "reflective" distributed processing mechanism permits ROS 602 to securely distribute

rights and permissions in a controlled manner, and effectively
restrict the characteristics of use of information content. The
controlled delegation of rights in a distributed environment and
the secure processing techniques used by ROS 602 to support this
5 approach provide significant advantages.

Certain control mechanisms within ROS 602 are
"reciprocal." Reciprocal control mechanisms place one or more
control components at one or more locations that interact with
10 one or more components at the same or other locations in a
controlled way. For example, a usage control associated with
object content at a user's location may have a reciprocal control at
a distributor's location that governs distribution of the usage
control, auditing of the usage control, and logic to process user
15 requests associated with the usage control. A usage control at a
user's location (in addition to controlling one or more aspects of
usage) may prepare audits for a distributor and format requests
associated with the usage control for processing by a distributor.
Processes at either end of a reciprocal control may be further
20 controlled by other processes (e.g., a distributor may be limited by
a budget for the number of usage control mechanisms they may
produce). Reciprocal control mechanisms may extend over many

5 sites and many levels (e.g., a creator to a distributor to a user)
and may take any relationship into account (e.g.,
creator/distributor, distributor/user, user/user, user/creator,
user/creator/distributor, etc.) Reciprocal control mechanisms
have many uses in VDE 100 in representing relationships and
agreements in a distributed environment.

10 ROS 602 is scalable. Many portions of ROS 602 control
structures and kernel(s) are easily portable to various host
platforms without recompilation. Any control structure may be
distributed (or redistributed) if a granting authority permits this
type of activity. The executable references within ROS 602 are
portable within a target platform. Different instances of ROS 602
may execute the references using different resources. For
15 example, one instance of ROS 602 may perform a task using an
SPU 500, while another instance of ROS 602 might perform the
same task using a host processing environment running in
protected memory that is emulating an SPU in software. ROS
602 control information is similarly portable; in many cases the
20 event processing structures may be passed between machines
and host platforms as easily as between cooperative processors in
a single computer. Appliances with different levels of usage

and/or resources available for ROS 602 functions may implement those functions in very different ways. Some services may be omitted entirely if insufficient resources exist. As described elsewhere, ROS 602 "knows" what services are available, and how to proceed based on any given event. Not all events may be processable if resources are missing or inadequate.

ROS 602 is component based. Much of the functionality provided by ROS 602 in the preferred embodiment may be based on "components" that can be securely, independently deliverable, replaceable and capable of being modified (e.g., under appropriately secure conditions and authorizations). Moreover, the "components" may themselves be made of independently deliverable elements. ROS 602 may assemble these elements together (using a construct provided by the preferred embodiment called a "channel") at execution time. For example, a "load module" for execution by SPU 500 may reference one or more "method cores," method parameters and other associated data structures that ROS 602 may collect and assemble together to perform a task such as billing or metering. Different users may have different combinations of elements, and some of the elements may be customizable by users with appropriate

authorization. This increases flexibility, allows elements to be reused, and has other advantages.

ROS 602 is highly secure. ROS 602 provides mechanisms
5 to protect information control structures from exposure by end
users and conduit hosts. ROS 602 can protect information, VDE
control structures and control executables using strong
encryption and validation mechanisms. These encryption and
validation mechanisms are designed to make them highly
10 resistant to undetected tampering. ROS 602 encrypts
information stored on secondary storage device(s) 652 to inhibit
tampering. ROS 602 also separately encrypts and validates its
various components. ROS 602 correlates control and data
structure components to prevent unauthorized use of elements.
15 These features permit ROS 602 to independently distribute
elements, and also allows integration of VDE functions 604 with
non-secure "other" OS functions 606.

ROS 602 provided by the preferred embodiment extends
20 conventional capabilities such as, for example, Access Control
List (ACL) structures, to user and process defined events,
including state transitions. ROS 602 may provide full control

information over pre-defined and user-defined application events. These control mechanisms include "go/no-go" permissions, and also include optional event-specific executables that permit complete flexibility in the processing and/or controlling of events.

5 This structure permits events to be individually controlled so that, for example, metering and budgeting may be provided using independent executables. For example, ROS 602 extends ACL structures to control arbitrary granularity of information. Traditional operating systems provide static "go-no go" control
10 mechanisms at a file or resource level; ROS 602 extends the control concept in a general way from the largest to the smallest sub-element using a flexible control structure. ROS 602 can, for example, control the printing of a single paragraph out of a document file.

15

ROS 602 provided by the preferred embodiment permits secure modification and update of control information governing each component. The control information may be provided in a template format such as method options to an end-user. An
20 end-user may then customize the actual control information used within guidelines provided by a distributor or content creator. Modification and update of existing control structures is

preferably also a controllable event subject to auditing and control information.

5 ROS 602 provided by the preferred embodiment validates control structures and secured executables prior to use. This validation provides assurance that control structures and executables have not been tampered with by end-users. The validation also permits ROS 602 to securely implement components that include fragments of files and other operating
10 system structures. ROS 602 provided by the preferred embodiment integrates security considerations at the operating system I/O level (which is below the access level), and provides "on-the-fly" decryption of information at release time. These features permit non-secure storage of ROS 602 secured
15 components and information using an OS layer "on top of" traditional operating system platforms.

ROS 602 is highly integratable with host platforms as an additional operating system layer. Thus, ROS 602 may be
20 created by "adding on" to existing operating systems. This involves hooking VDE "add ons" to the host operating system at the device driver and network interface levels. Alternatively,

ROS 602 may comprise a wholly new operating system that integrates both VDE functions and other operating system functions.

5 Indeed, there are at least three general approaches to integrating VDE functions into a new operating system, potentially based on an existing operating system, to create a Rights Operating System 602 including:

- 10 (1) Redesign the operating system based on VDE transaction management requirements;
- (2) Compile VDE API functions into an existing operating systems; and
- (3) Integrate a VDE Interpreter into an existing operating system.

15 The first approach could be most effectively applied when a new operating system is being designed, or if a significant upgrade to an existing operating system is planned. The transaction management and security requirements provided by
20 the VDE functions could be added to the design requirements list for the design of a new operating system that provides, in an optimally efficient manner, an integration of "traditional"

operating system capabilities and VDE capabilities. For example, the engineers responsible for the design of the new version or instance of an operating system would include the requirements of VDE metering/transaction management in addition to other requirements (if any) that they use to form their design approach, specifications, and actual implementations. This approach could lead to a "seamless" integration of VDE functions and capabilities by threading metering/transaction management functionality throughout the system design and implementation.

10

The second approach would involve taking an existing set of API (Application Programmer Interface) functions, and incorporating references in the operating system code to VDE function calls. This is similar to the way that the current Windows operating system is integrated with DOS, wherein DOS serves as both the launch point and as a significant portion of the kernel underpinning of the Windows operating system. This approach would be also provide a high degree of "seamless" integration (although not quite as "seamless" as the first approach). The benefits of this approach include the possibility that the incorporation of metering/transaction management functionality into the new version or instance of an operating

15
20

system may be accomplished with lower cost (by making use of the existing code embodied in an API, and also using the design implications of the API functional approach to influence the design of the elements into which the metering/transaction management functionality is incorporated).

The third approach is distinct from the first two in that it does not incorporate VDE functionality associated with metering/transaction management and data security directly into the operating system code, but instead adds a new generalized capability to the operating system for executing metering/transaction management functionality. In this case, an interpreter including metering/transaction management functions would be integrated with other operating system code in a "stand alone" mode. This interpreter might take scripts or other inputs to determine what metering/transaction management functions should be performed, and in what order and under which circumstances or conditions they should be performed.

Instead of (or in addition to) integrating VDE functions into/with an electronic appliance operating system, it would be

possible to provide certain VDE functionality available as an application running on a conventional operating system.

ROS Software Architecture

5 Figure 10 is a block diagram of one example of a software structure/architecture for Rights Operating System ("ROS") 602 provided by the preferred embodiment. In this example, ROS 602 includes an operating system ("OS") "core" 679, a user Application Program Interface ("API") 682, a "redirector" 684, an
10 "intercept" 692, a User Notification/Exception Interface 686, and a file system 687. ROS 602 in this example also includes one or more Host Event Processing Environments ("HPEs") 655 and/or one or more Secure Event Processing Environments ("SPEs") 503 (these environments may be generically referred to as "Protected
15 Processing Environments" 650).

 HPE(s) 655 and SPE(s) 503 are self-contained computing and processing environments that may include their own operating system kernel 688 including code and data processing
20 resources. A given electronic appliance 600 may include any number of SPE(s) 503 and/or any number of HPE(s) 655. HPE(s) 655 and SPE(s) 503 may process information in a secure way,

and provide secure processing support for ROS 602. For example, they may each perform secure processing based on one or more VDE component assemblies 690, and they may each offer secure processing services to OS kernel 680.

5

In the preferred embodiment, SPE 503 is a secure processing environment provided at least in part by an SPU 500. Thus, SPU 500 provides the hardware tamper-resistant barrier 503 surrounding SPE 503. SPE 503 provided by the preferred embodiment is preferably:

10

- C small and compact
- C loadable into resource constrained environments such as for example minimally configured SPUs 500
- C dynamically updatable
- C extensible by authorized users
- C integratable into object or procedural environments
- C secure.

15

20

In the preferred embodiment, HPE 655 is a secure processing environment supported by a processor other than an

SPU, such as for example an electronic appliance CPU 654
general-purpose microprocessor or other processing system or
device. In the preferred embodiment, HPE 655 may be
considered to "emulate" an SPU 500 in the sense that it may use
5 software to provide some or all of the processing resources
provided in hardware and/or firmware by an SPU. HPE 655 in
one preferred embodiment of the present invention is full-
featured and fully compatible with SPE 503—that is, HPE 655
can handle each and every service call SPE 503 can handle such
10 that the SPE and the HPE are "plug compatible" from an outside
interface standpoint (with the exception that the HPE may not
provide as much security as the SPE).

HPEs 655 may be provided in two types: secure and not
15 secure. For example, it may be desirable to provide non-secure
versions of HPE 655 to allow electronic appliance 600 to
efficiently run non-sensitive VDE tasks using the full resources of
a fast general purpose processor or computer. Such non-secure
versions of HPE 655 may run under supervision of an instance of
20 ROS 602 that also includes an SPE 503. In this way, ROS 602
may run all secure processes within SPE 503, and only use HPE
655 for processes that do not require security but that may

require (or run more efficiently) under potentially greater resources provided by a general purpose computer or processor supporting HPE 655. Non-secure and secure HPE 655 may operate together with a secure SPE 503.

5

HPEs 655 may (as shown in Figure 10) be provided with a software-based tamper resistant barrier 674 that makes them more secure. Such a software-based tamper resistant barrier 674 may be created by software executing on general-purpose CPU
10 654. Such a "secure" HPE 655 can be used by ROS 602 to execute processes that, while still needing security, may not require the degree of security provided by SPU 500. This can be especially beneficial in architectures providing both an SPE 503 and an HPE 655. The SPU 502 may be used to perform all truly
15 secure processing, whereas one or more HPEs 655 may be used to provide additional secure (albeit possibly less secure than the SPE) processing using host processor or other general purpose resources that may be available within an electronic appliance
600. Any service may be provided by such a secure HPE 655. In
20 the preferred embodiment, certain aspects of "channel processing" appears to be a candidate that could be readily exported from SPE 503 to HPE 655.

The software-based tamper resistant barrier 674 provided by HPE 655 may be provided, for example, by: introducing time checks and/or code modifications to complicate the process of stepping through code comprising a portion of kernel 688a and/or a portion of component assemblies 690 using a debugger; using a map of defects on a storage device (e.g., a hard disk, memory card, etc.) to form internal test values to impede moving and/or copying HPE 655 to other electronic appliances 600; using kernel code that contains false branches and other complications in flow of control to disguise internal processes to some degree from disassembly or other efforts to discover details of processes; using "self-generating" code (based on the output of a co-sine transform, for example) such that detailed and/or complete instruction sequences are not stored explicitly on storage devices and/or in active memory but rather are generated as needed; using code that "shuffles" memory locations used for data values based on operational parameters to complicate efforts to manipulate such values; using any software and/or hardware memory management resources of electronic appliance 600 to "protect" the operation of HPE 655 from other processes, functions, etc. Although such a software-based tamper resistant barrier 674 may provide a fair degree of security, it typically will not be as

secure as the hardware-based tamper resistant barrier 502 provided (at least in part) by SPU 500. Because security may be better/more effectively enforced with the assistance of hardware security features such as those provided by SPU 500 (and

5 because of other factors such as increased performance provided by special purpose circuitry within SPU 500), at least one SPE 503 is preferred for many or most higher security applications. However, in applications where lesser security can be tolerated and/or the cost of an SPU 500 cannot be tolerated, the SPE 503

10 may be omitted and all secure processing may instead be performed by one or more secure HPEs 655 executing on general-purpose CPUs 654. Some VDE processes may not be allowed to proceed on reduced-security electronic appliances of this type if insufficient security is provided for the particular

15 process involved.

Only those processes that execute completely within SPEs 503 (and in some cases, HPEs 655) may be considered to be truly secure. Memory and other resources external to SPE 503 and

20 HPEs 655 used to store and/or process code and/or data to be used in secure processes should only receive and handle that information in encrypted form unless SPE 503/HPE 655 can

protect secure process code and/or data from non-secure processes.

OS "core" 679 in the preferred embodiment includes a
5 kernel 680, an RPC manager 732, and an "object switch" 734.
API 682, HPE 655 and SPE 503 may communicate "event"
messages with one another via OS "core" 679. They may also
communicate messages directly with one another without
messages going through OS "core" 679.

10

Kernel 680 may manage the hardware of an electronic
appliance 600. For example, it may provide appropriate drivers
and hardware managers for interacting with input/output and/or
peripheral devices such as keyboard 612, display 614, other
15 devices such as a "mouse" pointing device and speech recognizer
613, modem 618, printer 622, and an adapter for network 672.
Kernel 680 may also be responsible for initially loading the
remainder of ROS 602, and may manage the various ROS tasks
(and associated underlying hardware resources) during
20 execution. OS kernel 680 may also manage and access secure
database 610 and file system 687. OS kernel 680 also provides

execution services for applications 608a(1), 608a(2), etc. and other applications.

5 RPC manager 732 performs messaging routing and resource management/integration for ROS 680. It receives and routes "calls" from/to API 682, HPE 655 and SPE 503, for example.

10 Object switch 734 may manage construction, deconstruction and other manipulation of VDE objects 300.

15 User Notification/Exception Interface 686 in the preferred embodiment (which may be considered part of API 682 or another application coupled to the API) provides "pop up" windows/displays on display 614. This allows ROS 602 to communicate directly with a user without having to pass information to be communicated through applications 608. For applications that are not "VDE aware," user notification/exception interface 686 may provide communications
20 between ROS 602 and the user.

API 682 in the preferred embodiment provides a standardized, documented software interface to applications 608. In part, API 682 may translate operating system "calls" generated by applications 608 into Remote Procedure Calls ("RPCs") specifying "events." RPC manager 732 may route these RPCs to kernel 680 or elsewhere (e.g., to HPE(s) 655 and/or SPE(s) 503, or to remote electronic appliances 600, processors, or VDE participants) for processing. The API 682 may also service RPC requests by passing them to applications 608 that register to receive and process specific requests.

API 682 provides an "Applications Programming Interface" that is preferably standardized and documented. It provides a concise set of function calls an application program can use to access services provided by ROS 602. In at least one preferred example, API 682 will include two parts: an application program interface to VDE functions 604; and an application program interface to other OS functions 606. These parts may be interwoven into the same software, or they may be provided as two or more discrete pieces of software (for example).

Some applications, such as application 608a(1) shown in Figure 11, may be "VDE aware" and may therefore directly access both of these parts of API 682. Figure 11A shows an example of this. A "VDE aware" application may, for example, include explicit calls to ROS 602 requesting the creation of new VDE objects 300, metering usage of VDE objects, storing information in VDE-protected form, etc. Thus, a "VDE aware" application can initiate (and, in some examples, enhance and/or extend) VDE functionality provided by ROS 602. In addition, "VDE aware" applications may provide a more direct interface between a user and ROS 602 (e.g., by suppressing or otherwise dispensing with "pop up" displays otherwise provided by user notification/exception interface 686 and instead providing a more "seamless" interface that integrates application and ROS messages).

Other applications, such as application 608b shown in Figure 11B, may not be "VDE Aware" and therefore may not "know" how to directly access an interface to VDE functions 604 provided by API 682. To provide for this, ROS 602 may include a "redirector" 684 that allows such "non-VDE aware" applications 608(b) to access VDE objects 300 and functions 604. Redirector

684, in the preferred embodiment, translates OS calls directed to the "other OS functions" 606 into calls to the "VDE functions" 604. As one simple example, redirector 684 may intercept a "file open" call from application 608(b), determine whether the file to be opened is contained within a VDE container 300, and if it is, generate appropriate VDE function call(s) to file system 687 to open the VDE container (and potentially generate events to HPE 655 and/or SPE 503 to determine the name(s) of file(s) that may be stored in a VDE object 300, establish a control structure associated with a VDE object 300, perform a registration for a VDE object 300, etc.). Without redirector 684 in this example, a non-VDE aware application such as 608b could access only the part of API 682 that provides an interface to other OS functions 606, and therefore could not access any VDE functions.

15

This "translation" feature of redirector 684 provides "transparency." It allows VDE functions to be provided to the application 608(b) in a "transparent" way without requiring the application to become involved in the complexity and details associated with generating the one or more calls to VDE functions 604. This aspect of the "transparency" features of ROS 602 has at least two important advantages:

20

- (a) it allows applications not written specifically for VDE functions 604 ("non-VDE aware applications") to nevertheless access critical VDE functions; and
- (b) it reduces the complexity of the interface between an application and ROS 602.

5 Since the second advantage (reducing complexity) makes it easier for an application creator to produce applications, even "VDE aware" applications 608a(2) may be designed so that some calls invoking VDE functions 604 are requested at the level of an

10 "other OS functions" call and then "translated" by redirector 684 into a VDE function call (in this sense, redirector 684 may be considered a part of API 682). Figure 11C shows an example of this. Other calls invoking VDE functions 604 may be passed directly without translation by redirector 684.

15 Referring again to Figure 10, ROS 620 may also include an "interceptor" 692 that transmits and/or receives one or more real time data feeds 694 (this may be provided over cable(s) 628 for example), and routes one or more such data feeds appropriately

20 while providing "translation" functions for real time data sent and/or received by electronic appliance 600 to allow "transparency" for this type of information analogous to the

transparency provided by redirector 684 (and/or it may generate one or more real time data feeds).

Secure ROS Components and Component Assemblies

5 As discussed above, ROS 602 in the preferred embodiment is a component-based architecture. ROS VDE functions 604 may be based on segmented, independently loadable executable "component assemblies" 690. These component assemblies 690 are independently securely deliverable. The component
10 assemblies 690 provided by the preferred embodiment comprise code and data elements that are themselves independently deliverable. Thus, each component assembly 690 provided by the preferred embodiment is comprised of independently securely deliverable elements which may be communicated using VDE
15 secure communication techniques, between VDE secure subsystems.

 These component assemblies 690 are the basic functional unit provided by ROS 602. The component assemblies 690 are
20 executed to perform operating system or application tasks. Thus, some component assemblies 690 may be considered to be part of the ROS operating system 602, while other component

assemblies may be considered to be "applications" that run under the support of the operating system. As with any system incorporating "applications" and "operating systems," the boundary between these aspects of an overall system can be
5 ambiguous. For example, commonly used "application" functions (such as determining the structure and/or other attributes of a content container) may be incorporated into an operating system. Furthermore, "operating system" functions (such as task management, or memory allocation) may be modified and/or
10 replaced by an application. A common thread in the preferred embodiment's ROS 602 is that component assemblies 690 provide functions needed for a user to fulfill her intended activities, some of which may be "application-like" and some of which may be "operating system-like."

15

Components 690 are preferably designed to be easily separable and individually loadable. ROS 602 assembles these elements together into an executable component assembly 690 prior to loading and executing the component assembly (e.g., in a
20 secure operating environment such as SPE 503 and/or HPE 655). ROS 602 provides an element identification and referencing mechanism that includes information necessary to automatically

assemble elements into a component assembly 690 in a secure manner prior to, and/or during, execution.

ROS 602 application structures and control parameters
5 used to form component assemblies 690 can be provided by
different parties. Because the components forming component
assemblies 690 are independently securely deliverable, they may
be delivered at different times and/or by different parties
("delivery" may take place within a local VDE secure subsystem,
10 that is submission through the use of such a secure subsystem of
control information by a chain of content control information
handling participant for the preparation of a modified control
information set constitutes independent, secure delivery). For
example, a content creator can produce a ROS 602 application
15 that defines the circumstances required for licensing content
contained within a VDE object 300. This application may
reference structures provided by other parties. Such references
might, for example, take the form of a control path that uses
content creator structures to meter user activities; and structures
20 created/owned by a financial provider to handle financial parts of
a content distribution transaction (e.g., defining a credit budget
that must be present in a control structure to establish

creditworthiness, audit processes which must be performed by the licensee, etc.). As another example, a distributor may give one user more favorable pricing than another user by delivering different data elements defining pricing to different users. This
5 attribute of supporting multiple party securely, independently deliverable control information is fundamental to enabling electronic commerce, that is, defining of a content and/or appliance control information set that represents the requirements of a collection of independent parties such as
10 content creators, other content providers, financial service providers, and/or users.

In the preferred embodiment, ROS 602 assembles securely independently deliverable elements into a component assembly
15 690 based in part on context parameters (e.g., object, user). Thus, for example, ROS 602 may securely assemble different elements together to form different component assemblies 690 for different users performing the same task on the same VDE object
300. Similarly, ROS 602 may assemble differing element sets
20 which may include, that is reuse, one or more of the same components to form different component assemblies 690 for the

same user performing the same task on different VDE objects
300.

5 The component assembly organization provided by ROS
602 is "recursive" in that a component assembly 690 may
comprise one or more component "subassemblies" that are
themselves independently loadable and executable component
assemblies 690. These component "subassemblies" may, in turn,
be made of one or more component "sub-sub-assemblies." In the
10 general case, a component assembly 690 may include N levels of
component subassemblies.

 Thus, for example, a component assembly 690(k) that may
includes a component subassembly 690(k + 1). Component
15 subassembly 690(k + 1), in turn, may include a component sub-
sub-assembly 690(3), ... and so on to N-level subassembly 690(k +
N). The ability of ROS 602 to build component assemblies 690
out of other component assemblies provides great advantages in
terms of, for example, code/data reusability, and the ability to
20 allow different parties to manage different parts of an overall
component.

Each component assembly 690 in the preferred embodiment is made of distinct components. Figures 11D-11H are abstract depictions of various distinct components that may be assembled to form a component assembly 690(k) showing
5 Figure 11I. These same components can be combined in different ways (e.g., with more or less components) to form different component assemblies 690 providing completely different functional behavior. Figure 11J is an abstract depiction of the same components being put together in a different way (e.g., with
10 additional components) to form a different component assembly 690(j). The component assemblies 690(k) and 690(j) each include a common feature 691 that interlocks with a "channel" 594 defined by ROS 602. This "channel" 594 assembles component assemblies 690 and interfaces them with the (rest of) ROS 602.

15

ROS 602 generates component assemblies 690 in a secure manner. As shown graphically in Figures 11I and 11J, the different elements comprising a component assembly 690 may be "interlocking" in the sense that they can only go together in ways
20 that are intended by the VDE participants who created the elements and/or specified the component assemblies. ROS 602 includes security protections that can prevent an unauthorized

person from modifying elements, and also prevent an unauthorized person from substituting elements. One can picture an unauthorized person making a new element having the same "shape" as the one of the elements shown in Figures 11D-11H, and then attempting to substitute the new element in place of the original element. Suppose one of the elements shown in Figure 11H establishes the price for using content within a VDE object 300. If an unauthorized person could substitute her own "price" element for the price element intended by the VDE content distributor, then the person could establish a price of zero instead of the price the content distributor intended to charge. Similarly, if the element establishes an electronic credit card, then an ability to substitute a different element could have disastrous consequences in terms of allowing a person to charge her usage to someone else's (or a non-existent) credit card. These are merely a few simple examples demonstrating the importance of ROS 602 ensuring that certain component assemblies 690 are formed in a secure manner. ROS 602 provides a wide range of protections against a wide range of "threats" to the secure handling and execution of component assemblies 690.

In the preferred embodiment, ROS 602 assembles component assemblies 690 based on the following types of elements:

Permissions Records ("PERC"s) 808;

5 Method "Cores" 1000;

Load Modules 1100;

Data Elements (e.g., User Data Elements ("UDEs") 1200 and Method Data Elements ("MDEs") 1202); and

Other component assemblies 690.

10

Briefly, a PERC 808 provided by the preferred embodiment is a record corresponding to a VDE object 300 that identifies to ROS 602, among other things, the elements ROS is to assemble together to form a component assembly 690. Thus PERC 808 in effect contains a "list of assembly instructions" or a "plan" specifying what elements ROS 602 is to assemble together into a component assembly and how the elements are to be connected together. PERC 808 may itself contain data or other elements that are to become part of the component assembly 690.

15

20

The PERC 808 may reference one or more method "cores" 1000N. A method core 1000N may define a basic "method" 1000 (e.g., "control," "billing," "metering," etc.)

5 In the preferred embodiment, a "method" 1000 is a collection of basic instructions, and information related to basic instructions, that provides context, data, requirements, and/or relationships for use in performing, and/or preparing to perform, basic instructions in relation to the operation of one or more
10 electronic appliances 600. Basic instructions may be comprised of, for example:

- 15 C machine code of the type commonly used in the programming of computers; pseudo-code for use by an interpreter or other instruction processing program operating on a computer;
- C a sequence of electronically represented logical operations for use with an electronic appliance 600;
- 20 C or other electronic representations of instructions, source code, object code, and/or pseudo code as those terms are commonly understood in the arts.

Information relating to said basic instructions may comprise, for example, data associated intrinsically with basic instructions such as for example, an identifier for the combined basic instructions and intrinsic data, addresses, constants, and/or the like. The information may also, for example, include one or more of the following:

- 10 C information that identifies associated basic instructions and said intrinsic data for access, correlation and/or validation purposes;
- C required and/or optional parameters for use with basic instructions and said intrinsic data;
- C information defining relationships to other methods;
- 15 C data elements that may comprise data values, fields of information, and/or the like;
- C information specifying and/or defining relationships among data elements, basic instructions and/or intrinsic data;
- 20 C information specifying relationships to external data elements;

- C information specifying relationships between and among internal and external data elements, methods, and/or the like, if any exist; and
- 5 C additional information required in the operation of basic instructions and intrinsic data to complete, or attempt to complete, a purpose intended by a user of a method, where required, including additional instructions and/or intrinsic data.

10

Such information associated with a method may be stored, in part or whole, separately from basic instructions and intrinsic data. When these components are stored separately, a method may nevertheless include and encompass the other information and one or more sets of basic instructions and intrinsic data (the latter being included because of said other information's reference to one or more sets of basic instructions and intrinsic data), whether or not said one or more sets of basic instructions and intrinsic data are accessible at any given point in time.

20

Method core 1000' may be parameterized by an "event code" to permit it to respond to different events in different ways.

For example, a METER method may respond to a "use" event by storing usage information in a meter data structure. The same METER method may respond to an "administrative" event by reporting the meter data structure to a VDE clearinghouse or
5 other VDE participant.

In the preferred embodiment, method core 1000' may "contain," either explicitly or by reference, one or more "load modules" 1100 and one or more data elements (UDEs 1200,
10 MDEs 1202). In the preferred embodiment, a "load module" 1100 is a portion of a method that reflects basic instructions and intrinsic data. Load modules 1100 in the preferred embodiment contain executable code, and may also contain data elements ("DTDs" 1108) associated with the executable code. In the
15 preferred embodiment, load modules 1100 supply the program instructions that are actually "executed" by hardware to perform the process defined by the method. Load modules 1100 may contain or reference other load modules.

20 Load modules 1100 in the preferred embodiment are modular and "code pure" so that individual load modules may be reenterable and reusable. In order for components 690 to be

dynamically updatable, they may be individually addressable within a global public name space. In view of these design goals, load modules 1100 are preferably small, code (and code-like) pure modules that are individually named and addressable. A single
5 method may provide different load modules 1100 that perform the same or similar functions on different platforms, thereby making the method scalable and/or portable across a wide range of different electronic appliances.

10 UDEs 1200 and MDEs 1202 may store data for input to or output from executable component assembly 690 (or data describing such inputs and/or outputs). In the preferred embodiment, UDEs 1200 may be user dependent, whereas MDEs 1202 may be user independent.

15 The component assembly example 690(k) shown in Figure 11E comprises a method core 1000', UDEs 1200a & 1200b, an MDE 1202, load modules 1100a-1100d, and a further component assembly 690(k+1). As mentioned above, a PERC 808(k) defines,
20 among other things, the "assembly instructions" for component assembly 690(k), and may contain or reference parts of some or

all of the components that are to be assembled to create a component assembly.

5 One of the load modules 1100b shown in this example is itself comprised of plural load modules 1100c, 1100d. Some of the load modules (e.g., 1100a, 1100d) in this example include one or more "DTD" data elements 1108 (e.g., 1108a, 1108b). "DTD" data elements 1108 may be used, for example, to inform load module 1100a of the data elements included in MDE 1202 and/or UDEs 10 1200a, 1200b. Furthermore, DTDs 1108 may be used as an aspect of forming a portion of an application used to inform a user as to the information required and/or manipulated by one or more load modules 1100, or other component elements. Such an application program may also include functions for creating 15 and/or manipulating UDE(s) 1200, MDE(s) 1202, or other component elements, subassemblies, etc.

Components within component assemblies 690 may be "reused" to form different component assemblies. As mentioned 20 above, figure 11F is an abstract depiction of one example of the same components used for assembling component assembly 690(k) to be reused (e.g., with some additional components

specified by a different set of "assembly instructions" provided in a different PERC 808(l) to form a different component assembly 690(l). Even though component assembly 690(l) is formed from some of the same components used to form component assembly 690(k), these two component assemblies may perform completely different processes in complete different ways.

As mentioned above, ROS 602 provides several layers of security to ensure the security of component assemblies 690. One important security layer involves ensuring that certain component assemblies 690 are formed, loaded and executed only in secure execution space such as provided within an SPU 500. Components 690 and/or elements comprising them may be stored on external media encrypted using local SPU 500 generated and/or distributor provided keys.

ROS 602 also provides a tagging and sequencing scheme that may be used within the loadable component assemblies 690 to detect tampering by substitution. Each element comprising a component assembly 690 may be loaded into an SPU 500, decrypted using encrypt/decrypt engine 522, and then tested/compared to ensure that the proper element has been loaded. Several independent comparisons may be used to ensure

there has been no unauthorized substitution. For example, the public and private copies of the element ID may be compared to ensure that they are the same, thereby preventing gross substitution of elements. In addition, a validation/correlation tag stored under the encrypted layer of the loadable element may be compared to make sure it matches one or more tags provided by a requesting process. This prevents unauthorized use of information. As a third protection, a device assigned tag (e.g., a sequence number) stored under an encryption layer of a loadable element may be checked to make sure it matches a corresponding tag value expected by SPU 500. This prevents substitution of older elements. Validation/correlation tags are typically passed only in secure wrappers to prevent plaintext exposure of this information outside of SPU 500.

The secure component based architecture of ROS 602 has important advantages. For example, it accommodates limited resource execution environments such as provided by a lower cost SPU 500. It also provides an extremely high level of configurability. In fact, ROS 602 will accommodate an almost unlimited diversity of content types, content provider objectives, transaction types and client requirements. In addition, the

ability to dynamically assemble independently deliverable components at execution time based on particular objects and users provides a high degree of flexibility, and facilitates or enables a distributed database, processing, and execution environment.

5

One aspect of an advantage of the component-based architecture provided by ROS 602 relates to the ability to "stage" functionality and capabilities over time. As designed, implementation of ROS 602 is a finite task. Aspects of its wealth of functionality can remain unexploited until market realities dictate the implementation of corresponding VDE application functionality. As a result, initial product implementation investment and complexity may be limited. The process of "surfacing" the full range of capabilities provided by ROS 602 in terms of authoring, administrative, and artificial intelligence applications may take place over time. Moreover, already-designed functionality of ROS 602 may be changed or enhanced at any time to adapt to changing needs or requirements.

10

15

20

More Detailed Discussion of Rights Operating System 602 Architecture

5 Figure 12 shows an example of a detailed architecture of
ROS 602 shown in Figure 10. ROS 602 may include a file system
687 that includes a commercial database manager 730 and
external object repositories 728. Commercial database manager
730 may maintain secure database 610. Object repository 728
may store, provide access to, and/or maintain VDE objects 300.

10

 Figure 12 also shows that ROS 602 may provide one or
more SPEs 503 and/or one or more HPEs 655. As discussed
above, HPE 655 may "emulate" an SPU 500 device, and such
HPEs 655 may be integrated in lieu of (or in addition to) physical
15 SPUs 500 for systems that need higher throughput. Some
security may be lost since HPEs 655 are typically protected by
operating system security and may not provide truly secure
processing. Thus, in the preferred embodiment, for high security
applications at least, all secure processing should take place
20 within an SPE 503 having an execution space within a physical
SPU 500 rather than a HPE 655 using software operating
elsewhere in electronic appliance 600.

As mentioned above, three basic components of ROS 602 are a kernel 680, a Remote Procedure Call (RPC) manager 732 and an object switch 734. These components, and the way they interact with other portions of ROS 602, will be discussed below.

5

Kernel 680

Kernel 680 manages the basic hardware resources of electronic appliance 600, and controls the basic tasking provided by ROS 602. Kernel 680 in the preferred embodiment may include a memory manager 680a, a task manager 680b, and an I/O manager 680c. Task manager 680b may initiate and/or manage initiation of executable tasks and schedule them to be executed by a processor on which ROS 602 runs (e.g., CPU 654 shown in Figure 8). For example, Task manager 680b may include or be associated with a "bootstrap loader" that loads other parts of ROS 602. Task manager 680b may manage all tasking related to ROS 602, including tasks associated with application program(s) 608. Memory manager 680a may manage allocation, deallocation, sharing and/or use of memory (e.g., RAM 656 shown in Figure 8) of electronic appliance 600, and may for example provide virtual memory capabilities as required by an electronic appliance and/or associated application(s). I/O manager 680c

10

15

20

may manage all input to and output from ROS 602, and may interact with drivers and other hardware managers that provide communications and interactivity with physical devices.

5 **RPC Manager 732**

ROS 602 in a preferred embodiment is designed around a "services based" Remote Procedure Call architecture/interface. All functions performed by ROS 602 may use this common interface to request services and share information. For example, 10 SPE(s) 503 provide processing for one or more RPC based services. In addition to supporting SPUs 500, the RPC interface permits the dynamic integration of external services and provides an array of configuration options using existing operating system components. ROS 602 also communicates with external services 15 through the RPC interface to seamlessly provide distributed and/or remote processing. In smaller scale instances of ROS 602, a simpler message passing IPC protocol may be used to conserve resources. This may limit the configurability of ROS 602 services, but this possible limitation may be acceptable in some 20 electronic appliances.

The RPC structure allows services to be called/requested without the calling process having to know or specify where the service is physically provided, what system or device will service the request, or how the service request will be fulfilled. This feature supports families of services that may be scaled and/or customized for specific applications. Service requests can be forwarded and serviced by different processors and/or different sites as easily as they can be forwarded and serviced by a local service system. Since the same RPC interface is used by ROS 602 in the preferred embodiment to request services within and outside of the operating system, a request for distributed and/or remote processing incurs substantially no additional operating system overhead. Remote processing is easily and simply integrated as part of the same service calls used by ROS 602 for requesting local-based services. In addition, the use of a standard RPC interface ("RSI") allows ROS 602 to be modularized, with the different modules presenting a standardized interface to the remainder of the operating system. Such modularization and standardized interfacing permits different vendors/operating system programmers to create different portions of the operating system independently, and also allows the functionality of ROS 602 to be flexibly updated

and/or changed based on different requirements and/or platforms.

5 RPC manager 732 manages the RPC interface. It receives service requests in the form of one or more "Remote Procedure Calls" (RPCs) from a service requestor, and routes the service requests to a service provider(s) that can service the request. For example, when rights operating system 602 receives a request from a user application via user API 682, RPC manager 732 may
10 route the service request to an appropriate service through the "RPC service interface" ("RSI"). The RSI is an interface between RPC manager 732, service requestors, and a resource that will accept and service requests.

15 The RPC interface (RSI) is used for several major ROS 602 subsystems in the preferred embodiment.

 RPC services provided by ROS 602 in the preferred embodiment are divided into subservices, i.e., individual
20 instances of a specific service each of which may be tracked individually by the RPC manager 732. This mechanism permits multiple instances of a specific service on higher throughput

systems while maintaining a common interface across a spectrum of implementations. The subservice concept extends to supporting multiple processors, multiple SPEs 503, multiple HPEs 655, and multiple communications services.

5

The preferred embodiment ROS 602 provides the following RPC based service providers/requestors (each of which have an RPC interface or "RSI" that communicates with RPC manager 732):

10

SPE device driver 736 (this SPE device driver is connected to an SPE 503 in the preferred embodiment);

HPE Device Driver 738 (this HPE device driver is connected to an HPE 738 in the preferred embodiment);

15

Notification Service 740 (this notification service is connected to user notification interface 686 in the preferred embodiment);

API Service 742 (this API service is connected to user API 682 in the preferred embodiment);

20

Redirector 684;

Secure Database (File) Manager 744 (this secure database or file manager 744 may connect to and interact with

commercial database manager 730 and secure files
610 through a cache manager 746, a database
interface 748, and a database driver 750);
Name Services Manager 752;
5 Outgoing Administrative Objects Manager 754;
Incoming Administrative Objects Manager 756;
a Gateway 734 to object switch 734 (this is a path used to
allow direct communication between RPC manager
732 and Object Switch 734); and
10 Communications Manager 776.

The types of services provided by HPE 655, SPE 503, User
Notification 686, API 742 and Redirector 684 have already been
described above. Here is a brief description of the type(s) of
15 services provided by OS resources 744, 752, 754, 756 and 776:

Secure Database Manager 744 services requests for access
to secure database 610;
Name Services Manager 752 services requests relating to
user, host, or service identification;
20 Outgoing Administrative Objects Manager 754 services
requests relating to outgoing administrative objects;

Incoming Administrative Objects Manager 756 services

requests relating to incoming administrative objects;

and

Communications Manager 776 services requests relating to

5 communications between electronic appliance 600

and the outside world.

Object Switch 734

Object switch 734 handles, controls and communicates
10 (both locally and remotely) VDE objects 300. In the preferred
embodiment, the object switch may include the following
elements:

a stream router 758;

a real time stream interface(s) 760 (which may be

15 connected to real time data feed(s) 694);

a time dependent stream interface(s) 762;

a intercept 692;

a container manager 764;

one or more routing tables 766; and

20 buffering/storage 768.

Stream router 758 routes to/from "real time" and "time
independent" data streams handled respectively by real time

stream interface(s) 760 and time dependent stream interface(s)
762. Intercept 692 intercepts I/O requests that involve real-time
information streams such as, for example, real time feed 694.
The routing performed by stream router 758 may be determined
5 by routing tables 766. Buffering/storage 768 provides temporary
store-and-forward, buffering and related services. Container
manager 764 may (typically in conjunction with SPE 503)
perform processes on VDE objects 300 such as constructing,
deconstructing, and locating portions of objects.

10

Object switch 734 communicates through an Object Switch
Interface ("OSI") with other parts of ROS 602. The Object Switch
Interface may resemble, for example, the interface for a Unix
socket in the preferred embodiment. Each of the "OSI" interfaces
15 shown in Figure 12 have the ability to communicate with object
switch 734.

ROS 602 includes the following object switch service
providers/resources (each of which can communicate with the
20 object switch 734 through an "OSI"):

Outgoing Administrative Objects Manager 754;

Incoming Administrative Objects Manager 756;

Gateway 734 (which may translate RPC calls into object switch calls and vice versa so RPC manager 732 may communicate with object switch 734 or any other element having an OSI to, for example, provide
5 and/or request services);

External Services Manager 772;
Object Submittal Manager 774; and
Communications Manager 776.

10

Briefly,

Object Repository Manager 770 provides services relating to access to object repository 728;

15

External Services Manager 772 provides services relating to requesting and receiving services externally, such as from a network resource or another site;

20

Object Submittal Manager 774 provides services relating to how a user application may interact with object switch 734 (since the object submittal manager provides an interface to an application program 608, it could be considered part of user API 682); and
Communications Manager 776 provides services relating to communicating with the outside world.

In the preferred embodiment, communications manager 776 may include a network manager 780 and a mail gateway (manager) 782. Mail gateway 782 may include one or more mail filters 784 to, for example, automatically route VDE related electronic mail between object switch 734 and the outside world electronic mail services. External Services Manager 772 may interface to communications manager 776 through a Service Transport Layer 786. Service Transport Layer 786a may enable External Services Manager 772 to communicate with external computers and systems using various protocols managed using the service transport layer 786.

The characteristics of and interfaces to the various subsystems of ROS 680 shown in Figure 12 are described in more detail below.

RPC Manager 732 and Its RPC Services Interface

As discussed above, the basic system services provided by ROS 602 are invoked by using an RPC service interface (RSI). This RPC service interface provides a generic, standardized interface for different services systems and subsystems provided by ROS 602.

RPC Manager 732 routes RPCs requesting services to an appropriate RPC service interface. In the preferred embodiment, upon receiving an RPC call, RPC manager 732 determines one or more service managers that are to service the request. RPC
5 manager 732 then routes a service request to the appropriate service(s) (via a RSI associated with a service) for action by the appropriate service manager(s).

For example, if a SPE 503 is to service a request, the RPC
10 Manager 732 routes the request to RSI 736a, which passes the request on to SPE device driver 736 for forwarding to the SPE. Similarly, if HPE 655 is to service the request, RPC Manager 732 routes the request to RSI 738a for forwarding to a HPE. In one preferred embodiment, SPE 503 and HPE 655 may perform
15 essentially the same services so that RSIs 736a, 738a are different instances of the same RSI. Once a service request has been received by SPE 503 (or HPE 655), the SPE (or HPE) typically dispatches the request internally using its own internal RPC manager (as will be discussed shortly). Processes within
20 SPEs 503 and HPEs 655 can also generate RPC requests. These requests may be processed internally by a SPE/HPE, or if not

internally serviceable, passed out of the SPE/HPE for dispatch by
RPC Manager 732.

Remote (and local) procedure calls may be dispatched by a
5 RPC Manager 732 using an "RPC Services Table." An RPC
Services Table describes where requests for specific services are
to be routed for processing. Each row of an RPC Services Table
in the preferred embodiment contains a services ID, the location
of the service, and an address to which control will be passed to
10 service a request. An RPC Services Table may also include
control information that indicates which instance of the RPC
dispatcher controls the service. Both RPC Manager 732 and any
attached SPEs 503 and HPEs 655 may have symmetric copies of
the RPC Services Table. If an RPC service is not found in the
15 RPC services tables, it is either rejected or passed to external
services manager 772 for remote servicing.

Assuming RPC manager 732 finds a row corresponding to
the request in an RPC Services Table, it may dispatch the
20 request to an appropriate RSI. The receiving RSI accepts a
request from the RPC manager 732 (which may have looked up
the request in an RPC service table), and processes that request

in accordance with internal priorities associated with the specific service.

In the preferred embodiment, RPC Service Interface(s) supported by RPC Manager 732 may be standardized and published to support add-on service modules developed by third party vendors, and to facilitate scalability by making it easier to program ROS 602. The preferred embodiment RSI closely follows the DOS and Unix device driver models for block devices so that common code may be developed for many platforms with minimum effort. An example of one possible set of common entry points are listed below in the table.

Interface call	Description
SVC_LOAD	Load a service manager and return its status.
SVC_UNLOAD	Unload a service manager.
SVC_MOUNT	Mount (load) a dynamically loaded subservice and return its status.
SVC_UNMOUNT	Unmount (unload) a dynamically loaded subservice.
SVC_OPEN	Open a mounted subservice.
SVC_CLOSE	Close a mounted subservice.
SVC_READ	Read a block from an opened subservice.

SVC_WRITE	Write a block to an opened subservice.
SVC_IOCTL	Control a subservice or a service manager.

Load

5 In the preferred embodiment, services (and the associated
RSIs they present to RPC manager 732) may be activated during
boot by an installation boot process that issues an RPC LOAD.
This process reads an RPC Services Table from a configuration
file, loads the service module if it is run time loadable (as opposed
10 to being a kernel linked device driver), and then calls the LOAD
entry point for the service. A successful return from the LOAD
entry point will indicate that the service has properly loaded and
is ready to accept requests.

15 RPC LOAD Call Example: SVC_LOAD (long service_id)

 This LOAD interface call is called by the RPC manager 732
during rights operating system 602 initialization. It permits a
service manager to load any dynamically loadable components
and to initialize any device and memory required by the service.
20 The service number that the service is loaded as is passed in as
service_id parameter. In the preferred embodiment, the service

returns 0 if the initialization process was completed successfully or an error number if some error occurred.

Mount

5 Once a service has been loaded, it may not be fully functional for all subservices. Some subservices (e.g., communications based services) may require the establishment of additional connections, or they may require additional modules to be loaded. If the service is defined as "mountable," a RPC
10 manager 732 will call the MOUNT subservice entry point with the requested subservice ID prior to opening an instance of a subservice.

RPC MOUNT Call Example:

15 SVC_MOUNT (long service_id, long subservice_id, BYTE *buffer)

 This MOUNT interface call instructs a service to make a specific subservice ready. This may include services related to networking, communications, other system services, or external
20 resources. The service_id and subservice_id parameters may be specific to the specific service being requested. The buffer

parameter is a memory address that references a control structure appropriate to a specific service.

Open

5 Once a service is loaded and "mounted," specific instances of a service may be "opened" for use. "Opening" an instance of a service may allocate memory to store control and status information. For example, in a BSD socket based network connection, a LOAD call will initialize the software and protocol control tables, a MOUNT call will specify networks and hardware
10 resources, and an OPEN will actually open a socket to a remote installation.

 Some services, such as commercial database manager 730
15 that underlies the secure database service, may not be "mountable." In this case, a LOAD call will make a connection to a database manager 730 and ensure that records are readable. An OPEN call may create instances of internal cache manager 746 for various classes of records.

20

RPC OPEN Call Example:

SVC_OPEN (long service_id, long subservice_id, BYTE

***buffer, int (*receive) (long request_id))**

This OPEN interface call instructs a service to open a specific subservice. The `service_id` and `subservice_id` parameters are specific to the specific service being requested, and the `buffer` parameter is a memory address that references a control structure appropriate to a specific service.

The optional `receive` parameter is the address of a notification callback function that is called by a service whenever a message is ready for the service to retrieve it. One call to this address is made for each incoming message received. If the caller passes a NULL to the interface, the software will not generate a callback for each message.

15 **Close, Unmount and Unload**

The converse of the OPEN, MOUNT, and LOAD calls are CLOSE, UNMOUNT, and UNLOAD. These interface calls release any allocated resources back to ROS 602 (e.g., memory manager 680a).

20

RPC CLOSE Call Example: SVC_CLOSE (long svc_handle)

This LOAD interface call closes an open service "handle."
A service "handle" describes a service and subservice that a user wants to close. The call returns 0 if the CLOSE request succeeds
5 (and the handle is no longer valid) or an error number.

RPC UNLOAD Call Example: SVC_UNLOAD (void)

This UNLOAD interface call is called by a RPC manager
732 during shutdown or resource reallocation of rights operating
10 system 602. It permits a service to close any open connections,
flush buffers, and to release any operating system resources that
it may have allocated. The service returns 0.

**RPC UNMOUNT Call Example: SVC_UNMOUNT (long
15 service_id, long subservice_id)**

This UNMOUNT interface call instructs a service to
deactivate a specific subservice. The service_id and
subservice_id parameters are specific to the specific service
being requested, and must have been previously mounted using
20 the SVC_MOUNT() request. The call releases all system
resources associated with the subservice before it returns.

Read and Write

The READ and WRITE calls provide a basic mechanism for sending information to and receiving responses from a mounted and opened service. For example, a service has requests written to it in the form of an RPC request, and makes its response available to be read by RPC Manager 732 as they become available.

RPC READ Call Example:

10 SVC_READ (long svc_handle, long request_id, BYTE
 *buffer, long size)

 This READ call reads a message response from a service. The svc_handle and request_id parameters uniquely identify a request. The results of a request will be stored in the user specified buffer up to size bytes. If the buffer is too small, the first size bytes of the message will be stored in the buffer and an error will be returned.

 If a message response was returned to the caller's buffer correctly, the function will return 0. Otherwise, an error message will be returned.

RPC WRITE Call Example:

SVC_write (long service_id, long subservice_id, BYTE
*buffer, long size, int (*receive) (long request_id)

5 This WRITE call writes a message to a service and
subservice specified by the service_id/subservice_id parameter
pair. The message is stored in buffer (and usually conforms to
the VDE RPC message format) and is size bytes long. The
function returns the request id for the message (if it was
accepted for sending) or an error number. If a user specifies the
10 receive callback functions, all messages regarding a request will
be sent to the request specific callback routine instead of the
generalized message callback.

Input/Output Control

15 The IOCTL ("Input/Output ConTroL") call provides a
mechanism for querying the status of and controlling a loaded
service. Each service type will respond to specific general IOCTL
requests, all required class IOCTL requests, and service specific
IOCTL requests.

20

RPC IOCTL Call Example: ROI_SVC_IOCTL (long service_id,
 long subservice_id,
 int command, BYTE *buffer)

5 This IOCTL function provides a generalized control
 interface for a RSI. A user specifies the service_id parameter
 and an optional subservice_id parameter that they wish to
 control. They specify the control command parameter(s), and a
 buffer into/from which the command parameters may be
 10 written/read. An example of a list of commands and the
 appropriate buffer structures are given below.

Command	Structure	Description
GET_INFO	SVC_INFO	Returns information about a service/subservice.
15 GET_STATS	SVC_STATS	Returns current statistics about a service/subservice.
CLR_STATS	None	Clears the statistics about a service/subservice.

20 * * * * *

Now that a generic RPC Service Interface provided by the preferred embodiment has been described, the following

description relates to particular examples of services provided by ROS 602.

SPE Device Driver 736

5 SPE device driver 736 provides an interface between ROS 602 and SPE 503. Since SPE 503 in the preferred embodiment runs within the confines of an SPU 500, one aspect of this device driver 736 is to provide low level communications services with the SPU 500 hardware. Another aspect of SPE device driver 736
10 is to provide an RPC service interface (RSI) 736a particular to SPE 503 (this same RSI may be used to communicate with HPE 655 through HPE device driver 738).

 SPE RSI 736a and driver 736 isolates calling processes
15 within ROS 602 (or external to the ROS) from the detailed service provided by the SPE 503 by providing a set of basic interface points providing a concise function set. This has several advantages. For example, it permits a full line of scaled SPUs 500 that all provide common functionality to the outside world
20 but which may differ in detailed internal structure and architecture. SPU 500 characteristics such as the amount of memory resident in the device, processor speed, and the number

of services supported within SPU 500 may be the decision of the specific SPU manufacturer, and in any event may differ from one SPU configuration to another. To maintain compatibility, SPE device driver 736 and the RSI 736a it provides conform to a basic common RPC interface standard that "hides" differences between detailed configurations of SPUs 500 and/or the SPEs 503 they may support.

To provide for such compatibility, SPE RSI 736a in the preferred embodiment follows a simple block based standard. In the preferred embodiment, an SPE RSI 736a may be modeled after the packet interfaces for network Ethernet cards. This standard closely models the block mode interface characteristics of SPUs 500 in the preferred embodiment.

An SPE RSI 736a allows RPC calls from RPC manager 732 to access specific services provided by an SPE 736. To do this, SPE RSI 736a provides a set of "service notification address interfaces." These provide interfaces to individual services provided by SPE 503 to the outside world. Any calling process within ROS 602 may access these SPE-provided services by directing an RPC call to SPE RSI 736a and specifying a

corresponding "service notification address" in an RPC call. The specified "service notification address" causes SPE 503 to internally route an RPC call to a particular service within an SPE. The following is a listing of one example of a SPE service breakdown for which individual service notification addresses may be provided:

Channel Services Manager

Authentication Manager/Secure Communications Manager

Secure Database Manager

10

The Channel Services Manager is the principal service provider and access point to SPE 503 for the rest of ROS 602. Event processing, as will be discussed later, is primarily managed (from the point of view of processes outside SPE 503) by this service. The Authentication Manager/Secure Communications Manager may provide login/logout services for users of ROS 602, and provide a direct service for managing communications (typically encrypted or otherwise protected) related to component assemblies 690, VDE objects 300, etc. Requests for display of information (e.g., value remaining in a financial budget) may be provided by a direct service request to a Secure Database Manager inside SPE 503. The instances of

15

20

Authentication Manager/Secure Communications Manager and Secure Database Manager, if available at all, may provide only a subset of the information and/or capabilities available to processes operating inside SPE 503. As stated above, most (potentially all) service requests entering SPE are routed to a Channel Services Manager for processing. As will be discussed in more detail later on, most control structures and event processing logic is associated with component assemblies 690 under the management of a Channel Services Manager.

10

The SPE 503 must be accessed through its associated SPE driver 736 in this example. Generally, calls to SPE driver 736 are made in response to RPC calls. In this example, SPE driver RSI 736a may translate RPC calls directed to control or ascertain information about SPE driver 736 into driver calls. SPE driver RSI 736a in conjunction with driver 736 may pass RPC calls directed to SPE 503 through to the SPE.

15

The following table shows one example of SPE device driver 736 calls:

20

Entry Point	Description
SPE_info()	Returns summary information about the SPE driver 736 (and SPE 503)
SPE_initialize_interface()	Initializes SPE driver 736, and sets the default notification address for received packets.
5 SPE_terminate_interface()	Terminates SPE driver 736 and resets SPU 500 and the driver 736.
SPE_reset_interface()	Resets driver 736 without resetting SPU 500.
SPE_get_stats()	Return statistics for notification addresses and/or an entire driver 736.
SPE_clear_stats()	Clears statistics for a specific notification address and/or an entire driver 736.
SPE_set_notify()	Sets a notification address for a specific service ID.
10 SPE_get_notify()	Returns a notification address for a specific service ID.
SPE_tx_pkt()	Sends a packet (e.g., containing an RPC call) to SPE 503 for processing.

15 The following are more detailed examples of each of the SPE driver calls set forth in the table above.

Example of an "SPE Information" Driver Call: SPE_info (void)

This function returns a pointer to an SPE_INFO data structure that defines the SPE device driver 736a. This data structure may provide certain information about SPE device driver 736, RSI 736a and/or SPU 500. An example of a SPE_INFO structure is described below:

5

10

15

Version Number/ID for SPE Device Driver 736
Version Number/ID for SPE Device Driver RSI 736
Pointer to name of SPE Device Driver 736
Pointer to ID name of SPU 500
Functionality Code Describing SPE Capabilities/functionality

20

Example of an SPE "Initialize Interface" Driver Call:

SPE_initialize_interface (int (fcn *receiver)(void))

A receiver function passed in by way of a parameter will be called for all packets received from SPE 503 unless their

destination service is over-ridden using the set_notify() call. A receiver function allows ROS 602 to specify a format for packet communication between RPC manager 732 and SPE 503.

5 This function returns "0" in the preferred embodiment if the initialization of the interface succeeds and non-zero if it fails. If the function fails, it will return a code that describes the reason for the failure as the value of the function.

10 **Example of an SPE "Terminate Interface" Driver Call:**

SPE_terminate_interface (void)

 In the preferred embodiment, this function shuts down SPE Driver 736, clears all notification addresses, and terminates all outstanding requests between an SPE and an ROS RPC manager 732. It also resets an SPE 503 (e.g., by a warm reboot of SPU 500) after all requests are resolved.

15

 Termination of driver 736 should be performed by ROS 602 when the operating system is starting to shut down. It may also be necessary to issue this call if an SPE 503 and ROS 602 get so far out of synchronization that all processing in an SPE must be reset to a known state.

20

Example of an SPE "Reset Interface" Driver Call:

SPE_reset_interface (void)

5 This function resets driver 736, terminates all outstanding requests between SPE 503 and an ROS RPC manager 732, and clears all statistics counts. It does not reset the SPU 500, but simply restores driver 736 to a known stable state.

Example of an SPE "Get Statistics" Driver Call: SPE_get_stats

10 (long service_id)

This function returns statistics for a specific service notification interface or for the SPE driver 736 in general. It returns a pointer to a static buffer that contains these statistics or NULL if statistics are unavailable (either because an interface is not initialized or because a receiver address was not specified).
 15 An example of the SPE_STATS structure may have the following definition:

20

Service id
packets rx
packets tx
bytes rx
bytes tx

5

errors rx
errors tx
requests tx
req tx completed
req tx cancelled
req rx
req rx completed
req rx cancelled

10

If a user specifies a service ID, statistics associated with packets sent by that service are returned. If a user specified 0 as the parameter, the total packet statistics for the interface are returned.

15

Example of an SPE "Clear Statistics" Driver Call:

SPE_clear_stats (long service_id)

20

This function clears statistics associated with the SPE service_id specified. If no service_id is specified (i.e., the caller passes in 0), global statistics will be cleared. The function returns 0 if statistics are successfully cleared or an error number if an error occurs.

Example of an SPE "Set Notification Address" Driver Call:

SPE_set_notify (long service_id, int (fcn*receiver) (void))

5 This function sets a notification address (receiver) for a specified service. If the notification address is set to NULL, SPE device driver 736 will send notifications for packets to the specified service to the default notification address.

Example of a SPE "Get Notification Address" Driver Call:

SPE_get_notify (long service_id)

10 This function returns a notification address associated with the named service or NULL if no specific notification address has been specified.

Example of an SPE "Send Packet" Driver Call:

15 send_pkt (BYTE *buffer, long size, int (far *receive) (void))

This function sends a packet stored in buffer of "length" size. It returns 0 if the packet is sent successfully, or returns an error code associated with the failure.

20 Redirector Service Manager 684

The redirector 684 is a piece of systems integration software used principally when ROS 602 is provided by "adding

on“ to a pre-existing operating system or when ”transparent“
operation is desired for some VDE functions, as described earlier.
In one embodiment the kernel 680, part of communications
manager 776, file system 687, and part of API service 742 may be
5 part of a pre-existing operating system such as DOS, Windows,
UNIX, Macintosh System, OS9, PSOS, OS/2, or other operating
system platform. The remainder of ROS 602 subsystems shown
in Figure 12 may be provided as an ”add on“ to a preexisting
operating system. Once these ROS subsystems have been
10 supplied and ”added on,“ the integrated whole comprises the ROS
602 shown in Figure 12.

In a scenario of this type of integration, ROS 602 will
continue to be supported by a preexisting OS kernel 680, but may
15 supplement (or even substitute) many of its functions by
providing additional add-on pieces such as, for example, a virtual
memory manager.

Also in this integration scenario, an add-on portion of API
20 service 742 that integrates readily with a preexisting API service
is provided to support VDE function calls. A pre-existing API
service integrated with an add-on portion supports an enhanced

set of operating system calls including both calls to VDE
functions 604 and calls to functions 606 other than VDE
functions (see Figure 11A). The add-on portion of API service
742 may translate VDE function calls into RPC calls for routing
5 by RPC manager 732.

ROS 602 may use a standard communications manager
776 provided by the preexisting operating system, or it may
provide "add ons" and/or substitutions to it that may be readily
10 integrated into it. Redirector 684 may provide this integration
function.

This leaves a requirement for ROS 602 to integrate with a
preexisting file system 687. Redirector 684 provides this
15 integration function.

In this integration scenario, file system 687 of the
preexisting operating system is used for all accesses to secondary
storage. However, VDE objects 300 may be stored on secondary
20 storage in the form of external object repository 728, file system
687, or remotely accessible through communications manager
776. When object switch 734 wants to access external object

repository 728, it makes a request to the object repository manager 770 that then routes the request to object repository 728 or to redirector 692 (which in turn accesses the object in file system 687).

5

Generally, redirector 684 maps VDE object repository 728 content into preexisting calls to file system 687. The redirector 684 provides preexisting OS level information about a VDE object 300, including mapping the object into a preexisting OS's name space. This permits seamless access to VDE protected content using "normal" file system 687 access techniques provided by a preexisting operating system.

10

In the integration scenarios discussed above, each preexisting target OS file system 687 has different interface requirements by which the redirector mechanism 684 may be "hooked." In general, since all commercially viable operating systems today provide support for network based volumes, file systems, and other devices (e.g., printers, modems, etc.), the redirector 684 may use low level network and file access "hooks" to integrate with a preexisting operating system. "Add-ons" for

20

supporting VDE functions 602 may use these existing hooks to integrate with a preexisting operating system.

User Notification Service Manager 740

5 User Notification Service Manager 740 and associated user notification exception interface ("pop up") 686 provides ROS 602 with an enhanced ability to communicate with a user of electronic appliance 600. Not all applications 608 may be designed to respond to messaging from ROS 602 passed through API 682,
10 and it may in any event be important or desirable to give ROS 602 the ability to communicate with a user no matter what state an application is in. User notification services manager 740 and interface 686 provides ROS 602 with a mechanism to communicate directly with a user, instead of or in addition to
15 passing a return call through API 682 and an application 608. This is similar, for example, to the ability of the Windows operating system to display a user message in a "dialog box" that displays "on top of" a running application irrespective of the state of the application.

20

 The User Notification 686 block in the preferred embodiment may be implemented as application code. The

implementation of interface 740a is preferably built over notification service manager 740, which may be implemented as part of API service manager 742. Notification services manager 740 in the preferred embodiment provides notification support to dispatch specific notifications to an appropriate user process via the appropriate API return, or by another path. This mechanism permits notifications to be routed to any authorized process—not just back to a process that specified a notification mechanism.

10 **API Service Manager 742**

The preferred embodiment API Service Manager 742 is implemented as a service interface to the RPC service manager 732. All user API requests are built on top of this basic interface. The API Service Manager 742 preferably provides a service instance for each running user application 608.

Most RPC calls to ROS functions supported by API Service Manager 742 in the preferred embodiment may map directly to service calls with some additional parameter checking. This mechanism permits developers to create their own extended API libraries with additional or changed functionality.

In the scenario discussed above in which ROS 602 is formed by integrating "add ons" with a preexisting operating system, the API service 742 code may be shared (e.g., resident in a host environment like a Windows DLL), or it may be directly
5 linked with an applications's code— depending on an application programmer's implementation decision, and/or the type of electronic appliance 600. The Notification Service Manager 740 may be implemented within API 682. These components interface with Notification Service component 686 to provide a
10 transition between system and user space.

Secure Database Service Manager ("SDSM") 744

There are at least two ways that may be used for managing secure database 600:

- 15 C a commercial database approach, and
- C a site record number approach.

Which way is chosen may be based on the number of records that a VDE site stores in the secure database 610.

20 The commercial database approach uses a commercial database to store securely wrapped records in a commercial database. This way may be preferred when there are a large

number of records that are stored in the secure database 610. This way provides high speed access, efficient updates, and easy integration to host systems at the cost of resource usage (most commercial database managers use many system resources).

5

The site record number approach uses a "site record number" ("SRN") to locate records in the system. This scheme is preferred when the number of records stored in the secure database 610 is small and is not expected to change extensively over time. This way provides efficient resources use with limited update capabilities. SRNs permit further grouping of similar data records to speed access and increase performance.

10

Since VDE 100 is highly scalable, different electronic appliances 600 may suggest one way more than the other. For example, in limited environments like a set top, PDA, or other low end electronic appliance, the SRN scheme may be preferred because it limits the amount of resources (memory and processor) required. When VDE is deployed on more capable electronic appliances 600 such as desktop computers, servers and at clearinghouses, the commercial database scheme may be more

15

20

desirable because it provides high performance in environments where resources are not limited.

5 One difference between the database records in the two approaches is whether the records are specified using a full VDE ID or SRN. To translate between the two schemes, a SRN reference may be replaced with a VDE ID database reference wherever it occurs. Similarly, VDE IDs that are used as indices or references to other items may be replaced by the appropriate
10 SRN value.

In the preferred embodiment, a commercially available database manager 730 is used to maintain secure database 610. ROS 602 interacts with commercial database manager 730
15 through a database driver 750 and a database interface 748. The database interface 748 between ROS 602 and external, third party database vendors' commercial database manager 730 may be an open standard to permit any database vendor to implement a VDE compliant database driver 750 for their products.

20 ROS 602 may encrypt each secure database 610 record so that a VDE-provided security layer is "on top of" the commercial

database structure. In other words, SPE 736 may write secure records in sizes and formats that may be stored within a database record structure supported by commercial database manager 730. Commercial database manager 730 may then be used to organize, store, and retrieve the records. In some embodiments, it may be desirable to use a proprietary and/or newly created database manager in place of commercial database manager 730. However, the use of commercial database manager 730 may provide certain advantages such as, for example, an ability to use already existing database management product(s).

The Secure Database Services Manager ("SDSM") 744 makes calls to an underlying commercial database manager 730 to obtain, modify, and store records in secure database 610. In the preferred embodiment, "SDSM" 744 provides a layer "on top of" the structure of commercial database manager 730. For example, all VDE-secure information is sent to commercial database manager 730 in encrypted form. SDSM 744 in conjunction with cache manager 746 and database interface 748 may provide record management, caching (using cache manager 746), and related services (on top of) commercial database systems 730 and/or record managers. Database Interface 748

and cache manager 746 in the preferred embodiment do not present their own RSI, but rather the RPC Manager 732 communicates to them through the Secure Database Manager RSI 744a.

5

Name Services Manager 752

The Name Services Manager 752 supports three subservices: user name services, host name services, and services name services. User name services provides mapping and lookup between user name and user ID numbers, and may also support other aspects of user-based resource and information security. Host name services provides mapping and lookup between the names (and other information, such as for example address, communications connection/routing information, etc.) of other processing resources (e.g., other host electronic appliances) and VDE node IDs. Services name service provides a mapping and lookup between services names and other pertinent information such as connection information (e.g., remotely available service routing and contact information) and service IDs.

20

Name Services Manager 752 in the preferred embodiment is connected to External Services Manager 772 so that it may provide external service routing information directly to the external services manager. Name services manager 752 is also
5 connected to secure database manager 744 to permit the name services manager 752 to access name services records stored within secure database 610.

External Services Manager 772 & Services Transport Layer 786

10 The External Services Manager 772 provides protocol support capabilities to interface to external service providers. External services manager 772 may, for example, obtain external service routing information from name services manager 752, and then initiate contact to a particular external service (e.g.,
15 another VDE electronic appliance 600, a financial clearinghouse, etc.) through communications manager 776. External services manager 772 uses a service transport layer 786 to supply communications protocols and other information necessary to provide communications.

20

There are several important examples of the use of External Services Manager 772. Some VDE objects may have

some or all of their content stored at an Object Repository 728 on an electronic appliance 600 other than the one operated by a user who has, or wishes to obtain, some usage rights to such VDE objects. In this case, External Services Manager 772 may
5 manage a connection to the electronic appliance 600 where the VDE objects desired (or their content) is stored. In addition, file system 687 may be a network file system (e.g., Netware, LANtastic, NFS, etc.) that allows access to VDE objects using
10 redirecter 684. Object switch 734 also supports this capability.

If External Services Manager 772 is used to access VDE objects, many different techniques are possible. For example, the VDE objects may be formatted for use with the World Wide Web protocols (HTML, HTTP, and URL) by including relevant
15 headers, content tags, host ID to URL conversion (e.g., using Name Services Manager 752) and an HTTP-aware instance of Services Transport Layer 786.

In other examples, External Services Manager 772 may be
20 used to locate, connect to, and utilize remote event processing services; smart agent execution services (both to provide these services and locate them); certification services for Public Keys;

remote Name Services; and other remote functions either supported by ROS 602 RPCs (e.g., have RSIs), or using protocols supported by Services Transport Layer 786.

5 **Outgoing Administrative Object Manager 754**

 Outgoing administrative object manager 754 receives administrative objects from object switch 734, object repository manager 770 or other source for transmission to another VDE electronic appliance. Outgoing administrative object manager
10 754 takes care of sending the outgoing object to its proper destination. Outgoing administrative object manager 754 may obtain routing information from name services manager 752, and may use communications service 776 to send the object.
 Outgoing administrative object manager 754 typically maintains
15 records (in concert with SPE 503) in secure database 610 (e.g., shipping table 444) that reflect when objects have been successfully transmitted, when an object should be transmitted, and other information related to transmission of objects.

20 **Incoming Administrative Object Manager 756**

 Incoming administrative object manager 756 receives administrative objects from other VDE electronic appliances 600

via communications manager 776. It may route the object to object repository manager 770, object switch 734 or other destination. Incoming administrative object manager 756 typically maintains records (in concert with SPE 503) in secure database 610 (e.g., receiving table 446) that record which objects have been received, objects expected for receipt, and other information related to received and/or expected objects.

Object Repository Manager 770

Object repository manager 770 is a form of database or file manager. It manages the storage of VDE objects 300 in object repository 728, in a database, or in the file system 687. Object repository manager 770 may also provide the ability to browse and/or search information related to objects (such as summaries of content, abstracts, reviewers' commentary, schedules, promotional materials, etc.), for example, by using INFORMATION methods associated with VDE objects 300.

Object Submittal Manager 774

Object submittal manager 774 in the preferred embodiment provides an interface between an application 608 and object switch 734, and thus may be considered in some

respects part of API 682. For example, it may allow a user application to create new VDE objects 300. It may also allow incoming/outgoing administrative object managers 756, 754 to create VDE objects 300 (administrative objects).

5

Figure 12A shows how object submittal manager 774 may be used to communicate with a user of electronic appliance 600 to help to create a new VDE object 300. Figure 12A shows that object creation may occur in two stages in the preferred embodiment: an object definition stage 1220, and an object creation stage 1230. The role of object submittal manager 774 is indicated by the two different "user input" depictions (774(1), 774(2)) shown in Figure 12A.

10

15

In one of its roles or instances, object submittal manager 774 provides a user interface 774a that allows the user to create an object configuration file 1240 specifying certain characteristics of a VDE object 300 to be created. This user interface 774a may, for example, allow the user to specify that she wants to create an object, allow the user to designate the content the object will contain, and allow the user to specify certain other aspects of the

20

information to be contained within the object (e.g., rules and control information, identifying information, etc.).

5 Part of the object definition task 1220 in the preferred embodiment may be to analyze the content or other information to be placed within an object. Object definition user interface 774a may issue calls to object switch 734 to analyze "content" or other information that is to be included within the object to be created in order to define or organize the content into "atomic
10 elements" specified by the user. As explained elsewhere herein, such "atomic element" organizations might, for example, break up the content into paragraphs, pages or other subdivisions specified by the user, and might be explicit (e.g., inserting a control character between each "atomic element") or implicit.
15 Object switch 734 may receive static and dynamic content (e.g., by way of time independent stream interface 762 and real time stream interface 760), and is capable of accessing and retrieving stored content or other information stored within file system 687.

20 The result of object definition 1240 may be an object configuration file 1240 specifying certain parameters relating to the object to be created. Such parameters may include, for

example, map tables, key management specifications, and event
method parameters. The object construction stage 1230 may
take the object configuration file 1240 and the information or
content to be included within the new object as input, construct
5 an object based on these inputs, and store the object within object
repository 728.

Object construction stage 1230 may use information in
object configuration file 1240 to assemble or modify a container.
10 This process typically involves communicating a series of events
to SPE 503 to create one or more PERCs 808, public headers,
private headers, and to encrypt content, all for storage in the new
object 300 (or within secure database 610 within records
associated with the new object).

15
The object configuration file 1240 may be passed to
container manager 764 within object switch 734. Container
manager 734 is responsible for constructing an object 300 based
on the object configuration file 1240 and further user input. The
20 user may interact with the object construction 1230 through
another instance 774(2) of object submittal manager 774. In this
further user interaction provided by object submittal manager

774, the user may specify permissions, rules and/or control information to be applied to or associated with the new object 300. To specify permissions, rules and control information, object submittal manager 774 and/or container manager 764 within
5 object switch 734 generally will, as mentioned above, need to issue calls to SPE 503 (e.g., through gateway 734) to cause the SPE to obtain appropriate information from secure database 610, generate appropriate database items, and store the database items into the secure database 610 and/or provide them in
10 encrypted, protected form to the object switch for incorporation into the object. Such information provided by SPE 503 may include, in addition to encrypted content or other information, one or more PERCs 808, one or more method cores 1000', one or more load modules 1100, one or more data structures such as
15 UDEs 1200 and/or MDEs 1202, along with various key blocks, tags, public and private headers, and error correction information.

The container manager 764 may, in cooperation with SPE
20 503, construct an object container 302 based at least in part on parameters about new object content or other information as specified by object configuration file 1240. Container manager

764 may then insert into the container 302 the content or other information (as encrypted by SPE 503) to be included in the new object. Container manager 764 may also insert appropriate permissions, rules and/or control information into the container 5 302 (this permissions, rules and/or control information may be defined at least in part by user interaction through object submittal manager 774, and may be processed at least in part by SPE 503 to create secure data control structures). Container manager 764 may then write the new object to object repository 10 687, and the user or the electronic appliance may "register" the new object by including appropriate information within secure database 610.

Communications Subsystem 776

15 Communications subsystem 776, as discussed above, may be a conventional communications service that provides a network manager 780 and a mail gateway manager 782. Mail filters 784 may be provided to automatically route objects 300 and other VDE information to/from the outside world.

20 Communications subsystem 776 may support a real time content feed 684 from a cable, satellite or other telecommunications link.

Secure Processing Environment 503

As discussed above in connection with Figure 12, each electronic appliance 600 in the preferred embodiment includes one or more SPEs 503 and/or one or more HPEs 655. These
5 secure processing environments each provide a protected execution space for performing tasks in a secure manner. They may fulfill service requests passed to them by ROS 602, and they may themselves generate service requests to be satisfied by other services within ROS 602 or by services provided by another VDE
10 electronic appliance 600 or computer.

In the preferred embodiment, an SPE 503 is supported by the hardware resources of an SPU 500. An HPE 655 may be supported by general purpose processor resources and rely on
15 software techniques for security/protection. HPE 655 thus gives ROS 602 the capability of assembling and executing certain component assemblies 690 on a general purpose CPU such as a microcomputer, minicomputer, mainframe computer or supercomputer processor. In the preferred embodiment, the
20 overall software architecture of an SPE 503 may be the same as the software architecture of an HPE 655. An HPE 655 can "emulate" SPE 503 and associated SPU 500, i.e., each may

include services and resources needed to support an identical set of service requests from ROS 602 (although ROS 602 may be restricted from sending to an HPE certain highly secure tasks to be executed only within an SPU 500).

5

Some electronic appliance 600 configurations might include both an SPE 503 and an HPE 655. For example, the HPE 655 could perform tasks that need lesser (or no) security protections, and the SPE 503 could perform all tasks that require a high
10 degree of security. This ability to provide serial or concurrent processing using multiple SPE and/or HPE arrangements provides additional flexibility, and may overcome limitations imposed by limited resources that can practically or cost-effectively be provided within an SPU 500. The cooperation of
15 an SPE 503 and an HPE 655 may, in a particular application, lead to a more efficient and cost effective but nevertheless secure overall processing environment for supporting and providing the secure processing required by VDE 100. As one example, an
20 HPE 655 could provide overall processing for allowing a user to manipulate released object 300 'contents,' but use SPE 503 to access the secure object and release the information from the object.

Figure 13 shows the software architecture of the preferred embodiment Secure Processing Environment (SPE) 503. This architecture may also apply to the preferred embodiment Host Processing Environment (HPE) 655. "Protected Processing Environment" ("PPE") 650 may refer generally to SPE 503 and/or HPE 655. Hereinafter, unless context indicates otherwise, references to any of "PPE 650," "HPE 655" and "SPE 503" may refer to each of them.

As shown in Figure 13, SPE 503 (PPE 650) includes the following service managers/major functional blocks in the preferred embodiment:

Kernel/Dispatcher 552

- C Channel Services Manager 562
- 15 C SPE RPC Manager 550
- C Time Base Manager 554
- C Encryption/Decryption Manager 556
- C Key and Tag Manager 558
- C Summary Services Manager 560
- 20 C Authentication Manager/Service Communications
Manager 564
- C Random Value Generator 565

C Secure Database Manager 566

C Other Services 592.

5 Each of the major functional blocks of PPE 650 is discussed
in detail below.

I. SPE Kernel/Dispatcher 552

10 The Kernel/Dispatcher 552 provides an operating system
"kernel" that runs on and manages the hardware resources of
SPU 500. This operating system "kernel" 552 provides a self-
15 contained operating system for SPU 500; it is also a part of
overall ROS 602 (which may include multiple OS kernels,
including one for each SPE and HPE ROS is
controlling/managing). Kernel/dispatcher 552 provides SPU task
15 and memory management, supports internal SPU hardware
interrupts, provides certain "low level services," manages "DTD"
data structures, and manages the SPU bus interface unit 530.
Kernel/dispatcher 552 also includes a load module execution
manager 568 that can load programs into secure execution space
20 for execution by SPU 500.

In the preferred embodiment, kernel/dispatcher 552 may include the following software/functional components:

load module execution manager 568

task manager 576

5 memory manager 578

virtual memory manager 580

"low level" services manager 582

internal interrupt handlers 584

BIU handler 586 (may not be present in HPE 655)

10 Service interrupt queues 588

DTD Interpreter 590.

At least parts of the kernel/dispatcher 552 are preferably stored in SPU firmware loaded into SPU ROM 532. An example
15 of a memory map of SPU ROM 532 is shown in Figure 14A. This memory map shows the various components of kernel/dispatcher 552 (as well as the other SPE services shown in Figure 13) residing in SPU ROM 532a and/or EEPROM 532b. The Figure 14B example of an NVRAM 534b memory map shows the task
20 manager 576 and other information loaded into NVRAM.

One of the functions performed by kernel/dispatcher 552 is to receive RPC calls from ROS RPC manager 732. As explained above, the ROS Kernel RPC manager 732 can route RPC calls to the SPE 503 (via SPE Device Driver 736 and its associated RSI 736a) for action by the SPE. The SPE kernel/dispatcher 552 receives these calls and either handles them or passes them on to SPE RPC manager 550 for routing internally to SPE 503. SPE 503 based processes can also generate RPC requests. Some of these requests can be processed internally by the SPE 503. If they are not internally serviceable, they may be passed out of the SPE 503 through SPE kernel/dispatcher 552 to ROS RPC manager 732 for routing to services external to SPE 503.

A. Kernel/Dispatcher Task Management

Kernel/dispatcher task manager 576 schedules and oversees tasks executing within SPE 503 (PPE 650). SPE 503 supports many types of tasks. A "channel" (a special type of task that controls execution of component assemblies 690 in the preferred embodiment) is treated by task manager 576 as one type of task. Tasks are submitted to the task manager 576 for execution. Task manager 576 in turn ensures that the SPE 503/SPU 500 resources necessary to execute the tasks are made

available, and then arranges for the SPU microprocessor 520 to execute the task.

Any call to kernel/dispatcher 552 gives the kernel an opportunity to take control of SPE 503 and to change the task or tasks that are currently executing. Thus, in the preferred embodiment kernel/dispatcher task manager 576 may (in conjunction with virtual memory manager 580 and/or memory manager 578) "swap out" of the execution space any or all of the tasks that are currently active, and "swap in" additional or different tasks.

SPE tasking managed by task manager 576 may be either "single tasking" (meaning that only one task may be active at a time) or "multi-tasking" (meaning that multiple tasks may be active at once). SPE 503 may support single tasking or multi-tasking in the preferred embodiment. For example, "high end" implementations of SPE 503 (e.g., in server devices) should preferably include multi-tasking with "preemptive scheduling." Desktop applications may be able to use a simpler SPE 503, although they may still require concurrent execution of several tasks. Set top applications may be able to use a relatively simple

implementation of SPE 503, supporting execution of only one task at a time. For example, a typical set top implementation of SPU 500 may perform simple metering, budgeting and billing using subsets of VDE methods combined into single "aggregate" load modules to permit the various methods to execute in a single tasking environment. However, an execution environment that supports only single tasking may limit use with more complex control structures. Such single tasking versions of SPE 503 trade flexibility in the number and types of metering and budgeting operations for smaller run time RAM size requirements. Such implementations of SPE 503 may (depending upon memory limitations) also be limited to metering a single object 300 at a time. Of course, variations or combinations are possible to increase capabilities beyond a simple single tasking environment without incurring the additional cost required to support "full multitasking."

In the preferred embodiment, each task in SPE 503 is represented by a "swap block," which may be considered a "task" in a traditional multitasking architecture. A "swap block" in the preferred embodiment is a bookkeeping mechanism used by task manager 576 to keep track of tasks and subtasks. It corresponds

to a chunk of code and associated references that "fits" within the secure execution environment provided by SPU 500. In the preferred embodiment, it contains a list of references to shared data elements (e.g., load modules 1100 and UDEs 1200), private data elements (e.g., method data and local stack), and swapped process "context" information (e.g., the register set for the process when it is not processing). Figure 14C shows an example of a snapshot of SPU RAM 532 storing several examples of "swap blocks" for a number of different tasks/methods such as a "channel" task, a "control" task, an "event" task, a "meter" task, a "budget" task, and a "billing" task. Depending on the size of SPU RAM 532, "swap blocks" may be swapped out of RAM and stored temporarily on secondary storage 652 until their execution can be continued. Thus, SPE 503 operating in a multi-tasking mode may have one or more tasks "sleeping." In the simplest form, this involves an active task that is currently processing, and another task (e.g., a control task under which the active task may be running) that is "sleeping" and is "swapped out" of active execution space. Kernel/dispatcher 522 may swap out tasks at any time.

Task manager 576 may use Memory Manager 578 to help it perform this swapping operation. Tasks may be swapped out of the secure execution space by reading appropriate information from RAM and other storage internal to SPU 500, for example, and writing a "swap block" to secondary storage 652. Kernel 552 may swap a task back into the secure execution space by reading the swap block from secondary storage 652 and writing the appropriate information back into SPU RAM 532. Because secondary storage 652 is not secure, SPE 503 must encrypt and cryptographically seal (e.g., using a one-way hash function initialized with a secret value known only inside the SPU 500) each swap block before it writes it to secondary storage. The SPE 503 must decrypt and verify the cryptographic seal for each swap block read from secondary storage 652 before the swap block can be returned to the secure execution space for further execution.

Loading a "swap block" into SPU memory may require one or more "paging operations" to possibly first save, and then flush, any "dirty pages" (i.e., pages changed by SPE 503) associated with the previously loaded swap blocks, and to load all required pages for the new swap block context.

Kernel/dispatcher 522 preferably manages the "swap blocks" using service interrupt queues 588. These service interrupt queues 588 allow kernel/dispatcher 552 to track tasks (swap blocks) and their status (running, "swapped out," or "asleep"). The kernel/dispatcher 552 in the preferred embodiment may maintain the following service interrupt queues 588 to help it manage the "swap blocks":

RUN queue
SWAP queue
SLEEP queue.

Those tasks that are completely loaded in the execution space and are waiting for and/or using execution cycles from microprocessor 502 are in the RUN queue. Those tasks that are "swapped" out (e.g., because they are waiting for other swappable components to be loaded) are referenced in the SWAP queue. Those tasks that are "asleep" (e.g., because they are "blocked" on some resource other than processor cycles or are not needed at the moment) are referenced in the SLEEP queue. Kernel/dispatcher task manager 576 may, for example, transition tasks between the RUN and SWAP queues based upon a "round-robin" scheduling algorithm that selects the next task waiting for service, swaps in any pieces that need to be paged in, and

executes the task. Kernel/dispatcher 552 task manager 576 may transition tasks between the SLEEP queue and the "awake" (i.e., RUN or SWAP) queues as needed.

5 When two or more tasks try to write to the same data structure in a multi-tasking environment, a situation exists that may result in "deadly embrace" or "task starvation." A "multi-threaded" tasking arrangement may be used to prevent "deadly embrace" or "task starvation" from happening. The preferred
10 embodiment kernel/dispatcher 552 may support "single threaded" or "multi-threaded" tasking.

 In single threaded applications, the kernel/dispatcher 552 "locks" individual data structures as they are loaded. Once
15 locked, no other SPE 503 task may load them and will "block" waiting for the data structure to become available. Using a single threaded SPE 503 may, as a practical matter, limit the ability of outside vendors to create load modules 1100 since there can be no assurance that they will not cause a "deadly embrace"
20 with other VDE processes about which outside vendors may know little or nothing. Moreover, the context swapping of a partially updated record might destroy the integrity of the

system, permit unmetered use, and/or lead to deadlock. In addition, such "locking" imposes a potentially indeterminate delay into a typically time critical process, may limit SPE 503 throughput, and may increase overhead.

5

This issue notwithstanding, there are other significant processing issues related to building single-threaded versions of SPE 503 that may limit its usefulness or capabilities under some circumstances. For example, multiple concurrently executing tasks may not be able to process using the same often-needed data structure in a single-threaded SPE 503. This may effectively limit the number of concurrent tasks to one.

10

Additionally, single-threadedness may eliminate the capability of producing accurate summary budgets based on a number of concurrent tasks since multiple concurrent tasks may not be able to effectively share the same summary budget data structure.

15

Single-threadedness may also eliminate the capability to support audit processing concurrently with other processing. For example, real-time feed processing might have to be shut down in order to audit budgets and meters associated with the monitoring process.

20

One way to provide a more workable "single-threaded" capability is for kernel/dispatcher 552 to use virtual page handling algorithms to track "dirty pages" as data areas are written to. The "dirty pages" can be swapped in and out with the task swap block as part of local data associated with the swap block. When a task exits, the "dirty pages" can be merged with the current data structure (possibly updated by another task for SPU 500) using a three-way merge algorithm (i.e., merging the original data structure, the current data structure, and the "dirty pages" to form a new current data structure). During the update process, the data structure can be locked as the pages are compared and swapped. Even though this virtual paging solution might be workable for allowing single threading in some applications, the vendor limitations mentioned above may limit the use of such single threaded implementations in some cases to dedicated hardware. Any implementation that supports multiple users (e.g., "smart home" set tops, many desk tops and certain PDA applications, etc.) may hit limitations of a single threaded device in certain circumstances.

20

It is preferable when these limitations are unacceptable to use a full "multi-threaded" data structure write capabilities. For

example, a type of "two-phase commit" processing of the type used by database vendors may be used to allow data structure sharing between processes. To implement this "two-phase commit" process, each swap block may contain page addresses for additional memory blocks that will be used to store changed information. A change page is a local copy of a piece of a data element that has been written by an SPE process. The changed page(s) references associated with a specific data structure are stored locally to the swap block in the preferred embodiment.

10

For example, SPE 503 may support two (change pages) per data structure. This limit is easily alterable by changing the size of the swap block structure and allowing the update algorithm to process all of the changed pages. The "commit" process can be invoked when a swap block that references changed pages is about to be discarded. The commit process takes the original data element that was originally loaded (e.g., UDE_0), the current data element (e.g., UDE_n) and the changed pages, and merges them to create a new copy of the data element (e.g., UDE_{n+1}). Differences can be resolved by the DTD interpreter 590 using a DTD for the data element. The original data element is

15

20

discarded (e.g., as determined by its DTD use count) if no other swap block references it.

B. Kernel/Dispatcher Memory Management

5 Memory manager 578 and virtual memory manager 580 in the preferred embodiment manage ROM 532 and RAM 534 memory within SPU 500 in the preferred embodiment. Virtual memory manager 580 provides a fully "virtual" memory system to increase the amount of "virtual" RAM available in the SPE
10 secure execution space beyond the amount of physical RAM 534a provided by SPU 500. Memory manager 578 manages the memory in the secure execution space, controlling how it is accessed, allocated and deallocated. SPU MMU 540, if present, supports virtual memory manager 580 and memory manager 578
15 in the preferred embodiment. In some "minimal" configurations of SPU 500 there may be no virtual memory capability and all memory management functions will be handled by memory manager 578. Memory management can also be used to help enforce the security provided by SPE 503. In some classes of
20 SPUs 500, for example, the kernel memory manager 578 may use hardware memory management unit (MMU) 540 to provide page level protection within the SPU 500. Such a hardware-based

memory management system provides an effective mechanism for protecting VDE component assemblies 690 from compromise by "rogue" load modules.

5 In addition, memory management provided by memory manager 578 operating at least in part based on hardware-based MMU 540 may securely implement and enforce a memory architecture providing multiple protection domains. In such an architecture, memory is divided into a plurality of domains that
10 are largely isolated from each other and share only specific memory areas under the control of the memory manager 578. An executing process cannot access memory outside its domain and can only communicate with other processes through services provided by and mediated by privileged kernel/dispatcher
15 software 552 within the SPU 500. Such an architecture is more secure if it is enforced at least in part by hardware within MMU 540 that cannot be modified by any software-based process executing within SPU 500.

20 In the preferred embodiment, access to services implemented in the ROM 532 and to physical resources such as NVRAM 534b and RTC 528 are mediated by the combination of

privileged kernel/dispatcher software 552 and hardware within MMU 540. ROM 532 and RTC 528 requests are privileged in order to protect access to critical system component routines (e.g., RTC 528).

5

Memory manager 578 is responsible for allocating and deallocating memory; supervising sharing of memory resources between processes; and enforcing memory access/use restriction. The SPE kernel/dispatcher memory manager 578 typically

10 initially allocates all memory to kernel 552, and may be configured to permit only process-level access to pages as they are loaded by a specific process. In one example SPE operating system configuration, memory manager 578 allocates memory using a simplified allocation mechanism. A list of each memory

15 page accessible in SPE 503 may be represented using a bit map allocation vector, for example. In a memory block, a group of contiguous memory pages may start at a specific page number. The size of the block is measured by the number of memory pages it spans. Memory allocation may be recorded by setting/clearing

20 the appropriate bits in the allocation vector.

To assist in memory management functions, a "dope vector" may be prepended to a memory block. The "dope vector" may contain information allowing memory manager 578 to manage that memory block. In its simplest form, a memory block may be structured as a "dope vector" followed by the actual memory area of the block. This "dope vector" may include the block number, support for dynamic paging of data elements, and a marker to detect memory overwrites. Memory manager 578 may track memory blocks by their block number and convert the block number to an address before use. All accesses to the memory area can be automatically offset by the size of the "dope vector" during conversion from a block memory to a physical address. "Dope vectors" can also be used by virtual memory manager 580 to help manage virtual memory.

15

The ROM 532 memory management task performed by memory manager 578 is relatively simple in the preferred embodiment. All ROM 532 pages may be flagged as "read only" and as "non-pagable." EEPROM 532B memory management may be slightly more complex since the "burn count" for each EEPROM page may need to be retained. SPU EEPROM 532B may need to be protected from all uncontrolled writes to conserve

20

the limited writable lifetime of certain types of this memory. Furthermore, EEPROM pages may in some cases not be the same size as memory management address pages.

5 SPU NVRAM 534b is preferably battery backed RAM that has a few access restrictions. Memory manager 578 can ensure control structures that must be located in NVRAM 534b are not relocated during "garbage collection" processes. As discussed above, memory manager 578 (and MMU 540 if present) may
10 protect NVRAM 534b and RAM 534a at a page level to prevent tampering by other processes.

Virtual memory manager 580 provides paging for programs and data between SPU external memory and SPU
15 internal RAM 534a. It is likely that data structures and executable processes will exceed the limits of any SPU 500 internal memory. For example, PERCs 808 and other fundamental control structures may be fairly large, and "bit map
meters" may be, or become, very large. This eventuality may be
20 addressed in two ways:

- (1) subdividing load modules 1100; and
- (2) supporting virtual paging.

Load modules 1100 can be "subdivided" in that in many instances they can be broken up into separate components only a subset of which must be loaded for execution. Load modules 1100 are the smallest pagable executable element in this example.

5 Such load modules 1100 can be broken up into separate components (e.g., executable code and plural data description blocks), only one of which must be loaded for simple load modules to execute. This structure permits a load module 1100 to initially
10 load only the executable code and to load the data description blocks into the other system pages on a demand basis. Many load modules 1100 that have executable sections that are too large to fit into SPU 500 can be restructured into two or more smaller independent load modules. Large load modules may be manually "split" into multiple load modules that are "chained"
15 together using explicit load module references.

Although "demand paging" can be used to relax some of these restrictions, the preferred embodiment uses virtual paging to manage large data structures and executables. Virtual
20 Memory Manager 580 "swaps" information (e.g., executable code and/or data structures) into and out of SPU RAM 534a, and provides other related virtual memory management services to

allow a full virtual memory management capability. Virtual memory management may be important to allow limited resource SPU 500 configurations to execute large and/or multiple tasks.

5 **C. SPE Load Module Execution Manager 568**

The SPE (HPE) load module execution manager ("LMEM") 568 loads executables into the memory managed by memory manager 578 and executes them. LMEM 568 provides mechanisms for tracking load modules that are currently loaded
10 inside the protected execution environment. LMEM 568 also provides access to basic load modules and code fragments stored within, and thus always available to, SPE 503. LMEM 568 may be called, for example, by load modules 1100 that want to execute other load modules.

15

In the preferred embodiment, the load module execution manager 568 includes a load module executor ("program loader") 570, one or more internal load modules 572, and library routines 574. Load module executor 570 loads executables into memory
20 (e.g., after receiving a memory allocation from memory manager 578) for execution. Internal load module library 572 may provide a set of commonly used basic load modules 1100 (stored in ROM

532 or NVRAM 534b, for example). Library routines 574 may provide a set of commonly used code fragments/routines (e.g., bootstrap routines) for execution by SPE 503.

5 Library routines 574 may provide a standard set of library functions in ROM 532. A standard list of such library functions along with their entry points and parameters may be used. Load modules 1100 may call these routines (e.g., using an interrupt reserved for this purpose). Library calls may reduce the size of
10 load modules by moving commonly used code into a central location and permitting a higher degree of code reuse. All load modules 1100 for use by SPE 503 are preferably referenced by a load module execution manager 568 that maintains and scans a list of available load modules and selects the appropriate load
15 module for execution. If the load module is not present within SPE 503, the task is "slept" and LMEM 568 may request that the load module 1100 be loaded from secondary storage 562. This request may be in the form of an RPC call to secure database manager 566 to retrieve the load module and associated data
20 structures, and a call to encrypt/decrypt manager 556 to decrypt the load module before storing it in memory allocated by memory manager 578.

In somewhat more detail, the preferred embodiment executes a load module 1100 by passing the load module execution manager 568 the name (e.g., VDE ID) of the desired load module 1100. LMEM 568 first searches the list of "in
5 memory" and "built-in" load modules 572. If it cannot find the desired load module 1100 in the list, it requests a copy from the secure database 610 by issuing an RPC request that may be handled by ROS secure database manager 744 shown in Figure 12. Load module execution manager 568 may then request
10 memory manager 578 to allocate a memory page to store the load module 1100. The load module execution manager 568 may copy the load module into that memory page, and queue the page for decryption and security checks by encrypt/decrypt manager 556 and key and tag manager 558. Once the page is decrypted and
15 checked, the load module execution manager 568 checks the validation tag and inserts the load module into the list of paged in modules and returns the page address to the caller. The caller may then call the load module 1100 directly or allow the load module execution module 570 to make the call for it.

20

Figure 15a shows a detailed example of a possible format for a channel header 596 and a channel 594 containing channel

5 detail records 594(1), 594(2), . . . 594(N). Channel header 596
may include a channel ID field 597(1), a user ID field 597(2), an
object ID field 597(3), a field containing a reference or other
identification to a "right" (i.e., a collection of events supported by
10 methods referenced in a PERC 808 and/or "user rights table" 464)
597(4), an event queue 597(5), and one or more fields 598 that
cross-reference particular event codes with channel detail records
("CDRs"). Channel header 596 may also include a "jump" or
reference table 599 that permits addressing of elements within
15 an associated component assembly or assemblies 690. Each CDR
594(1), . . . 594(N) corresponds to a specific event (event code) to
which channel 594 may respond. In the preferred embodiment,
these CDRs may include explicitly and/or by reference each
method core 1000N (or fragment thereof), load module 1100 and
20 data structure(s), (e.g., URT, UDE 1200 and/or MDE 1202)
needed to process the corresponding event. In the preferred
embodiment, one or more of the CDRs (e.g., 594(1)) may reference
a control method and a URT 464 as a data structure.

20 Figure 15b shows an example of program control steps
performed by SPE 503 to "open" a channel 594 in the preferred
embodiment. In the preferred embodiment, a channel 594

provides event processing for a particular VDE object 300, a particular authorized user, and a particular "right" (i.e., type of event). These three parameters may be passed to SPE 503. Part of SPE kernel/dispatcher 552 executing within a "channel 0" constructed by low level services 582 during a "bootstrap" routine may respond initially to this "open channel" event by allocating an available channel supported by the processing resources of SPE 503 (block 1125). This "channel 0" "open channel" task may then issue a series of requests to secure database manager 566 to obtain the "blueprint" for constructing one or more component assemblies 690 to be associated with channel 594 (block 1127). In the preferred embodiment, this "blueprint" may comprise a PERC 808 and/or URT 464. It may be obtained by using the "Object, User, Right" parameters passed to the "open channel" routine to "chain" together object registration table 460 records, user/object table 462 records, URT 464 records, and PERC 808 records. This "open channel" task may preferably place calls to key and tag manager 558 to validate and correlate the tags associated with these various records to ensure that they are authentic and match. The preferred embodiment process then may write appropriate information to channel header 596 (block 1129). Such information may include, for example, User ID,

Object ID, and a reference to the "right" that the channel will process. The preferred embodiment process may next use the "blueprint" to access (e.g, the secure database manager 566 and/or from load module execution manager library(ies) 568) the appropriate "control method" that may be used to, in effect, supervise execution of all of the other methods 1000 within the channel 594 (block 1131). The process may next "bind" this control method to the channel (block 1133), which step may include binding information from a URT 464 into the channel as a data structure for the control method. The process may then pass an "initialization" event into channel 594 (block 1135). This "initialization" event may be created by the channel services manager 562, the process that issued the original call requesting a service being fulfilled by the channel being built, or the control method just bound to the channel could itself possibly generate an initialization event which it would in effect pass to itself.

In response to this "initialization" event, the control method may construct the channel detail records 594(1), . . . 594(N) used to handle further events other than the "initialization" event. The control method executing "within" the channel may access the various components it needs to construct

associated component assemblies 690 based on the "blueprint"
accessed at step 1127 (block 1137). Each of these components is
bound to the channel 594 (block 1139) by constructing an
associated channel detail record specifying the method core(s)
5 1000N, load module(s) 1100, and associated data structure(s) (e.g.,
UDE(s) 1200 and/or MDE(s) 1202) needed to respond to the
event. The number of channel detail records will depend on the
number of events that can be serviced by the "right," as specified
by the "blueprint" (i.e., URT 464). During this process, the
10 control method will construct "swap blocks" to, in effect, set up all
required tasks and obtain necessary memory allocations from
kernel 562. The control method will, as necessary, issue calls to
secure database manager 566 to retrieve necessary components
from secure database 610, issue calls to encrypt/decrypt manager
15 556 to decrypt retrieved encrypted information, and issue calls to
key and tag manager 558 to ensure that all retrieved components
are validated. Each of the various component assemblies 690 so
constructed are "bound" to the channel through the channel
header event code/pointer records 598 and by constructing
20 appropriate swap blocks referenced by channel detail records
594(1), . . . 594(N). When this process is complete, the channel
594 has been completely constructed and is ready to respond to

further events. As a last step, the Figure 15b process may, if desired, deallocate the "initialization" event task in order to free up resources.

5 Once a channel 594 has been constructed in this fashion, it will respond to events as they arrive. Channel services manager 562 is responsible for dispatching events to channel 594. Each time a new event arrives (e.g., via an RPC call), channel services manager 562 examines the event to determine whether a channel
10 already exists that is capable of processing it. If a channel does exist, then the channel services manager 562 passes the event to that channel. To process the event, it may be necessary for task manager 576 to "swap in" certain "swappable blocks" defined by the channel detail records as active tasks. In this way,
15 executable component assemblies 690 formed during the channel open process shown in Figure 15b are placed into active secure execution space, the particular component assembly that is activated being selected in response to the received event code. The activated task will then perform its desired function in
20 response to the event.

To destroy a channel, the various swap blocks defined by the channel detail records are destroyed, the identification information in the channel header 596 is wiped clean, and the channel is made available for re-allocation by the "channel 0" "open channel" task.

D. SPE Interrupt Handlers 584

As shown in Figure 13, kernel/dispatcher 552 also provides internal interrupt handler(s) 584. These help to manage the resources of SPU 500. SPU 500 preferably executes in either "interrupt" or "polling" mode for all significant components. In polling mode, kernel/dispatcher 552 may poll each of the sections/circuits within SPU 500 and emulate an interrupt for them. The following interrupts are preferably supported by SPU 500 in the preferred embodiment:

- C "tick" of RTC 528
- C interrupt from bus interface 530
- C power fail interrupt
- C watchdog timer interrupt
- C interrupt from encrypt/decrypt engine 522
- C memory interrupt (e.g., from MMU 540).

When an interrupt occurs, an interrupt controller within microprocessor 520 may cause the microprocessor to begin executing an appropriate interrupt handler. An interrupt handler is a piece of software/firmware provided by kernel/dispatcher 552 that allows microprocessor 520 to perform particular functions upon the occurrence of an interrupt. The interrupts may be "vectored" so that different interrupt sources may effectively cause different interrupt handlers to be executed.

10 A "timer tick" interrupt is generated when the real-time RTC 528 "pulses." The timer tick interrupt is processed by a timer tick interrupt handler to calculate internal device date/time and to generate timer events for channel processing.

15 The bus interface unit 530 may generate a series of interrupts. In the preferred embodiment, bus interface 530, modeled after a USART, generates interrupts for various conditions (e.g., "receive buffer full," "transmitter buffer empty," and "status word change"). Kernel/dispatcher 552 services the transmitter buffer empty interrupt by sending the next character
20 from the transmit queue to the bus interface 530. Kernel/dispatcher interrupt handler 584 may service the received

buffer full interrupt by reading a character, appending it to the current buffer, and processing the buffer based on the state of the service engine for the bus interface 530. Kernel/dispatcher 552 preferably processes a status word change interrupt and
5 addresses the appropriate send/receive buffers accordingly.

SPU 500 generates a power fail interrupt when it detects an imminent power fail condition. This may require immediate action to prevent loss of information. For example, in the
10 preferred embodiment, a power fail interrupt moves all recently written information (i.e., "dirty pages") into non-volatile NVRAM 534b, marks all swap blocks as "swapped out," and sets the appropriate power fail flag to facilitate recovery processing. Kernel/dispatcher 552 may then periodically poll the "power fail
15 bit" in a status word until the data is cleared or the power is removed completely.

SPU 500 in the example includes a conventional watchdog timer that generates watchdog timer interrupts on a regular
20 basis. A watchdog timer interrupt handler performs internal device checks to ensure that tampering is not occurring. The internal clocks of the watchdog timer and RTC 528 are compared

to ensure SPU 500 is not being paused or probed, and other internal checks on the operation of SPU 500 are made to detect tampering.

5 The encryption/decryption engine 522 generates an interrupt when a block of data has been processed. The kernel interrupt handler 584 adjusts the processing status of the block being encrypted or decrypted, and passes the block to the next stage of processing. The next block scheduled for the encryption
10 service then has its key moved into the encrypt/decrypt engine 522, and the next cryptographic process started.

 A memory management unit 540 interrupt is generated when a task attempts to access memory outside the areas
15 assigned to it. A memory management interrupt handler traps the request, and takes the necessary action (e.g., by initiating a control transfer to memory manager 578 and/or virtual memory manager 580). Generally, the task will be failed, a page fault exception will be generated, or appropriate virtual memory
20 page(s) will be paged in.

E. Kernel/Dispatcher Low Level Services 582

Low level services 582 in the preferred embodiment provide "low level" functions. These functions in the preferred embodiment may include, for example, power-on initialization, device POST, and failure recovery routines. Low level services 582 may also in the preferred embodiment provide (either by themselves or in combination with authentication manager/service communications manager 564) download response-challenge and authentication communication protocols, and may provide for certain low level management of SPU 500 memory devices such as EEPROM and FLASH memory (either alone or in combination with memory manager 578 and/or virtual memory manager 580).

F. Kernel/Dispatcher BIU handler 586

BIU handler 586 in the preferred embodiment manages the bus interface unit 530 (if present). It may, for example, maintain read and write buffers for the BIU 530, provide BIU startup initialization, etc.

G. Kernel/Dispatcher DTD Interpreter 590

DTD interpreter 590 in the preferred embodiment handles data formatting issues. For example, the DTD interpreter 590 may automatically open data structures such as UDEs 1200
5 based on formatting instructions contained within DTDs.

The SPE kernel/dispatcher 552 discussed above supports all of the other services provided by SPE 503. Those other services are discussed below.

10

II. SPU Channel Services Manager 562

"Channels" are the basic task processing mechanism of SPE 503 (HPE 655) in the preferred embodiment. ROS 602 provides an event-driven interface for "methods." A "channel" allows component assemblies 690 to service events. A "channel" is a conduit for passing "events" from services supported by SPE 503 (HPE 655) to the various methods and load modules that have been specified to process these events, and also supports the assembly of component assemblies 690 and interaction between component assemblies. In more detail, "channel" 594 is a data structure maintained by channel manager 593 that "binds" together one or more load modules 1100 and data structures (e.g.,
15
20

UDEs 1200 and/or MDEs 1202) into a component assembly 690. Channel services manager 562 causes load module execution manager 569 to load the component assembly 690 for execution, and may also be responsible for passing events into the channel 594 for response by a component assembly 690. In the preferred embodiment, event processing is handled as a message to the channel service manager 562.

Figure 15 is a diagram showing how the preferred embodiment channel services manager 562 constructs a "channel" 594, and also shows the relationship between the channel and component assemblies 690. Briefly, the SPE channel manager 562 establishes a "channel" 594 and an associated "channel header" 596. The channel 594 and its header 596 comprise a data structure that "binds" or references elements of one or more component assemblies 690. Thus, the channel 594 is the mechanism in the preferred embodiment that collects together or assembles the elements shown in Figure 11E into a component assembly 690 that may be used for event processing.

20

The channel 594 is set up by the channel services manager 562 in response to the occurrence of an event. Once the channel

is created, the channel services manager 562 may issue function calls to load module execution manager 568 based on the channel 594. The load module execution manager 568 loads the load modules 1100 referenced by a channel 594, and requests
5 execution services by the kernel/dispatcher task manager 576. The kernel/dispatcher 552 treats the event processing request as a task, and executes it by executing the code within the load modules 1100 referenced by the channel.

10 The channel services manager 562 may be passed an identification of the event (e.g., the "event code"). The channel services manager 562 parses one or more method cores 1000' that are part of the component assembly(ies) 690 the channel services manager is to assemble. It performs this parsing to determine
15 which method(s) and data structure(s) are invoked by the type of event. Channel manager 562 then issues calls (e.g., to secure database manager 566) to obtain the methods and data structure(s) needed to build the component assembly 690. These called-for method(s) and data structure(s) (e.g., load modules
20 1100, UDEs 1200 and/or MDEs 1202) are each decrypted using encrypt/decrypt manager 556 (if necessary), and are then each validated using key and tag manager 558. Channel manager 562

constructs any necessary "jump table" references to, in effect,
"link" or "bind" the elements into a single cohesive executable so
the load module(s) can reference data structures and any other
load module(s) in the component assembly. Channel manager
5 562 may then issue calls to LMEM 568 to load the executable as
an active task.

Figure 15 shows that a channel 594 may reference another
channel. An arbitrary number of channels 594 may be created by
10 channel manager 594 to interact with one another.

"Channel header" 596 in the preferred embodiment is (or
references) the data structure(s) and associated control
program(s) that queues events from channel event sources,
15 processes these events, and releases the appropriate tasks
specified in the "channel detail record" for processing. A "channel
detail record" in the preferred embodiment links an event to a
"swap block" (i.e., task) associated with that event. The "swap
block" may reference one or more load modules 1100, UDEs 1200
20 and private data areas required to properly process the event.
One swap block and a corresponding channel detail item is
created for each different event the channel can respond to.

In the preferred embodiment, Channel Services Manager 562 may support the following (internal) calls to support the creation and maintenance of channels 562:

5

Call Name	Source	Description
"Write Event"	Write	Writes an event to the channel for response by the channel. The <u>Write Event</u> call thus permit the caller to insert an event into the event queue associated with the channel. The event will be processed in turn by the channel 594.
"Bind Item"	Ioctl	Binds an item to a channel with the appropriate processing algorithm. The <u>Bind Item</u> call permits the caller to bind a VDE item ID to a channel (e.g., to create one or more swap blocks associated with a channel). This call may manipulate the contents of individual swap blocks.
"Unbind Item"	Ioctl	Unbinds an item from a channel with the appropriate processing algorithm. The <u>Unbind Item</u> call permits the caller to break the binding of an item to a swap block. This call may manipulate the contents of individual swap blocks.

10

SPE RPC Manager 550

As described in connection with Figure 12, the architecture of ROS 602 is based on remote procedure calls in the preferred embodiment. ROS 602 includes an RPC Manager 732 that passes RPC calls between services each of which present an RPC service interface ("RSI") to the RPC manager. In the preferred embodiment, SPE 503 (HPE 655) is also built around the same RPC concept. The SPE 503 (HPE 655) may include a number of internal modular service providers each presenting an RSI to an RPC manager 550 internal to the SPE (HPE). These internal service providers may communicate with each other and/or with ROS RPC manager 732 (and thus, with any other service provided by ROS 602 and with external services), using RPC service requests.

RPC manager 550 within SPE 503 (HPE 655) is not the same as RPC manager 732 shown in Figure 12, but it performs a similar function within the SPE (HPE): it receives RPC requests and passes them to the RSI presented by the service that is to fulfill the request. In the preferred embodiment, requests are passed between ROS RPC manager 732 and the outside world (i.e., SPE device driver 736) via the SPE (HPE)

Kernel/Dispatcher 552. Kernel/Dispatcher 552 may be able to service certain RPC requests itself, but in general it passes received requests to RPC manager 550 for routing to the appropriate service internal to the SPE (HPE). In an alternate
5 embodiment, requests may be passed directly between the HPE, SPE, API, Notification interface, and other external services instead of routing them through the ROS RPC manager 732. The decision on which embodiment to use is part of the scalability of the system; some embodiments are more efficient than others
10 under various traffic loads and system configurations. Responses by the services (and additional service requests they may themselves generate) are provided to RPC Manager 550 for routing to other service(s) internal or external to SPE 503 (HPE 655).

15

SPE RPC Manager 550 and its integrated service manager uses two tables to dispatch remote procedure calls: an RPC services table, and an optional RPC dispatch table. The RPC services table describes where requests for specific services are to
20 be routed for processing. In the preferred embodiment, this table is constructed in SPU RAM 534a or NVRAM 534b, and lists each RPC service "registered" within SPU 500. Each row of the RPC

services table contains a service ID, its location and address, and a control byte. In simple implementations, the control byte indicates only that the service is provided internally or externally. In more complex implementations, the control byte can indicate an instance of the service (e.g., each service may have multiple "instances" in a multi-tasking environment). ROS RPC manager 732 and SPE 503 may have symmetric copies of the RPC services table in the preferred embodiment. If an RPC service is not found in the RPC services table, SPE 503 may either reject it or pass it to ROS RPC manager 732 for service.

The SPE RPC manager 550 accepts the request from the RPC service table and processes that request in accordance with the internal priorities associated with the specific service. In SPE 503, the RPC service table is extended by an RPC dispatch table. The preferred embodiment RPC dispatch table is organized as a list of Load Module references for each RPC service supported internally by SPE 503. Each row in the table contains a load module ID that services the call, a control byte that indicates whether the call can be made from an external caller, and whether the load module needed to service the call is permanently resident in SPU 500. The RPC dispatch table may

be constructed in SPU ROM 532 (or EEPROM) when SPU firmware 508 is loaded into the SPU 500. If the RPC dispatch table is in EEPROM, it flexibly allows for updates to the services without load module location and version control issues.

5

In the preferred embodiment, SPE RPC manager 550 first references a service request against the RPC service table to determine the location of the service manager that may service the request. The RPC manager 550 then routes the service request to the appropriate service manager for action. Service requests are handled by the service manager within the SPE 503 using the RPC dispatch table to dispatch the request. Once the RPC manager 550 locates the service reference in the RPC dispatch table, the load module that services the request is called and loaded using the load module execution manager 568. The load module execution manager 568 passes control to the requested load module after performing all required context configuration, or if necessary may first issue a request to load it from the external management files 610.

20

SPU Time Base Manager 554

The time base manager 554 supports calls that relate to the real time clock ("RTC") 528. In the preferred embodiment, the time base manager 554 is always loaded and ready to respond to time based requests.

The table below lists examples of basic calls that may be supported by the time base manager 554:

10	Call Name	Description
Independent requests		
	Get Time	Returns the time (local, GMT, or ticks).
	Set time	Sets the time in the RTC 528. Access to this command may be restricted to a VDE administrator.
	Adjust time	Changes the time in the RTC 528. Access to this command may be restricted to a VDE administrator.
15	Set Time Parameter	Set GMT / local time conversion and the current and allowable magnitude of user adjustments to RTC 528 time.
Channel Services Manager Requests		
	Bind Time	Bind timer services to a channel as an event source.
20	Unbind Time	Unbind timer services from a channel as an event source.

10

Call Name	Description
Set Alarm	Sets an alarm notification for a specific time. The user will be notified by an alarm event at the time of the alarm. Parameters to this request determine the event, frequency, and requested processing for the alarm.
Clear Alarm	Cancels a requested alarm notification.

5

SPU Encryption/Decryption Manager 556

The Encryption/Decryption Manager 556 supports calls to the various encryption/decryption techniques supported by SPE 503/HPE 655. It may be supported by a hardware-based encryption/decryption engine 522 within SPU 500. Those encryption/decryption technologies not supported by SPU encrypt/decrypt engine 522 may be provided by encrypt/decrypt manager 556 in software. The primary bulk encryption/decryption load modules preferably are loaded at all times, and the load modules necessary for other algorithms are preferably paged in as needed. Thus, if the primary bulk encryption/decryption algorithm is DES, only the DES load modules need be permanently resident in the RAM 534a of SPE 503/HPE 655.

The following are examples of RPC calls supported by Encrypt/Decrypt Manager 556 in the preferred embodiment:

5	Call Name	Description
	PK Encrypt	Encrypt a block using a PK (public key) algorithm.
	PK Decrypt	Decrypt a block using a PK algorithm.
	DES Encrypt	Encrypt a block using DES.
10	DES Decrypt	Decrypt a block using DES.
	RC-4 Encrypt	Encrypt a block using the RC-4 (or other bulk encryption) algorithm.
	RC-4 Decrypt	Decrypt a block using the RC-4 (or other bulk encryption) algorithm.
15	Initialize DES Instance	Initialize DES instance to be used.
20	Initialize RC-4 Instance	Initialize RC-4 instance to be used.
	Initialize MD5 Instance	Initialize MD5 instance to be used.
25	Process MD5 Block	Process MD5 block.

The call parameters passed may include the key to be used; mode (encryption or decryption); any needed Initialization Vectors; the desired cryptographic operating (e.g., type of feedback); the identification of the cryptographic instance to be used; and the start address, destination address, and length of the block to be encrypted or decrypted.

SPU Key and Tag Manager 558

The SPU Key and Tag Manager 558 supports calls for key storage, key and management file tag look up, key convolution, and the generation of random keys, tags, and transaction numbers.

The following table shows an example of a list of SPE/HPE key and tag manager service 558 calls:

Call Name	Description
Key Requests	
Get Key	Retrieve the requested key.
Set Key	Set (store) the specified key.
Generate Key	Generate a key (pair) for a specified algorithm.
Generate Convolution Key	Generate a key using a specified convolution algorithm and algorithm parameter block.
Get Convolution Algorithm	Return the currently set (default) convolution parameters for a specific convolution algorithm.

	Set Convolution Algorithm	Sets the convolution parameters for a specific convolution algorithm (calling routine must provide tag to read returned contents).
	Tag Requests	
	Get Tag	Get the validation (or other) tag for a specific VDE Item ID.
5	Set Tag	Set the validation (or other) tag for a specific VDE Item ID to a known value.
	Calculate Hash Block Number	Calculate the "hash block number" for a specific VDE Item ID.
	Set Hash Parameters	Set the hash parameters and hash algorithm. Force resynchronization of the hash table.
	Get Hash Parameters	Retrieve the current hash parameters/algorithm.
10	Synchronize Management Files	Synchronize the management files and rebuild the hash block tables based on information found in the tables Reserved for VDE administrator.

Keys and tags may be securely generated within SPE 503 (HPE 655) in the preferred embodiment. The key generation algorithm is typically specific to each type of encryption supported. The generated keys may be checked for cryptographic weakness before they are used. A request for Key and Tag Manager 558 to generate a key, tag and/or transaction number preferably takes a length as its input parameter. It generates a random number (or other appropriate key value) of the requested length as its output.

The key and tag manager 558 may support calls to retrieve specific keys from the key storage areas in SPU 500 and any keys

stored external to the SPU. The basic format of the calls is to request keys by key type and key number. Many of the keys are periodically updated through contact with the VDE administrator, and are kept within SPU 500 in NVRAM 534b or
5 EEPROM because these memories are secure, updatable and non-volatile.

SPE 503/HPE 655 may support both Public Key type keys and Bulk Encryption type keys. The public key (PK) encryption
10 type keys stored by SPU 500 and managed by key and tag manager 558 may include, for example, a device public key, a device private key, a PK certificate, and a public key for the certificate. Generally, public keys and certificates can be stored externally in non-secured memory if desired, but the device
15 private key and the public key for the certificate should only be stored internally in an SPU 500 EEPROM or NVRAM 534b. Some of the types of bulk encryption keys used by the SPU 500 may include, for example, general-purpose bulk encryption keys, administrative object private header keys, stationary object
20 private header keys, traveling object private header keys, download/initialization keys, backup keys, trail keys, and management file keys.

As discussed above, preferred embodiment Key and Tag Manager 558 supports requests to adjust or convolute keys to make new keys that are produced in a deterministic way dependent on site and/or time, for example. Key convolution is an algorithmic process that acts on a key and some set of input parameter(s) to yield a new key. It can be used, for example, to increase the number of keys available for use without incurring additional key storage space. It may also be used, for example, as a process to "age" keys by incorporating the value of real-time RTC 528 as parameters. It can be used to make keys site specific by incorporating aspects of the site ID as parameters.

Key and Tag Manager 558 also provides services relating to tag generation and management. In the preferred embodiment, transaction and access tags are preferably stored by SPE 503 (HPE 655) in protected memory (e.g., within the NVRAM 534b of SPU 500). These tags may be generated by key and tag manager 558. They are used to, for example, check access rights to, validate and correlate data elements. For example, they may be used to ensure components of the secured data structures are not tampered with outside of the SPU 500.

Key and tag manager 558 may also support a trail transaction tag and a communications transaction tag.

SPU Summary Services Manager 560

5 SPE 503 maintains an audit trail in reprogrammable non-volatile memory within the SPU 500 and/or in secure database 610. This audit trail may consist of an audit summary of budget activity for financial purposes, and a security summary of SPU use. When a request is made to the SPU, it logs the request as
 10 having occurred and then notes whether the request succeeded or failed. All successful requests may be summed and stored by type in the SPU 500. Failure information, including the elements listed below, may be saved along with details of the failure:

15

Control Information Retained in an SPE on Access Failures	
Object ID	
User ID	
Type of failure	
Time of failure	

20

This information may be analyzed to detect cracking attempts or
 25 to determine patterns of usage outside expected (and budgeted)

norms. The audit trail histories in the SPU 500 may be retained until the audit is reported to the appropriate parties. This will allow both legitimate failure analysis and attempts to cryptoanalyze the SPU to be noted.

5

Summary services manager 560 may store and maintain this internal summary audit information. This audit information can be used to check for security breaches or other aspects of the operation of SPE 503. The event summaries may be maintained, analyzed and used by SPE 503 (HPE 655) or a VDE administrator to determine and potentially limit abuse of electronic appliance 600. In the preferred embodiment, such parameters may be stored in secure memory (e.g., within the NVRAM 534b of SPU 500).

10

15

There are two basic structures for which summary services are used in the preferred embodiment. One (the "event summary data structure") is VDE administrator specific and keeps track of events. The event summary structure may be maintained and audited during periodic contact with VDE administrators. The other is used by VDE administrators and/or distributors for overall budget. A VDE administrator may register for event

20

summaries and an overall budget summary at the time an electronic appliance 600 is initialized. The overall budget summary may be reported to and used by a VDE administrator in determining distribution of consumed budget (for example) in the case of corruption of secure management files 610.

Participants that receive appropriate permissions can register their processes (e.g., specific budgets) with summary services manager 560, which may then reserve protected memory space (e.g., within NVRAM 534b) and keep desired use and/or access parameters. Access to and modification of each summary can be controlled by its own access tag.

The following table shows an example of a list of PPE summary service manager 560 service calls:

15

Call Name	Description
Create summary info	Create a summary service if the user has a "ticket" that permits her to request this service.
Get value	Return the current value of the summary service. The caller must present an appropriate tag (and/or "ticket") to use this request.
Set value	Set the value of a summary service.

20

<p>Increment</p>	<p>Increment the specified summary service(e.g., a scalar meter summary data area). The caller must present an appropriate tag (and/or "ticket") to use this request.</p>
<p>Destroy</p>	<p>Destroy the specified summary service if the user has a tag and/or "ticket" that permits them to request this service.</p>

5 In the preferred embodiment, the event summary data structure uses a fixed event number to index into a look up table. The look up table contains a value that can be configured as a counter or a counter plus limit. Counter mode may be used by VDE administrators to determine device usage. The limit mode
 10 may be used to limit tampering and attempts to misuse the electronic appliance 600. Exceeding a limit will result in SPE 503 (HPE 655) refusing to service user requests until it is reset by a VDE administrator. Calls to the system wide event summary process may preferably be built into all load modules
 15 that process the events that are of interest.

The following table shows examples of events that may be separately metered by the preferred embodiment event summary data structure:

Event Type	
Successful Events	Initialization completed successfully.
	User authentication accepted.
	Communications established.
	Channel loads set for specified values.
	Decryption completed.
	Key information updated.
	New budget created or existing budget updated.
	New billing information generated or existing billing updated.
	New meter set up or existing meter updated.
	New PERC created or existing PERC updated.
	New objects registered.
	Administrative objects successfully processed.
	Audit processed successfully.
	All other events.
Failed Events	Initialization failed.
	Authentication failed.
	Communication attempt failed.
	Request to load a channel failed.
	Validation attempt unsuccessful.
	Link to subsidiary item failed correlation tag match.

5

	Authorization attempt failed.
	Decryption attempt failed.
	Available budget insufficient to complete requested procedure.
	Audit did not occur.
	Administrative object did not process correctly.
	Other failed events.

Another, "overall currency budget" summary data structure maintained by the preferred embodiment summary services manager 560 allows registration of VDE electronic appliance 600. The first entry is used for an overall currency budget consumed value, and is registered by the VDE administrator that first initializes SPE 503 (HPE 655). Certain currency consuming load modules and audit load modules that complete the auditing process for consumed currency budget may call the summary services manager 560 to update the currency consumed value. Special authorized load modules may have access to the overall currency summary, while additional summaries can be registered for by individual providers.

15

**SPE Authentication Manager/Service Communications
Manager 564**

The Authentication Manager/Service Communications
 5 Manager 564 supports calls for user password validation and
 "ticket" generation and validation. It may also support secure
 communications between SPE 503 and an external node or device
 (e.g., a VDE administrator or distributor). It may support the
 following examples of authentication-related service requests in
 10 the preferred embodiment:

Call Name	Description
User Services	
Create User	Creates a new user and stores Name Services Records (NSRs) for use by the Name Services Manager 752.
15 Authenticate User	Authenticates a user for use of the system. This request lets the caller authenticate as a specific user ID. Group membership is also authenticated by this request. The authentication returns a "ticket" for the user.
Delete User	Deletes a user's NSR and related records.
Ticket Services	
Generate Ticket	Generates a "ticket" for use of one or more services.
20 Authenticate Ticket	Authenticates a "ticket."

Not included in the table above are calls to the secure communications service. The secure communications service provided by manager 564 may provide (e.g., in conjunction with low-level services manager 582 if desired) secure communications based on a public key (or others) challenge-response protocol. This protocol is discussed in further detail elsewhere in this document. Tickets identify users with respect to the electronic appliance 600 in the case where the appliance may be used by multiple users. Tickets may be requested by and returned to VDE software applications through a ticket-granting protocol (e.g., Kerberos). VDE components may require tickets to be presented in order to authorize particular services.

15

SPE Secure Database Manager 566

Secure database manager 566 retrieves, maintains and stores secure database records within secure database 610 on memory external to SPE 503. Many of these secure database files 610 are in encrypted form. All secure information retrieved by secure database manager 566 therefore must be decrypted by encrypt/decrypt manager 556 before use. Secure information (e.g., records of use) produced by SPE 503 (HPE 655) which must

20

be stored external to the secure execution environment are also encrypted by encrypt/decrypt manager 556 before they are stored via secure database manager 566 in a secure database file 610.

5 For each VDE item loaded into SPE 503, Secure Database manager 566 in the preferred embodiment may search a master list for the VDE item ID, and then check the corresponding transaction tag against the one in the item to ensure that the item provided is the current item. Secure Database Manager 566
10 may maintain list of VDE item ID and transaction tags in a "hash structure" that can be paged into SPE 503 to quickly locate the appropriate VDE item ID. In smaller systems, a look up table approach may be used. In either case, the list should be structured as a pagable structure that allows VDE item ID to be
15 located quickly.

 The "hash based" approach may be used to sort the list into "hash buckets" that may then be accessed to provide more rapid and efficient location of items in the list. In the "hash based"
20 approach, the VDE item IDs are "hashed" through a subset of the full item ID and organized as pages of the "hashed" table. Each "hashed" page may contain the rest of the VDE item ID and

current transaction tag for each item associated with that page.

The "hash" table page number may be derived from the components of the VDE item ID, such as distribution ID, item ID, site ID, user ID, transaction tag, creator ID, type and/or version.

5 The hashing algorithm (both the algorithm itself and the parameters to be hashed) may be configurable by a VDE administrator on a site by site basis to provide optimum hash page use. An example of a hash page structure appears below:

10

Field
Hash Page Header
Distributor ID
Item ID
15 Site ID
User ID
Transaction Tag
Hash Page Entry
Creator ID
20 Item ID
Type
Version
Transaction Tag

25

In this example, each hash page may contain all of the VDE item IDs and transaction tags for items that have identical distributor ID, item ID, and user ID fields (site ID will be fixed for a given electronic appliance 600). These four pieces of
5 information may thus be used as hash algorithm parameters.

The "hash" pages may themselves be frequently updated, and should carry transaction tags that are checked each time a "hash" page is loaded. The transaction tag may also be updated
10 each time a "hash" page is written out.

As an alternative to the hash-based approach, if the number of updatable items is kept small (such as in a dedicated consumer electronic appliance 600), then assigning each
15 updatable item a unique sequential site record number as part of its VDE item ID may allow a look up table approach to be used. Only a small number of bytes of transaction tag are needed per item, and a table transaction tag for all frequently updatable items can be kept in protected memory such as SPU NVRAM
20 534b.

Random Value Generator Manager 565

Random Value Generator Manager 565 may generate random values. If a hardware-based SPU random value generator 542 is present, the Random Value Generator Manager
5 565 may use it to assist in generating random values.

Other SPE RPC Services 592

Other authorized RPC services may be included in SPU 500 by having them "register" themselves in the RPC Services
10 Table and adding their entries to the RPC Dispatch Table. For example, one or more component assemblies 690 may be used to provide additional services as an integral part of SPE 503 and its associated operating system. Requests to services not registered in these tables will be passed out of SPE 503 (HPE 655) for
15 external servicing.

SPE 503 Performance Considerations

Performance of SPE 503 (HPE 655) is a function of:

- C complexity of the component assemblies used
- 20 C number of simultaneous component assembly operations
- C amount of internal SPU memory available
- C speed of algorithm for block encryption/decryption

The complexity of component assembly processes along with the number of simultaneous component assembly processes is perhaps the primary factor in determining performance. These factors combine to determine the amount of code and data and must be resident in SPU 500 at any one time (the minimum device size) and thus the number of device size "chunks" the processes must be broken down into. Segmentation inherently increases run time size over simpler models. Of course, feature limited versions of SPU 500 may be implemented using significantly smaller amounts of RAM 534. "Aggregate" load modules as described above may remove flexibility in configuring VDE structures and also further limit the ability of participants to individually update otherwise separated elements, but may result in a smaller minimum device size. A very simple metering version of SPU 500 can be constructed to operate with minimal device resources.

The amount of RAM 534 internal to SPU 500 has more impact on the performance of the SPE 503 than perhaps any other aspect of the SPU. The flexible nature of VDE processes allows use of a large number of load modules, methods and user data elements. It is impractical to store more than a small

number of these items in ROM 532 within SPU 500. Most of the code and data structures needed to support a specific VDE process will need to be dynamically loaded into the SPU 500 for the specific VDE process when the process is invoked. The
5 operating system within SPU 500 then may page in the necessary VDE items to perform the process. The amount of RAM 534 within SPU 500 will directly determine how large any single VDE load module plus its required data can be, as well as the number of page swaps that will be necessary to run a VDE
10 process. The SPU I/O speed, encryption/decryption speed, and the amount of internal memory 532, 534 will directly affect the number of page swaps required in the device. Insecure external memory may reduce the wait time for swapped pages to be loaded into SPU 500, but will still incur substantial
15 encryption/decryption penalty for each page.

In order to maintain security, SPE 503 must encrypt and cryptographically seal each block being swapped out to a storage device external to a supporting SPU 500, and must similarly
20 decrypt, verify the cryptographic seal for, and validate each block as it is swapped into SPU 500. Thus, the data movement and

encryption/decryption overhead for each swap block has a very large impact on SPE performance.

5 The performance of an SPU microprocessor 520 may not significantly impact the performance of the SPE 503 it supports if the processor is not responsible for moving data through the encrypt/decrypt engine 522.

VDE Secure Database 610

10 VDE 100 stores separately deliverable VDE elements in a secure (e.g., encrypted) database 610 distributed to each VDE electronic appliance 610. The database 610 in the preferred embodiment may store and/or manage three basic classes of VDE items:

15 VDE objects,
VDE process elements, and
VDE data structures.

20 The following table lists examples of some of the VDE items stored in or managed by information stored in secure database 610:

5

Class		Brief Description
Objects	Content Objects	Provide a container for content.
	Administrative Objects	Provide a container for information used to keep VDE 100 operating.
	Traveling Objects	Provide a container for content and control information.
	Smart Objects	Provide a container for (user-specified) processes and data.
Process Elements	Method Cores	Provide a mechanism to relate events to control mechanisms and permissions.
	Load Modules ("LMs")	Secure (tamper-resistant) executable code.
	Method Data Elements ("MDEs")	Independently deliverable data structures used to control/customize methods.
Data Structures	Permissions Records ("PERCs")	Permissions to use objects; "blueprints" to build component assemblies.
	User Data Elements ("UDEs")	Basic data structure for storing information used in conjunction with load modules.
	Administrative Data Structures	Used by VDE node to maintain administrative information.

Each electronic appliance 600 may have an instance of a secure database 610 that securely maintains the VDE items. Figure 16 shows one example of a secure database 610. The secure database 610 shown in this example includes the following

5 VDE-protected items:

- C one or more PERCs 808;
- C methods 1000 (including static and dynamic method "cores" 1000, and MDEs 1202);
- C Static UDEs 1200a and Dynamic UDEs 1200b; and
- 10 C load modules 1100.

Secure database 610 may also include the following additional data structures used and maintained for administrative purposes:

- 15 C an "object registry" 450 that references an object storage 728 containing one or more VDE objects;
- C name service records 452; and
- C configuration records 454 (including site configuration records 456 and user configuration records 458).
- 20

Secure database 610 in the preferred embodiment does not include VDE objects 300, but rather references VDE objects stored, for example, on file system 687 and/or in a separate object repository 728. Nevertheless, an appropriate "starting point" for understanding VDE-protected information may be a discussion of VDE objects 300.

VDE Objects 300

VDE 100 provides a media independent container model for encapsulating content. Figure 17 shows an example of a "logical" structure or format 800 for an object 300 provided by the preferred embodiment.

The generalized "logical object" structure 800 shown in Figure 17 used by the preferred embodiment supports digital content delivery over any currently used media. "Logical object" in the preferred embodiment may refer collectively to: content; computer software and/or methods used to manipulate, record, and/or otherwise control use of said content; and permissions, limitations, administrative control information and/or requirements applicable to said content, and/or said computer software and/or methods. Logical objects may or may not be

stored, and may or may not be present in, or accessible to, any given electronic appliance 600. The content portion of a logical object may be organized as information contained in, not contained in, or partially contained in one or more objects.

5

Briefly, the Figure 17 "logical object" structure 800 in the preferred embodiment includes a public header 802, private header 804, a "private body" 806 containing one or more methods 1000, permissions record(s) (PERC) 808 (which may include one or more key blocks 810), and one or more data blocks or areas 812. These elements may be "packaged" within a "container" 302. This generalized, logical object structure 800 is used in the preferred embodiment for different types of VDE objects 300 categorized by the type and location of their content.

10

15

The "container" concept is a convenient metaphor used to give a name to the collection of elements required to make use of content or to perform an administrative-type activity. Container 302 typically includes identifying information, control structures and content (e.g., a property or administrative data). The term "container" is often (e.g., Bento/OpenDoc and OLE) used to describe a collection of information stored on a computer system's

20

secondary storage system(s) or accessible to a computer system over a communications network on a "server's" secondary storage system. The "container" 302 provided by the preferred embodiment is not so limited or restricted. In VDE 100, there is
5 no requirement that this information is stored together, received at the same time, updated at the same time, used for only a single object, or be owned by the same entity. Rather, in VDE 100 the container concept is extended and generalized to include real-time content and/or online interactive content passed to an
10 electronic appliance over a cable, by broadcast, or communicated by other electronic communication means.

Thus, the "complete" VDE container 302 or logical object structure 800 may not exist at the user's location (or any other
15 location, for that matter) at any one time. The "logical object" may exist over a particular period of time (or periods of time), rather than all at once. This concept includes the notion of a "virtual container" where important container elements may exist either as a plurality of locations and/or over a sequence of
20 time periods (which may or may not overlap). Of course, VDE 100 containers can also be stored with all required control structures and content together. This represents a continuum:

from all content and control structures present in a single container, to no locally accessible content or container specific control structures.

5 Although at least some of the data representing the object is typically encrypted and thus its structure is not discernible, within a PPE 650 the object may be viewed logically as a "container" 302 because its structure and components are automatically and transparently decrypted.

10

A container model merges well with the event-driven processes and ROS 602 provided by the preferred embodiment. Under this model, content is easily subdivided into small, easily manageable pieces, but is stored so that it maintains the structural richness inherent in unencrypted content. An object oriented container model (such as Bento/OpenDoc or OLE) also provides many of the necessary "hooks" for inserting the necessary operating system integration components, and for defining the various content specific methods.

15
20

In more detail, the logical object structure 800 provided by the preferred embodiment includes a public (or unencrypted)

header 802 that identifies the object and may also identify one or more owners of rights in the object and/or one or more distributors of the object. Private (or encrypted) header 804 may include a part or all of the information in the public header and
5 further, in the preferred embodiment, will include additional data for validating and identifying the object 300 when a user attempts to register as a user of the object with a service clearinghouse, VDE administrator, or an SPU 500.
Alternatively, information identifying one or more rights owners
10 and/or distributors of the object may be located in encrypted form within encrypted header 804, along with any of said additional validating and identifying data.

Each logical object structure 800 may also include a
15 "private body" 806 containing or referencing a set of methods 1000 (i.e., programs or procedures) that control use and distribution of the object 300. The ability to optionally incorporate different methods 1000 with each object is important to making VDE 100 highly configurable. Methods 1000 perform
20 the basic function of defining what users (including, where appropriate, distributors, client administrators, etc.), can and cannot do with an object 300. Thus, one object 300 may come

with relatively simple methods, such as allowing unlimited viewing within a fixed period of time for a fixed fee (such as the newsstand price of a newspaper for viewing the newspaper for a period of one week after the paper's publication), while other
5 objects may be controlled by much more complicated (e.g., billing and usage limitation) methods.

Logical object structure 800 shown in Figure 17 may also include one or more PERCs 808. PERCs 808 govern the use of an
10 object 300, specifying methods or combinations of methods that must be used to access or otherwise use the object or its contents. The permission records 808 for an object may include key block(s) 810, which may store decryption keys for accessing the content of the encrypted content stored within the object 300.

15 The content portion of the object is typically divided into portions called data blocks 812. Data blocks 812 may contain any sort of electronic information, such as, "content," including computer programs, images, sound, VDE administrative
20 information, etc. The size and number of data blocks 812 may be selected by the creator of the property. Data blocks 812 need not all be the same size (size may be influenced by content usage,

database format, operating system, security and/or other considerations). Security will be enhanced by using at least one key block 810 for each data block 812 in the object, although this is not required. Key blocks 810 may also span portions of a plurality of data blocks 812 in a consistent or pseudo-random manner. The spanning may provide additional security by applying one or more keys to fragmented or seemingly random pieces of content contained in an object 300, database, or other information entity.

10

Many objects 300 that are distributed by physical media and/or by "out of channel" means (e.g., redistributed after receipt by a customer to another customer) might not include key blocks 810 in the same object 300 that is used to transport the content protected by the key blocks. This is because VDE objects may contain data that can be electronically copied outside the confines of a VDE node. If the content is encrypted, the copies will also be encrypted and the copier cannot gain access to the content unless she has the appropriate decryption key(s). For objects in which maintaining security is particularly important, the permission records 808 and key blocks 810 will frequently be distributed electronically, using secure communications techniques

15

20

(discussed below) that are controlled by the VDE nodes of the sender and receiver. As a result, permission records 808 and key blocks 810 will frequently, in the preferred embodiment, be stored only on electronic appliances 600 of registered users (and
5 may themselves be delivered to the user as part of a registration/initialization process). In this instance, permission records 808 and key blocks 810 for each property can be encrypted with a private DES key that is stored only in the secure memory of an SPU 500, making the key blocks unusable
10 on any other user's VDE node. Alternately, the key blocks 810 can be encrypted with the end user's public key, making those key blocks usable only to the SPU 500 that stores the corresponding private key (or other, acceptably secure, encryption/security techniques can be employed).

15

In the preferred embodiment, the one or more keys used to encrypt each permission record 808 or other management information record will be changed every time the record is updated (or after a certain one or more events). In this event, the
20 updated record is re-encrypted with new one or more keys. Alternately, one or more of the keys used to encrypt and decrypt management information may be "time aged" keys that

automatically become invalid after a period of time.

Combinations of time aged and other event triggered keys may also be desirable; for example keys may change after a certain number of accesses, and/or after a certain duration of time or
5 absolute point in time. The techniques may also be used together for any given key or combination of keys. The preferred embodiment procedure for constructing time aged keys is a one-way convolution algorithm with input parameters including user and site information as well as a specified portion of the real
10 time value provided by the SPU RTC 528. Other techniques for time aging may also be used, including for example techniques that use only user or site information, absolute points in time, and/or duration of time related to a subset of activities related to using or decrypting VDE secured content or the use of the VDE
15 system.

VDE 100 supports many different types of "objects" 300 having the logical object structure 800 shown in Figure 17. Objects may be classified in one sense based on whether the
20 protection information is bound together with the protected information. For example, a container that is bound by its control(s) to a specific VDE node is called a "stationary object"

(see Figure 18). A container that is not bound by its control information to a specific VDE node but rather carries sufficient control and permissions to permit its use, in whole or in part, at any of several sites is called a "Traveling Object" (see Figure 19).

5

Objects may be classified in another sense based on the nature of the information they contain. A container with information content is called a "Content Object" (see Figure 20). A container that contains transaction information, audit trails, VDE structures, and/or other VDE control/administrative information is called an "Administrative Object" (see Figure 21). Some containers that contain executable code operating under VDE control (as opposed to being VDE control information) are called "Smart Objects." Smart Objects support user agents and provide control for their execution at remote sites. There are other categories of objects based upon the location, type and access mechanism associated with their content, that can include combinations of the types mentioned above. Some of these objects supported by VDE 100 are described below. Some or all of the data blocks 812 shown in Figure 17 may include "embedded" content, administrative, stationary, traveling and/or other objects.

10

15

20

1. Stationary Objects

Figure 18 shows an example of a "Stationary Object" structure 850 provided by the preferred embodiment.

5 "Stationary Object" structure 850 is intended to be used only at specific VDE electronic appliance/installations that have received explicit permissions to use one or more portions of the stationary object. Therefore, stationary object structure 850 does not contain a permissions record (PERC) 808; rather, this permissions record is supplied and/or delivered separately (e.g.,
10 at a different time, over a different path, and/or by a different party) to the appliance/installation 600. A common PERC 808 may be used with many different stationary objects.

As shown in Figure 18, public header 802 is preferably
15 "plaintext" (i.e., unencrypted). Private header 804 is preferably encrypted using at least one of many "private header keys." Private header 804 preferably also includes a copy of identification elements from public header 802, so that if the identification information in the plaintext public header is
20 tampered with, the system can determine precisely what the tamperer attempted to alter. Methods 1000 may be contained in a section called the "private body" 806 in the form of object local

methods, load modules, and/or user data elements. This private
body (method) section 806 is preferably encrypted using one or
more private body keys contained in the separate permissions
record 808. The data blocks 812 contain content (information or
5 administrative) that may be encrypted using one or more content
keys also provided in permissions record 808.

2. Traveling Objects

Figure 19 shows an example of a "traveling object"
10 structure 860 provided by the preferred embodiment. Traveling
objects are objects that carry with them sufficient information to
enable at least some use of at least a portion of their content
when they arrive at a VDE node.

15 Traveling object structure 860 may be the same as
stationary object structure 850 shown in Figure 18 except that
the traveling object structure includes a permissions record
(PERC) 808 within private header 804. The inclusion of PERC
808 within traveling object structure 860 permits the traveling
20 object to be used at any VDE electronic appliance/participant 600
(in accordance with the methods 1000 and the contained PERC
808).

"Traveling" objects are a class of VDE objects 300 that can specifically support "out of channel" distribution. Therefore, they include key block(s) 810 and are transportable from one electronic appliance 600 to another. Traveling objects may come with a quite limited usage related budget so that a user may use, in whole or part, content (such as a computer program, game, or database) and evaluate whether to acquire a license or further license or purchase object content. Alternatively, traveling object PERCs 808 may contain or reference budget records with, for example:

10

(a) budget(s) reflecting previously purchased rights or credit for future licensing or purchasing and enabling at least one or more types of object content usage, and/or

15

(b) budget(s) that employ (and may debit) available credit(s) stored on and managed by the local VDE node in order to enable object content use, and/or

20

(c) budget(s) reflecting one or more maximum usage criteria before a report to a local VDE node (and, optionally, also a report to a clearinghouse) is

required and which may be followed by a reset allowing further usage, and/or modification of one or more of the original one or more budget(s).

5 As with standard VDE objects 300, a user may be required to contact a clearinghouse service to acquire additional budgets if the user wishes to continue to use the traveling object after the exhaustion of an available budget(s) or if the traveling object (or a copy thereof) is moved to a different electronic appliance and
10 the new appliance does not have a available credit budget(s) that corresponds to the requirements stipulated by permissions record 808.

 For example, a traveling object PERC 808 may include a
15 reference to a required budget VDE 1200 or budget options that may be found and/or are expected to be available. For example, the budget VDE may reference a consumer's VISA, MC, AMEX, or other "generic" budget that may be object independent and may be applied towards the use of a certain or classes of traveling
20 object content (for example any movie object from a class of traveling objects that might be Blockbuster Video rentals). The budget VDE itself may stipulate one or more classes of objects it

may be used with, while an object may specifically reference a certain one or more generic budgets. Under such circumstances, VDE providers will typically make information available in such a manner as to allow correct referencing and to enable billing
5 handling and resulting payments.

Traveling objects can be used at a receiving VDE node electronic appliance 600 so long as either the appliance carries the correct budget or budget type (e.g. sufficient credit available
10 from a clearinghouse such as a VISA budget) either in general or for specific one or more users or user classes, or so long as the traveling object itself carries with it sufficient budget allowance or an appropriate authorization (e.g., a stipulation that the traveling object may be used on certain one or more installations
15 or installation classes or users or user classes where classes correspond to a specific subset of installations or users who are represented by a predefined class identifiers stored in a secure database 610). After receiving a traveling object, if the user (and/or installation) doesn't have the appropriate budget(s)
20 and/or authorizations, then the user could be informed by the electronic appliance 600 (using information stored in the traveling object) as to which one or more parties the user could

contact. The party or parties might constitute a list of alternative clearinghouse providers for the traveling object from which the user selects his desired contact).

5 As mentioned above, traveling objects enable objects 300 to be distributed "Out-Of-Channel;" that is, the object may be distributed by an unauthorized or not explicitly authorized individual to another individual. "Out of channel" includes paths of distribution that allow, for example, a user to directly
10 redistribute an object to another individual. For example, an object provider might allow users to redistribute copies of an object to their friends and associates (for example by physical delivery of storage media or by delivery over a computer network) such that if a friend or associate satisfies any certain criteria
15 required for use of said object, he may do so.

 For example, if a software program was distributed as a traveling object, a user of the program who wished to supply it or a usable copy of it to a friend would normally be free to do so.

20 Traveling Objects have great potential commercial significance, since useful content could be primarily distributed by users and through bulletin boards, which would require little or no

distribution overhead apart from registration with the "original" content provider and/or clearinghouse.

5 The "out of channel" distribution may also allow the provider to receive payment for usage and/or otherwise maintain at least a degree of control over the redistributed object. Such certain criteria might involve, for example, the registered presence at a user's VDE node of an authorized third party financial relationship, such as a credit card, along with sufficient
10 available credit for said usage.

Thus, if the user had a VDE node, the user might be able to use the traveling object if he had an appropriate, available budget available on his VDE node (and if necessary, allocated to
15 him), and/or if he or his VDE node belonged to a specially authorized group of users or installations and/or if the traveling object carried its own budget(s).

Since the content of the traveling object is encrypted, it can
20 be used only under authorized circumstances unless the traveling object private header key used with the object is broken—a potentially easier task with a traveling object as compared to, for

example, permissions and/or budget information since many objects may share the same key, giving a cryptanalyst both more information in cyphertext to analyze and a greater incentive to perform cryptoanalysis.

5

In the case of a "traveling object," content owners may distribute information with some or all of the key blocks 810 included in the object 300 in which the content is encapsulated. Putting keys in distributed objects 300 increases the exposure to attempts to defeat security mechanisms by breaking or
10 cryptoanalyzing the encryption algorithm with which the private header is protected (e.g., by determining the key for the header's encryption). This breaking of security would normally require considerable skill and time, but if broken, the algorithm and key
15 could be published so as to allow large numbers of individuals who possess objects that are protected with the same key(s) and algorithm(s) to illegally use protected information. As a result, placing keys in distributed objects 300 may be limited to content that is either "time sensitive" (has reduced value after the
20 passage of a certain period of time), or which is somewhat limited in value, or where the commercial value of placing keys in objects (for example convenience to end-users, lower cost of eliminating

the telecommunication or other means for delivering keys and/or permissions information and/or the ability to supporting objects going "out-of-channel") exceeds the cost of vulnerability to sophisticated hackers. As mentioned elsewhere, the security of keys may be improved by employing convolution techniques to avoid storing "true" keys in a traveling object, although in most cases using a shared secret provided to most or all VDE nodes by a VDE administrator as an input rather than site ID and/or time in order to allow objects to remain independent of these values.

10

As shown in Figure 19 and discussed above, a traveling object contains a permissions record 808 that preferably provides at least some budget (one, the other, or both, in a general case). Permission records 808 can, as discussed above, contain a key block(s) 810 storing important key information. PERC 808 may also contain or refer to budgets containing potentially valuable quantities/values. Such budgets may be stored within a traveling object itself, or they may be delivered separately and protected by highly secure communications keys and administrative object keys and management database techniques.

15

20

The methods 1000 contained by a traveling object will typically include an installation procedure for "self registering" the object using the permission records 808 in the object (e.g., a REGISTER method). This may be especially useful for objects
5 that have time limited value, objects (or properties) for which the end user is either not charged or is charged only a nominal fee (e.g., objects for which advertisers and/or information publishers are charged based on the number of end users who actually access published information), and objects that require widely
10 available budgets and may particularly benefit from out-of-channel distribution (e.g., credit card derived budgets for objects containing properties such as movies, software programs, games, etc.). Such traveling objects may be supplied with or without contained budget UDEs.

15

One use of traveling objects is the publishing of software, where the contained permission record(s) may allow potential customers to use the software in a demonstration mode, and possibly to use the full program features for a limited time before
20 having to pay a license fee, or before having to pay more than an initial trial fee. For example, using a time based billing method and budget records with a small pre-installed time budget to

allow full use of the program for a short period of time. Various control methods may be used to avoid misuse of object contents. For example, by setting the minimum registration interval for the traveling object to an appropriately large period of time (e.g.,
5 a month, or six months or a year), users are prevented from re-using the budget records in the same traveling object.

Another method for controlling the use of traveling objects is to include time-aged keys in the permission records that are
10 incorporated in the traveling object. This is useful generally for traveling objects to ensure that they will not be used beyond a certain date without re-registration, and is particularly useful for traveling objects that are electronically distributed by broadcast,
network, or telecommunications (including both one and two way
15 cable), since the date and time of delivery of such traveling objects aging keys can be set to accurately correspond to the time the user came into possession of the object.

Traveling objects can also be used to facilitate "moving" an
20 object from one electronic appliance 600 to another. A user could move a traveling object, with its incorporated one or more permission records 808 from a desktop computer, for example, to

his notebook computer. A traveling object might register its user within itself and thereafter only be useable by that one user. A traveling object might maintain separate budget information, one for the basic distribution budget record, and another for the "active" distribution budget record of the registered user. In this way, the object could be copied and passed to another potential user, and then could be a portable object for that user.

Traveling objects can come in a container which contains other objects. For example, a traveling object container can include one or more content objects and one or more administrative objects for registering the content object(s) in an end user's object registry and/or for providing mechanisms for enforcing permissions and/or other security functions. Contained administrative object(s) may be used to install necessary permission records and/or budget information in the end user's electronic appliance.

Content Objects

Figure 20 shows an example of a VDE content object structure 880. Generally, content objects 880 include or provide information content. This "content" may be any sort of electronic

information. For example, content may include: computer software, movies, books, music, information databases, multimedia information, virtual reality information, machine instructions, computer data files, communications messages and/or signals, and other information, at least a portion of which is used and/or manipulated by one or more electronic appliances. VDE 100 can also be configured for authenticating, controlling, and/or auditing electronic commercial transactions and communications such as inter-bank transactions, electronic purchasing communications, and the transmission of, auditing of, and secure commercial archiving of, electronically signed contracts and other legal documents; the information used for these transactions may also be termed "content." As mentioned above, the content need not be physically stored within the object container but may instead be provided separately at a different time (e.g., a real time feed over a cable).

Content object structure 880 in the particular example shown in Figure 20 is a type of stationary object because it does not include a PERC 808. In this example, content object structure 880 includes, as at least part of its content 812, at least one embedded content object 882 as shown in Figure 5A.

Content object structure 880 may also include an administrative object 870. Thus, objects provided by the preferred embodiment may include one or more "embedded" objects.

5

Administrative Objects

Figure 21 shows an example of an administrative object structure 870 provided by the preferred embodiment. An "administrative object" generally contains permissions, administrative control information, computer software and/or methods associated with the operation of VDE 100.

Administrative objects may also or alternatively contain records of use, and/or other information used in, or related to, the operation of VDE 100. An administrative object may be distinguished from a content object by the absence of VDE protected "content" for release to an end user for example. Since objects may contain other objects, it is possible for a single object to contain one or more content containing objects and one or more administrative objects. Administrative objects may be used to transmit information between electronic appliances for update, usage reporting, billing and/or control purposes. They contain information that helps to administer VDE 100 and keep it operating properly. Administrative objects generally are sent

between two VDE nodes, for example, a VDE clearinghouse service, distributor, or client administrator and an end user's electronic appliance 600.

5 Administrative object structure 870 in this example includes a public header 802, private header 804 (including a "PERC" 808) and a "private body" 806 containing methods 1000. Administrative object structure 870 in this particular example shown in Figure 20 is a type of traveling object because it
10 contains a PERC 808, but the administrative object could exclude the PERC 808 and be a stationary object. Rather than storing information content, administrative object structure 870 stores "administrative information content" 872. Administrative information content 872 may, for example, comprise a number of
15 records 872a, 872b, . . . 872n each corresponding to a different "event." Each record 872a, 872b, . . . 872n may include an "event" field 874, and may optionally include a parameter field 876 and/or a data field 878. These administrative content records 872 may be used by VDE 100 to define events that may be
20 processed during the course of transactions, e.g., an event designed to add a record to a secure database might include parameters 896 indicating how and where the record should be

stored and data field 878 containing the record to be added. In another example, a collection of events may describe a financial transaction between the creator(s) of an administrative object and the recipient(s), such as a purchase, a purchase order, or an invoice. Each event record 872 may be a set of instructions to be executed by the end user's electronic appliance 600 to make an addition or modification to the end user's secure database 610, for example. Events can perform many basic management functions, for example: add an object to the object registry, including providing the associated user/group record(s), rights records, permission record and/or method records; delete audit records (by "rolling up" the audit trail information into, for example, a more condensed, e.g. summary form, or by actual deletion); add or update permissions records 808 for previously registered objects; add or update budget records; add or update user rights records; and add or update load modules.

In the preferred embodiment, an administrative object may be sent, for example, by a distributor, client administrator, or, perhaps, a clearinghouse or other financial service provider, to an end user, or, alternatively, for example, by an object creator to a distributor or service clearinghouse. Administrative objects, for

example, may increase or otherwise adjust budgets and/or permissions of the receiving VDE node to which the administrative object is being sent. Similarly, administrative objects containing audit information in the data area 878 of an event record 872 can be sent from end users to distributors, and/or clearinghouses and/or client administrators, who might themselves further transmit to object creators or to other participants in the object's chain of handling.

10

Methods

Methods 1000 in the preferred embodiment support many of the operations that a user encounters in using objects and communicating with a distributor. They may also specify what method fields are displayable to a user (e.g., use events, user request events, user response events, and user display events). Additionally, if distribution capabilities are supported in the method, then the method may support distribution activities, distributor communications with a user about a method, method modification, what method fields are displayable to a distributor, and any distribution database checks and record keeping (e.g., distribution events, distributor request events, and distributor response events).

15

20

Given the generality of the existing method structure, and the diverse array of possibilities for assembling methods, a generalized structure may be used for establishing relationships between methods. Since methods 1000 may be independent of an object that requires them during any given session, it is not possible to define the relationships within the methods themselves. "Control methods" are used in the preferred embodiment to define relationships between methods. Control methods may be object specific, and may accommodate an individual object's requirements during each session.

A control method of an object establishes relationships between other methods. These relationships are parameterized with explicit method identifiers when a record set reflecting desired method options for each required method is constructed during a registration process.

An "aggregate method" in the preferred embodiment represents a collection of methods that may be treated as a single unit. A collection of methods that are related to a specific property, for example, may be stored in an aggregate method. This type of aggregation is useful from an implementation point

of view because it may reduce bookkeeping overhead and may improve overall database efficiency. In other cases, methods may be aggregated because they are logically coupled. For example, two budgets may be linked together because one of the budgets represents an overall limitation, and a second budget represents the current limitation available for use. This would arise if, for example, a large budget is released in small amounts over time.

For example, an aggregate method that includes meter, billing and budget processes can be used instead of three separate methods. Such an aggregate method may reference a single "load module" 1100 that performs all of the functions of the three separate load modules and use only one user data element that contains meter, billing and budget data. Using an aggregate method instead of three separate methods may minimize overall memory requirements, database searches, decryptions, and the number of user data element writes back to a secure database 610. The disadvantage of using an aggregate method instead of three separate methods can be a loss of some flexibility on the part of a provider and user in that various functions may no longer be independently replaceable.

Figure 16 shows methods 1000 as being part of secure database 610.

5 A "method" 1000 provided by the preferred embodiment is a collection of basic instructions and information related to the basic instructions, that provides context, data, requirements and/or relationships for use in performing, and/or preparing to perform, the basic instructions in relation to the operation of one or more electronic appliances 600. As shown in Figure 16,
10 methods 1000 in the preferred embodiment are represented in secure database 610 by:

- C method "cores" 1000N;
- C Method Data Elements (MDEs) 1202;
- C User Data Elements (UDEs) 1200; and
- 15 C Data Description Elements (DTDs).

Method "core" 1000N in the preferred embodiment may contain or reference one or more data elements such as MDEs 1202 and UDEs 1200. In the preferred embodiment, MDEs 1202
20 and UDEs 1200 may have the same general characteristics, the main difference between these two types of data elements being that a UDE is preferably tied to a particular method as well as a

particular user or group of users, whereas an MDE may be tied to a particular method but may be user independent. These MDE and UDE data structures 1200, 1202 are used in the preferred embodiment to provide input data to methods 1000, to receive data outputted by methods, or both. MDEs 1202 and UDEs 1200 may be delivered independently of method cores 1000N that reference them, or the data structures may be delivered as part of the method cores. For example, the method core 1000N in the preferred embodiment may contain one or more MDEs 1202 and/or UDEs 1200 (or portions thereof). Method core 1000N may, alternately or in addition, reference one or more MDE and/or UDE data structures that are delivered independently of method core(s) that reference them.

Method cores 1000N in the preferred embodiment also reference one or more "load modules" 1100. Load modules 1100 in the preferred embodiment comprise executable code, and may also include or reference one or more data structures called "data descriptor" ("DTD") information. This "data descriptor" information may, for example, provide data input information to the DTD interpreter 590. DTDs may enable load modules 1100

to access (e.g., read from and/or write to) the MDE and/or UDE data elements 1202, 1200.

5 Method cores 1000' may also reference one or more DTD and/or MDE data structures that contain a textual description of their operations suitable for inclusion as part of an electronic contract. The references to the DTD and MDE data structures may occur in the private header of the method core 1000', or may be specified as part of the event table described below.

10

Figure 22 shows an example of a format for a method core 1000N provided by the preferred embodiment. A method core 1000N in the preferred embodiment contains a method event table 1006 and a method local data area 1008. Method event table 15 1006 lists "events." These "events" each reference "load modules" 1100 and/or PERCs 808 that control processing of an event. Associated with each event in the list is any static data necessary to parameterize the load module 1000 or permissions record 808, and reference(s) into method user data area 1008 that are needed 20 to support that event. The data that parameterizes the load module 1100 can be thought of, in part, as a specific function call to the load module, and the data elements corresponding to it

may be thought of as the input and/or output data for that specific function call.

Method cores 1000N can be specific to a single user, or they
5 may be shared across a number of users (e.g., depending upon the uniqueness of the method core and/or the specific user data element). Specifically, each user/group may have its own UDE 1200 and use a shared method core 1000N. This structure allows for lower database overhead than when associating an entire
10 method core 1000N with a user/group. To enable a user to use a method, the user may be sent a method core 1000N specifying a UDE 1200. If that method core 1000N already exists in the site's secure database 610, only the UDE 1200 may need to be added. Alternately, the method may create any required UDE 1200 at
15 registration time.

The Figure 22 example of a format for a method core 1000N provided by the preferred embodiment includes a public (unencrypted) header 802, a private (encrypted) header 804,
20 method event table 1006, and a method local data area 1008.

An example of a possible field layout for method core 1000N public header 802 is shown in the following table:

Field Type		Description	
5	Method ID	Creator ID	Site ID of creator of this method.
		Distributor ID	Distributor of this method (e.g., last change).
		Type ID	Constant, indicates method "type."
		Method ID	Unique sequence number for this method.
		Version ID	Version number of this method.
	Other classification information	Class ID	ID to support different method "classes."
		Type ID	ID to support method type compatible searching.
10	Descriptive Information	Description(s)	Textual description(s) of the method.
		Event Summary	Summary of event classes (e.g., USE) that this method supports.

An example of a possible field layout for private header 804 is shown below:

FIG. 20

Field Type		Description
Copy of Public Header 802 Method ID and "Other Classification Information"		Method ID from Public Header
5 Descriptive Information	# of Events	# of events supported in this method.
Access and Reference Tags	Access tag	Tags used to determine if this method is the correct method under management by the SPU; ensure that the method core 1000N is used only under appropriate circumstances.
	Validation tag	
	Correlation tag	
Data Structure Reference		Optional Reference to DTD(s) and/or MDE(s)
10 Check Value		Check value for Private Header and method event table.
Check Value for Public Header		Check Value for Public Header

Referring once again to Figure 22, method event table 1006
 15 may in the preferred embodiment include from 1 to N method
 event records 1012. Each of these method event records 1012
 corresponds to a different event the method 1000 represented by
 method core 1000N may respond to. Methods 1000 in the
 preferred embodiment may have completely different behavior
 20 depending upon the event they respond to. For example, an

AUDIT method may store information in an audit trail UDE 1200 in response to an event corresponding to a user's use of an object or other resource. This same AUDIT method may report the stored audit trail to a VDE administrator or other participant
5 in response to an administrative event such as, for example, a timer expiring within a VDE node or a request from another VDE participant to report the audit trail. In the preferred embodiment, each of these different events may be represented by an "event code." This "event code" may be passed as a
10 parameter to a method when the method is called, and used to "look up" the appropriate method event record 1012 within method event table 1006. The selected method event record 1012, in turn, specifies the appropriate information (e.g., load module(s) 1100, data element UDE(s) and MDE(s) 1200, 1202,
15 and/or PERC(s) 808) used to construct a component assembly 690 for execution in response to the event that has occurred.

Thus, in the preferred embodiment, each method event record 1012 may include an event field 1014, a LM/PERC
20 reference field 1016, and any number of data reference fields 1018. Event fields 1014 in the preferred embodiment may contain a "event code" or other information identifying the

corresponding event. The LM/PERC reference field 1016 may provide a reference into the secure database 610 (or other "pointer" information) identifying a load module 1100 and/or a PERC 808 providing (or referencing) executable code to be loaded and executed to perform the method in response to the event.

5 Data reference fields 1018 may include information referencing a UDE 1200 or a MDE 1202. These data structures may be contained in the method local data area 1008 of the method core 1000N, or they may be stored within the secure database 610 as

10 independent deliverables.

The following table is an example of a possible more detailed field layout for a method event record 1012:

15	Field Type		Description
	Event Field 1014		Identifies corresponding event.
	Access tag		Secret tag to grant access to this row of the method event record.
20	LM/PERC Reference Field 1016	DB ID or offset/size	Database reference (or local pointer).
		Correlation tag	Correlation tag to assert when referencing this element.

15

Field Type		Description
# of Data Element Reference Fields		Count of data reference fields in the method event record.
Data Reference Field 1	UDE ID or offset/size	Database 610 reference (or local pointer).
	Correlation tag	Correlation tag to assert when referencing this element.
! . . .		
Data Reference Field n	UDE ID or offset/size	Database 610 reference (or local pointer).
	Correlation tag	Correlation tag to assert when referencing this element.

5

10

Load Modules

Figure 23 is an example of a load module 1100 provided by the preferred embodiment. In general, load modules 1100 represent a collection of basic functions that are used for control operations.

15

Load module 1100 contains code and static data (that is functionally the equivalent of code), and is used to perform the basic operations of VDE 100. Load modules 1100 will generally be shared by all the control structures for all objects in the

20

system, though proprietary load modules are also permitted. Load modules 1100 may be passed between VDE participants in administrative object structures 870, and are usually stored in secure database 610. They are always encrypted and

5 authenticated in both of these cases. When a method core 1000N references a load module 1100, a load module is loaded into the SPE 503, decrypted, and then either passed to the electronic appliance microprocessor for executing in an HPE 655 (if that is where it executes), or kept in the SPE (if that is where it

10 executes). If no SPE 503 is present, the load module may be decrypted by the HPE 655 prior to its execution.

Load module creation by parties is preferably controlled by a certification process or a ring based SPU architecture. Thus,

15 the process of creating new load modules 1100 is itself a controlled process, as is the process of replacing, updating or deleting load modules already stored in a secured database 610.

A load module 1100 is able to perform its function only

20 when executed in the protected environment of an SPE 503 or an HPE 655 because only then can it gain access to the protected elements (e.g., UDEs 1200, other load modules 1100) on which it

operates. Initiation of load module execution in this environment is strictly controlled by a combination of access tags, validation tags, encryption keys, digital signatures and/or correlation tags. Thus, a load module 1100 may only be referenced if the caller
5 knows its ID and asserts the shared secret correlation tag specific to that load module. The decrypting SPU may match the identification token and local access tag of a load module after decryption. These techniques make the physical replacement of any load module 1100 detectable at the next physical access of
10 the load module. Furthermore, load modules 1100 may be made "read only" in the preferred embodiment. The read-only nature of load modules 1100 prevents the write-back of load modules that have been tampered with in non-secure space.

15 Load modules are not necessarily directly governed by PERCs 808 that control them, nor must they contain any time/date information or expiration dates. The only control consideration in the preferred embodiment is that one or more
20 methods 1000 reference them using a correlation tag (the value of a protected object created by the load module's owner, distributed to authorized parties for inclusion in their methods, and to which access and use is controlled by one or more PERCs 808). If a

method core 1000N references a load module 1100 and asserts the proper correlation tag (and the load module satisfies the internal tamper checks for the SPE 503), then that load module can be loaded and executed, or it can be acquired from, shipped to, updated, or deleted by, other systems.

As shown in Figure 23, load modules 1100 in the preferred embodiment may be constructed of a public (unencrypted) header 802, a private (encrypted) header 804, a private body 1106 containing the encrypted executable code, and one or more data description elements ("DTDs") 1108. The DTDs 1108 may be stored within a load module 1100, or they may be references to static data elements stored in secure database 610.

The following is an example of a possible field layout for load module public header 802:

Field Type	Description
LM ID	VDE ID of Load Module.
Creator ID	Site ID of creator of this load module.
Type ID	Constant indicates load module type.

Field Type		Description
	LM ID	Unique sequence number for this load module, which uniquely identifies the load module in a sequence of load modules created by an authorized VDE participant.
	Version ID	Version number of this load module.
Other classification information	Class ID	ID to support different load module classes.
	Type ID	ID to support method type compatible searching.
Descriptive Information	Description	Textual description of the load module.
	Execution space code	Value that describes what execution space (e.g., SPE or HPE) this load module.

5

10

15

Many load modules 1100 contain code that executes in an SPE 503. Some load modules 1100 contain code that executes in an HPE 655. This allows methods 1000 to execute in whichever environment is appropriate. For example, an INFORMATION method 1000 can be built to execute only in SPE 503 secure space for government classes of security, or in an HPE 655 for commercial applications. As described above, the load module public header 802 may contain an "execution space code" field

that indicates where the load module 1100 needs to execute. This functionality also allows for different SPE instruction sets as well as different user platforms, and allows methods to be constructed without dependencies on the underlying load module instruction set.

5

Load modules 1100 operate on three major data areas: the stack, load module parameters, and data structures. The stack and execution memory size required to execute the load module 1100 are preferably described in private header 804, as are the data descriptions from the stack image on load module call, return, and any return data areas. The stack and dynamic areas are described using the same DTD mechanism. The following is an example of a possible layout for a load module private header 1104:

10

15

5

10

Field Type		Description
Copy of some or all of information from public header 802		Object ID from Public Header.
Other classification information	Check Value	Check Value for Public Header.
Descriptive Information	LM Size	Size of executable code block.
	LM Exec Size	Executable code size for the load modul
	LM Exec Stack	Stack size required for the load module
	Execution space code	Code that describes the execution spac load module.
Access and reference tags	Access tag	Tags used to determine if the load mod correct LM requested by the SPE.
	Validation tag	
	Correlation tag	Tag used to determine if the caller of th the right to execute this LM.
	Digital Signature	Used to determine if the LM executable is intact and was created by a trusted s with a correct certificate for creating L
Data record descriptor information	DTD count	Number of DTDs that follow the code bl
	DTD 1 reference	If locally defined, the physical size and bytes of the first DTD defined for this L If publicly referenced DTD, this is the and the correlation tag to permit access record.

	DTD N reference	<p>If locally defined, the physical size and bytes of the Nth DTD defined for this L</p> <p>If publicly referenced DTD, this is the and the correlation tag to permit access record.</p>
Check Value		Check Value for entire LM.

Each load module 1100 also may use DTD 1108

5 information to provide the information necessary to support building methods from a load module. This DTD information contains the definition expressed in a language such as SGML for the names and data types of all of the method data fields that the load module supports, and the acceptable ranges of values that

10 can be placed in the fields. Other DTDs may describe the function of the load module 1100 in English for inclusion in an electronic contract, for example.

The next section of load module 1100 is an encrypted

15 executable body 1106 that contains one or more blocks of encrypted code. Load modules 1100 are preferably coded in the "native" instruction set of their execution environment for efficiency and compactness. SPU 500 and platform providers may provide versions of the standard load modules 1100 in order

to make their products cooperate with the content in distribution mechanisms contemplated by VDE 100. The preferred embodiment creates and uses native mode load modules 1100 in lieu of an interpreted or "p-code" solution to optimize the performance of a limited resource SPU. However, when sufficient SPE (or HPE) resources exist and/or platforms have sufficient resources, these other implementation approaches may improve the cross platform utility of load module code.

The following is an example of a field layout for a load module DTD 1108:

5

Field Type		Description
DTD ID		Uses Object ID from Private Header.
	Creator ID	Site ID of creator of this DTD.
	Type ID	Constant.
	DTD ID	Unique sequence number for this DTD.
	Version ID	Version number of this DTD.
Descriptive Information	DTD Size	Size of DTD block.
Access and reference tags	Access tag	Tags used to determine if the DTD is the DTD requested by the SPE.
	Validation tag	
	Correlation tag	Tag used to determine if the caller of this the right to use the DTD.
DTD Body	DTD Data Definition 1	
	DTD Data Definition 2	
	!	
	DTD Data Definition N	
	Check Value	Check Value for entire DTD record.

10

Some examples of how load modules 1100 may use DTDs 1108 include:

- 5 C Increment data element (defined by name in DTD3) value in data area DTD4 by value in DTD1
- C Set data element (defined by name in DTD3) value in data area DTD4 to value in DTD3
- 10 C Compute atomic element from event in DTD1 from table in DTD3 and return in DTD2
- C Compute atomic element from event in DTD1 from equation in DTD3 and return in DTD2
- 15 C Create load module from load module creation template referenced in DTD3
- C Modify load module in DTD3 using content in DTD4
- C Destroy load module named in DTD3
- 20 Commonly used load modules 1100 may be built into a SPU 500 as space permits. VDE processes that use built-in load modules 1100 will have significantly better performance than processes that have to find, load and decrypt external load modules. The most useful load modules 1100 to build into a SPU
- 25 might include scaler meters, fixed price billing, budgets and load modules for aggregate methods that perform these three processes.

User Data Elements (UDEs) 1200 and Method Data Elements (MDEs) 1202

5 User Data Elements (UDEs) 1200 and Method Data
Elements (MDEs) 1202 in the preferred embodiment store data.
There are many types of UDEs 1200 and MDEs 1202 provided by
the preferred embodiment. In the preferred embodiment, each of
these different types of data structures shares a common overall
format including a common header definition and naming
10 scheme. Other UDEs 1200 that share this common structure
include "local name services records" (to be explained shortly)
and account information for connecting to other VDE
participants. These elements are not necessarily associated with
an individual user, and may therefore be considered MDEs 1202.
15 All UDEs 1200 and all MDEs 1202 provided by the preferred
embodiment may, if desired, (as shown in Figure 16) be stored in
a common physical table within secure database 610, and
database access processes may commonly be used to access all of
these different types of data structures.

20

In the preferred embodiment, PERCs 808 and user rights
table records are types of UDE 1200. There are many other types
of UDEs 1200/MDEs 1202, including for example, meters, meter

trails, budgets, budget trails, and audit trails. Different formats for these different types of UDEs/MDEs are defined, as described above, by SGML definitions contained within DTDs 1108.

5 Methods 1000 use these DTDs to appropriately access UDEs/MDEs 1200, 1202.

Secure database 610 stores two types of items: static and dynamic. Static data structures and other items are used for information that is essentially static information. This includes
10 load modules 1100, PERCs 808, and many components of methods. These items are not updated frequently and contain expiration dates that can be used to prevent "old" copies of the information from being substituted for newly received items. These items may be encrypted with a site specific secure
15 database file key when they are stored in the secure database 610, and then decrypted using that key when they are loaded into the SPE.

Dynamic items are used to support secure items that must
20 be updated frequently. The UDEs 1200 of many methods must be updated and written out of the SPE 503 after each use. Meters and budgets are common examples of this. Expiration

dates cannot be used effectively to prevent substitution of the previous copy of a budget UDE 1200. To secure these frequently updated items, a transaction tag is generated and included in the encrypted item each time that item is updated. A list of all VDE
5 item IDs and the current transaction tag for each item is maintained as part of the secure database 610.

Figure 24 shows an example of a user data element ("UDE") 1200 provided by the preferred embodiment. As shown
10 in Figure 24, UDE 1200 in the preferred embodiment includes a public header 802, a private header 804, and a data area 1206. The layout for each of these user data elements 1200 is generally defined by an SGML data definition contained within a DTD
1108 associated with one or more load modules 1100 that operate
15 on the UDE 1200.

UDEs 1200 are preferably encrypted using a site specific key once they are loaded into a site. This site-specific key masks a validation tag that may be derived from a cryptographically
20 strong pseudo-random sequence by the SPE 503 and updated each time the record is written back to the secure database 610. This technique provides reasonable assurance that the UDE 1200

has not been tampered with nor substituted when it is requested
by the system for the next use.

5 Meters and budgets are perhaps among the most common
data structures in VDE 100. They are used to count and record
events, and also to limit events. The data structures for each
meter and budget are determined by the content provider or a
distributor/redistributor authorized to change the information.
Meters and budgets, however, generally have common
10 information stored in a common header format (e.g., user ID, site
ID and related identification information).

The content provider or distributor/redistributor may
specify data structures for each meter and budget UDE.
15 Although these data structures vary depending upon the
particular application, some are more common than others. The
following table lists some of the more commonly occurring data
structures for METER and BUDGET methods:

20

Field type	Format	Typical Use	Description or Use
Ascending Use Counter	byte, short, long, or unsigned versions of the same widths	Meter/Budget	Ascending count of use
Descending Use Counter	byte, short, long, or unsigned versions of the same widths	Budget	Descending count of permitted use; eg., remaining budget.
Counter/Limit	2, 4 or 8 byte integer split into two related bytes or words	Meter/Budget	usage limits since a sp time; generally used in compound meter data structures.
Bitmap	Array bytes	Meter/Budget	Bit indicator of use or ownership.
Wide bitmap	Array of bytes	Meter/Budget	Indicator of use or ownership that may ag with time.
Last Use Date	time_t	Meter/Budget	Date of last use.
Start Date	time_t	Budget	Date of first allowable
Expiration Date	time_t	Meter/Budget	Expiration Date.
Last Audit Date	time_t	Meter/Budget	Date of last audit.
Next Audit Date	time_t	Meter/Budget	Date of next required a
Auditor	VDE ID	Meter/Budget	VDE ID of authorized auditor.

The information in the table above is not complete or comprehensive, but rather is intended to show some examples of types of information that may be stored in meter and budget

related data structures. The actual structure of particular
meters and budgets is determined by one or more DTDs 1108
associated with the load modules 1100 that create and
manipulate the data structure. A list of data types permitted by
5 the DTD interpreter 590 in VDE 100 is extensible by properly
authorized parties.

Figure 25 shows an example of one particularly advantageous kind of UDE 1200 data area 1206. This data area 1206 defines a "map" that may be used to record usage information. For example, a meter method 1000 may maintain one or more "usage map" data areas 1206. The usage map may be a "usage bit map" in the sense that it stores one or more bits of information (i.e., a single or multi-dimensional bit image) corresponding to each of several types or categories of usage. Usage maps are an efficient means for referencing prior usage. For example, a usage map data area may be used by a meter method 1000 to record all applicable portions of information content that the user has paid to use, thus supporting a very efficient and flexible means for allowing subsequent user usage of the same portions of the information content. This may enable certain VDE related security functions such as "contiguousness," "logical relatedness," randomization of usage, and other usage types. Usage maps may be analyzed for other usage patterns (e.g., quantity discounting, or for enabling a user to reaccess information content for which the user previously paid for unlimited usage).

The "usage map" concept provided by the preferred embodiment may be tied to the concept of "atomic elements." In the preferred embodiment, usage of an object 300 may be

metered in terms of "atomic elements." In the preferred embodiment, an "atomic element" in the metering context defines a unit of usage that is "sufficiently significant" to be recorded in a meter. The definition of what constitutes an "atomic element" is determined by the creator of an object 300. For instance, a "byte" of information content contained in an object 300 could be defined as an "atomic element," or a record of a database could be defined as an "atomic element," or each chapter of an electronically published book could be defined as an "atomic element."

An object 300 can have multiple sets of overlapping atomic elements. For example, an access to any database in a plurality of databases may be defined as an "atomic element." Simultaneously, an access to any record, field of records, sectors of informations, and/or bytes contained in any of the plurality of databases might also be defined as an "atomic element." In an electronically published newspaper, each hundred words of an article could be defined as an "atomic element," while articles of more than a certain length could be defined as another set of "atomic elements." Some portions of a newspaper (e.g., advertisements, the classified section, etc.) might not be mapped into an atomic element.

The preferred embodiment provides an essentially unbounded ability for the object creator to define atomic element

types. Such atomic element definitions may be very flexible to accommodate a wide variety of different content usage. Some examples of atomic element types supported by the preferred embodiment include bytes, records, files, sectors, objects, a quantity of bytes, contiguous or relatively contiguous bytes (or other predefined unit types), logically related bytes containing content that has some logical relationship by topic, location or other user specifiable logic of relationship, etc. Content creators preferably may flexibly define other types of atomic elements.

The preferred embodiment of the present invention provides EVENT methods to provide a mapping between usage events and atomic elements. Generally, there may be an EVENT method for each different set of atomic elements defined for an object 300. In many cases, an object 300 will have at least one type of atomic element for metering relating to billing, and at least one other atomic element type for non-billing related metering (e.g., used to, for example, detect fraud, bill advertisers, and/or collect data on end user usage activities).

In the preferred embodiment, each EVENT method in a usage related context performs two functions: (1) it maps an accessed event into a set of zero or more atomic elements, and (2) it provides information to one or more METER methods for metering object usage. The definition used to define this

mapping between access events and atomic elements may be in the form of a mathematical definition, a table, a load module, etc. When an EVENT method maps an access request into "zero" atomic elements, a user accessed event is not mapped into any atomic element based on the particular atomic element definition that applies. This can be, for example, the object owner is not interested in metering usage based on such accesses (e.g., because the object owner deems such accesses to be insignificant from a metering standpoint).

A "usage map" may employ a "bit map image" for storage of usage history information in a highly efficient manner.

Individual storage elements in a usage map may correspond to atomic elements. Different elements within a usage map may correspond to different atomic elements (e.g., one map element may correspond to number of bytes read, another map element may correspond to whether or not a particular chapter was opened, and yet another map element may correspond to some other usage event).

One of the characteristics of a usage map provided by the preferred embodiment of the present invention is that the significance of a map element is specified, at least in part, by the position of the element within the usage map. Thus, in a usage map provided by the preferred embodiment, the information

indicated or encoded by a map element is a function of its position (either physically or logically) within the map structure. As one simple example, a usage map for a twelve-chapter novel could consist of twelve elements, one for each chapter of the novel. When the user opens the first chapter, one or more bits within the element corresponding to the first chapter could be changed in value (e.g., set to "one"). In this simple example where the owner of the content object containing the novel was interested only in metering which chapters had been opened by the user, the usage map element corresponding to a chapter could be set to "one" the first time the user opened that corresponding chapter, and could remain "one" no matter how many additional times the user opened the chapter. The object owner or other interested VDE participant would be able to rapidly and efficiently tell which chapter(s) had been opened by the user simply by examining the compact usage map to determine which elements were set to "one."

Suppose that the content object owner wanted to know how many times the user had opened each chapter of the novel. In this case, the usage map might comprise, for a twelve-chapter novel, twelve elements each of which has a one-to-one correspondence with a different one of the twelve chapters of the novel. Each time a user opens a particular chapter, the corresponding METER method might increment the value contained in the corresponding usage map element. In this way,

an account could be readily maintained for each of the chapters of the novel.

The position of elements within a usage map may encode a multi-variable function. For example, the elements within a usage map may be arranged in a two-dimensional array as shown in Figure 25B. Different array coordinates could correspond to independent variables such as, for example, atomic elements and time. Suppose, as an example, that a content object owner distributes an object containing a collection of audio recordings. Assume further that the content object owner wants to track the number of times the user listens to each recording within the collection, and also wants to track usage based on month of the year. Thus, assume that the content object owner wishes to know how many times the user during the month of January listened to each of the recordings on a recording-by-recording basis, similarly wants to know this same information for the month of February, March, etc. In this case, the usage map (see Figure 25B) might be defined as a two-dimensional array of elements. One dimension of the array might encode audio recording number. The other dimension of the array might encode month of the year. During the month of January, the corresponding METER method would increment elements in the array in the "January" column of the array, selecting which element to increment as a function of recording number. When January

comes to an end, the METER method might cease writing into the array elements in the January column, and instead write values into a further set of February array elements—once again selecting the particular array element in this column as a function of recording number. This concept may be extended to N dimensions encoding N different variables.

Usage map meters are thus an efficient means for referencing prior usage. They may be used to enable certain VDE related security functions such as testing for contiguousness (including relative contiguousness), logical relatedness (including relative logical relatedness), usage randomization, and other usage patterns. For example, the degree or character of the "randomness" of content usage by a user might serve as a potential indicator of attempts to circumvent VDE content budget limitations. A user or groups of users might employ multiple sessions to extract content in a manner which does not violate contiguousness, logical relatedness or quantity limitations, but which nevertheless enables reconstruction of a material portion or all of a given, valuable unit of content. Usage maps can be analyzed to determine other patterns of usage for pricing such as, for example, quantity discounting after usage of a certain quantity of any or certain atomic units, or for enabling a user to reaccess an object for which the user previously paid for unlimited accesses (or unlimited accesses over a certain time

duration). Other useful analyses might include discounting for a given atomic unit for a plurality of uses.

A further example of a map meter includes storing a record of all applicable atomic elements that the user has paid to use (or alternatively, has been metered as having used, though payment may not yet have been required or made). Such a usage map would support a very efficient and flexible way to allow subsequent user usage of the same atomic elements.

A further usage map could be maintained to detect fraudulent usage of the same object. For example, the object might be stored in such a way that sequential access of long blocks should never occur. A METER method could then record all applicable atomic elements accesses during, for example, any specified increment of time, such as ten minutes, an hour, a day, a month, a year, or other time duration). The usage map could be analyzed at the end of the specified time increment to check for an excessively long contiguous set of accessed blocks, and/or could be analyzed at the initiation of each access to applicable atomic elements. After each time duration based analysis, if no fraudulent use is detected, the usage map could be cleared (or partially cleared) and the mapping process could begin in whole or in part anew. If a fraudulent use pattern is suspected or detected, that information might be recorded and the use of the

object could be halted. For example, the user might be required to contact a content provider who might then further analyze the usage information to determine whether or not further access should be permitted.

Figure 25c shows a particular type of "wide bit map" usage record 1206 wherein each entry in the usage record corresponds to usage during a particular time period (e.g., current month usage, last month's usage, usage in the month before last, etc.). The usage record shown thus comprises an array of "flags" or fields 1206, each element in the array being used to indicate usage in a different time period in this particular example. When a time period ends, all elements 1206 in the array may be shifted one position, and thus usage information (or the purchase of user access rights) over a series of time periods can be reflected by a series of successive array elements. In the specific example shown in Figure 25c, the entire wide array 1206 is shifted by one array position each month, with the oldest array element being deleted and the new array element being "turned" in a new array map corresponding to the current time period. In this example, record 1302 tracks usage access rights and/or other usage related activities during the present calendar month as well for the five immediately prior calendar months. Corresponding billing and/or billing method 406 may inspect the map, determine usage as related to billing and/or security monitoring for current usage

based on a formula that employs the usage data stored in the record, and updates the wide record to indicate the applicable array elements for which usage occurred or the like. A wide bit map may also be used for many other purposes such as maintaining an element by element count of usage, or the contiguousness, relatedness, etc. function described above, or some combination of functionality.

Audit trail maps may be generated at any frequency determined by control, meter, budget and billing methods and load modules associated with those methods. Audit trails have a similar structure to meters and budgets and they may contain user specific information in addition to information about the usage event that caused them to be created. Like meters and budgets, audit trails have a dynamic format that is defined by the content provider or their authorized designee, and share the basic element types for meters and budgets shown in the table above. In addition to these types, the following table lists some examples of other significant data fields that may be found in audit trails:

Field type	Format	Typical Use	Description of Use
Use Event ID	unsigned long	Meter/Budget/ Billing	Event ID that started a processing sequence.
Internal Sequence Number	unsigned long	Meter/Budget/ Billing	Transaction number to help detect audits that have been tampered with.

Field type	Format	Typical Use	Description of Use
Atomic Element(s) & Object ID	Unsigned integers) of appropriate width	Meter/Billing	Atomic element(s) and ID of object that was used.
Personal User Information	Character or other information	Budget/Billing	Personal information about user.
Use Date/Time	time_t	Meter/Budget/Billing	Date/time of use.
Site ID/User ID	VDE ID	Meter/Budget/Billing	VDE ID of user.

Audit trail records may be automatically combined into single records to conserve header space. The combination process may, for example, occur under control of a load module that creates individual audit trail records.

Permissions Record Overview

Figure 16 also shows that PERCs 808 may be stored as part of secure database 610. Permissions records ("PERCs") 808 are at the highest level of the data driven control hierarchy provided by the preferred embodiment of VDE 100. Basically, there is at least one PERC 808 that corresponds to each information and/or transactional content distributed by VDE 100. Thus, at least one PERC 808 exists for each VDE object 300 in the preferred embodiment. Some objects may have multiple

corresponding PERCs 808. PERC 808 controls how access and/or manipulation permissions are distributed and/or how content and/or other information may otherwise be used. PERC 808 also specifies the "rights" of each VDE participant in and to the content and/or other information.

In the preferred embodiment, no end user may use or access a VDE object unless a permissions record 808 has been delivered to the end user. As discussed above, a PERC 808 may be delivered as part of a traveling object 860 or it may be delivered separately (for example, within an administrative object). An electronic appliance 600 may not access an object unless a corresponding PERC 808 is present, and may only use the object and related information as permitted by the control structures contained within the PERC.

Briefly, the PERC 808 stores information concerning the methods, method options, decryption keys and rights with respect to a corresponding VDE object 300.

PERC 808 includes control structures that define high level categories or classifications of operations. These high level categories are referred to as "rights." The "right" control structures, in turn, provide internal control structures that reference "methods" 1000. The internal structure of preferred

embodiment PERC 808 organizes the "methods" that are required to perform each allowable operation on an object or associated control structure (including operations performed on the PERC itself). For example, PERC 808 contains decryption keys for the object, and usage of the keys is controlled by the methods that are required by the PERC for performing operations associated with the exercise of a "right."

PERC 808 for an object is typically created when the object is created, and future substantive modifications of a PERC, if allowed, are controlled by methods associated with operations using the distribution right(s) defined by the same (or different) PERC.

Figure 22 shows the internal structures present in an example of a PERC 808 provided by the preferred embodiment. All of the structures shown represent (or reference) collections of methods required to process a corresponding object in some specific way. PERCs 808 are organized as a hierarchical structure, and the basic elements of the hierarchy are as follows:

"rights" records 906

"control sets" 914

"required method" records 920 and

"required method options" 924.

There are other elements that may be included in a PERC 808 hierarchy that describe rules and the rule options to support the negotiation of rule sets and control information for smart objects and for the protection of a user's personal information by a privacy filter. These alternate elements may include:

- optional rights records
- optional control sets
- optional method records
- permitted rights records
- permitted rights control sets
- permitted method records
- required DTD descriptions
- optional DTD descriptions
- permitted DTD descriptions

These alternate fields can control other processes that may, in part, base negotiations or decisions regarding their operation on the contents of these fields. Rights negotiation, smart object control information, and related processes can use these fields for more precise control of their operation.

The PERC 808 shown in Figure 26 includes a PERC header 900, a CS0 ("control set 0") 902, private body keys 904, and one or more rights sub-records 906. Control set 0 902 in the preferred embodiment contains information that is common to one or more "rights" associated with an object 300. For example,

a particular "event" method or methods might be the same for usage rights, extraction rights and/or other rights. In that case, "control set 0" 902 may reference this event that is common across multiple "rights." The provision of "control set 0" 902 is actually an optimization, since it would be possible to store different instances of a commonly-used event within each of plural "rights" records 906 of a PERC 808.

Each rights record 906 defines a different "right" corresponding to an object. A "right" record 906 is the highest level of organization present in PERC 808. There can be several different rights in a PERC 808. A "right" represents a major functional partitioning desired by a participant of the basic architecture of VDE 100. For example, the right to use an object and the right to distribute rights to use an object are major functional groupings within VDE 100. Some examples of possible rights include access to content, permission to distribute rights to access content, the ability to read and process audit trails related to content and/or control structures, the right to perform transactions that may or may not be related to content and/or related control structures (such as banking transactions, catalog purchases, the collection of taxes, EDI transactions, and such), and the ability to change some or all of the internal structure of PERCs created for distribution to other users. PERC 808

contains a rights record 906 for each type of right to object access/use the PERC grants.

Normally, for VDE end users, the most frequently granted right is a usage right. Other types of rights include the "extraction right," the "audit right" for accessing audit trail information of end users, and a "distribution right" to distribute an object. Each of these different types of rights may be embodied in a different rights record 906 (or alternatively, different PERCs 808 corresponding to an object may be used to grant different rights).

Each rights record 906 includes a rights record header 908, a CSR ("control set for right") 910, one or more "right keys" 912, and one or more "control sets" 914. Each "rights" record 906 contains one or more control sets 914 that are either required or selectable options to control an object in the exercise of that "right." Thus, at the next level, inside of a "right" 906, are control sets 914. Control sets 914, in turn, each includes a control set header 916, a control method 918, and one or more required methods records 920. Required methods records 920, in turn, each includes a required method header 922 and one or more required method options 924.

Control sets 914 exist in two types in VDE 100: common required control sets which are given designations "control set 0" or "control set for right," and a set of control set options. "Control set 0" 902 contains a list of required methods that are common to all control set options, so that the common required methods do not have to be duplicated in each control set option. A "control set for right" ("CSR") 910 contains a similar list for control sets within a given right. "Control set 0" and any "control sets for rights" are thus, as mentioned above, optimizations; the same functionality for the control sets can be accomplished by listing all the common required methods in each control set option and omitting "control set 0" and any "control sets for rights."

One of the control set options, "control set 0" and the appropriate "control set for right" together form a complete control set necessary to exercise a right.

Each control set option contains a list of required methods 1000 and represents a different way the right may be exercised. Only one of the possible complete control sets 914 is used at any one time to exercise a right in the preferred embodiment.

Each control set 914 contains as many required methods records 920 as necessary to satisfy all of the requirements of the creators and/or distributors for the exercise of a right. Multiple

ways a right may be exercised, or multiple control sets that govern how a given right is exercised, are both supported. As an example, a single control set 914 might require multiple meter and budget methods for reading the object's content, and also require different meter and budget methods for printing an object's content. Both reading and printing an object's content can be controlled in a single control set 914.

Alternatively, two different control set options could support reading an object's content by using one control set option to support metering and budgeting the number of bytes read, and the other control set option to support metering and budgeting the number of paragraphs read. One or the other of these options would be active at a time.

Typically, each control set 914 will reference a set of related methods, and thus different control sets can offer a different set of method options. For example, one control set 914 may represent one distinct kind of metering methodology, and another control set may represent another, entirely different distinct metering methodology.

At the next level inside a control set 914 are the required methods records 920. Methods records 920 contain or reference methods 1000 in the preferred embodiment. Methods 1000 are a

collection of "events," references to load modules associated with these events, static data, and references to a secure database 610 for automatic retrieval of any other separately deliverable data elements that may be required for processing events (e.g., UDEs). A control set 914 contains a list of required methods that must be used to exercise a specific right (i.e., process events associated with a right). A required method record 920 listed in a control set 914 indicates that a method must exist to exercise the right that the control set supports. The required methods may reference "load modules" 1100 to be discussed below. Briefly, load modules 1100 are pieces of executable code that may be used to carry out required methods.

Each control set 914 may have a control method record 918 as one of its required methods. The referenced control method may define the relationships between some or all of the various methods 1000 defined by a control set 906. For example, a control method may indicate which required methods are functionally grouped together to process particular events, and the order for processing the required methods. Thus, a control method may specify that required method referenced by record 920(a)(1)(i) is the first to be called and then its output is to go to required method referenced by record 920(a)(1)(ii) and so on. In this way, a meter method may be tied to one or more billing

methods and then the billing methods may be individually tied to different budget methods, etc.

Required method records 920 specify one or more required method options 924. Required method options are the lowest level of control structure in a preferred embodiment PERC 808. By parameterizing the required methods and specifying the required method options 924 independently of the required methods, it becomes possible to reuse required methods in many different circumstances.

For example, a required method record 920 may indicate that an actual budget method ID must be chosen from the list of budget method IDs in the required method option list for that required method. Required method record 920 in this case does not contain any method IDs for information about the type of method required, it only indicates that a method is required. Required method option 924 contains the method ID of the method to be used if this required method option is selected. As a further optimization, an actual method ID may be stored if only one option exists for a specific required method. This allows the size of this data structure to be decreased.

PERC 808 also contains the fundamental decryption keys for an object 300, and any other keys used with "rights" (for

encoding and/or decoding audit trails, for example). It may contain the keys for the object content or keys to decrypt portions of the object that contain other keys that then can be used to decrypt the content of the object. Usage of the keys is controlled by the control sets 914 in the same "right" 906 within PERC 808.

In more detail, Figure 26 shows PERC 808 as including private body keys 904, and right keys 912. Private body keys 904 are used to decrypt information contained within a private body 806 of a corresponding VDE object 300. Such information may include, for example, methods 1000, load modules 1100 and/or UDEs 1200, for example. Right keys 912 are keys used to exercise a right in the preferred embodiment. Such right keys 912 may include, for example, decryption keys that enable a method specified by PERC 808 to decrypt content for release by a VDE node to an end user. These right keys 912 are, in the preferred embodiment, unique to an object 300. Their usage is preferably controlled by budgets in the preferred embodiment.

Detailed Example of a PERC 808

Figures 26A and 26B show one example of a preferred embodiment PERC 808. In this example, PERC header 900 includes:

a site record number 926,

a field 928 specifying the length of the private body key block,
a field 930 specifying the length of the PERC,
an expiration date/time field 932 specifying the expiration date and/or time for the PERC,
a last modification date/time field 934 specifying the last date and/or time the PERC 808 was modified,
the original distributor ID field 936 that specifies who originally distributed the PERC and/or corresponding object,
a last distributor field 938 that specifies who was the last distributor of the PERC and/or the object,
an object ID field 940 identifying the corresponding VDE object 300,
a field 942 that specifies the class and/or type of PERC and/or the instance ID for the record class to differentiate the PERCs of the same type that may differ in their particulars,
a field 944 specifying the number of "rights" sub-records 906 within the PERC, and
a validation tag 948.

The PERC 808 shown in Figures 26a, 26b also has private body keys stored in a private body key block 950.

This PERC 808 includes a control set 0 sub-record 914 (0) that may be used commonly by all of rights 906 within the PERC.

This control set 0 record 914(0) may include the following fields:

- a length field 952 specifying the length of the control set 0 record

- a field 954 specifying the number of required method records 920 within the control set

- an access tag field 956 specifying an access tag to control modification of the record and one or more required method records 920.

Each required method record 920, in turn may include:

- a length field 958 specifying the length of the required method record

- a field 960 specifying the number of method option records within the required method record 920

- an access tag field 962 specifying an access tag to control modification of the record and one or more method option records 924.

Each method option sub-record 924 may include:

- a length field 964 specifying the length of the method option record

- a length field 966 specifying the length of the data area (if any) corresponding to the method option record

a method ID field 968 specifying a method ID (e.g.,
type/owner/class/instance)
a correlation tag field 970 specifying a correlation
tag for correlating with the method specified in
field 968
an access tag field 972 specifying an access tag to
control modification of this record
a method-specific attributes field 974
a data area 976 and
a check value field 978 for validation purposes

In this example of PERC 808 also includes one or more
rights records 906, and an overall check value field 980. Figure
23b is an example of one of right records 906 shown in Figure
16a. In this particular example, rights record 906a includes a
rights record header 908 comprising:

a length field 982 specifying the length of the rights
key block 912
a length field 984 specifying the length of the rights
record 908
an expiration date/time field 986 specifying the
expiration date and/or time for the rights
record
a right ID field 988 identifying a right

a number field 990 specifying the number of control sets 914 within the rights record 906, and an access tag field 992 specifying an access tag to control modification of the right record.

This example of rights record 906 includes:

a control set for this right (CSR) 910
a rights key block 912
one or more control sets 914, and
a check value field 994.

Object Registry

Referring once again to Figure 16, secure database 610 provides data structures that support a "lookup" mechanism for "registered" objects. This "lookup" mechanism permits electronic appliance 600 to associate, in a secure way, VDE objects 300 with PERCs 808, methods 1000 and load modules 1100. In the preferred embodiment, this lookup mechanism is based in part on data structures contained within object registry 450.

In one embodiment, object registry 450 includes the following tables:

- an object registration table 460;
- a subject table 462;
- a User Rights Table ("URT") 464;

- an Administrative Event Log 442;
- a shipping table 444; and
- a receiving table 446.

Object registry 460 in the example embodiment is a database of information concerning registered VDE objects 300 and the rights of users and user groups with regard to those objects. When electronic appliance 600 receives an object 300 containing a new budget or load module 1100, the electronic appliance usually needs to add the information contained by the object to secure database 610. Moreover, when any new VDE object 300 arrives at an electronic appliance 600, the electronic appliance must "register" the object within object registry 450 so that it can be accessed. The lists and records for a new object 300 are built in the preferred embodiment when the object is "registered" by the electronic appliance 600. The information for the object may be obtained from the object's encrypted private header, object body, and encrypted name services record. This information may be extracted or derived from the object 300 by SPE 503, and then stored within secure database 610 as encrypted records.

In one embodiment, object registration table 460 includes information identifying objects within object storage (repository) 728. These VDE objects 300 stored within object storage 728 are

not, in the example embodiment, necessarily part of secure database 610 since the objects typically incorporate their own security (as necessary and required) and are maintained using different mechanisms than the ones used to maintain the secure database. Even though VDE objects 300 may not strictly be part of secure database 610, object registry 450 (and in particular, object registration table 460) refers to the objects and thus "incorporates them by reference" into the secure database. In the preferred embodiment, an electronic appliance 600 may be disabled from using any VDE object 300 that has not been appropriately registered with a corresponding registration record stored within object registration table 460.

Subject table 462 in the example embodiment establishes correspondence between objects referred to by object registration table 460 and users (or groups of users) of electronic appliance 600. Subject table 462 provides many of the attributes of an access control list ("ACL"), as will be explained below.

User rights table 464 in the example embodiment provides permissioning and other information specific to particular users or groups of users and object combinations set forth in subject table 462. In the example embodiment, permissions records 808 (also shown in Figure 16 and being stored within secure database 610) may provide a universe of permissioning for a particular

object-user combination. Records within user rights table 464 may specify a sub-set of this permissioning universe based on, for example, choices made by users during interaction at time of object registration.

Administrative event log 442, shipping table 444, and receiving table 446 provide information about receipts and deliveries of VDE objects 300. These data structures keep track of administrative objects sent or received by electronic appliance 600 including, for example, the purpose and actions of the administrative objects in summary and detailed form. Briefly, shipping table 444 includes a shipping record for each administrative object sent (or scheduled to be sent) by electronic appliance 600 to another VDE participant. Receiving table 446 in the preferred embodiment includes a receiving record for each administrative object received (or scheduled to be received) by electronic appliance 600. Administrative event log 442 includes an event log record for each shipped and each received administrative object, and may include details concerning each distinct event specified by received administrative objects.

Administrative Object Shipping and Receiving

Figure 27 is an example of a detailed format for a shipping table 444. In the preferred embodiment, shipping table 444 includes a header 444A and any number of shipping records 445.

Header 444A includes information used to maintain shipping table 444. Each shipping record 445 within shipping table 444 provides details concerning a shipping event (i.e., either a completed shipment of an administrative object to another VDE participant, or a scheduled shipment of an administrative object).

In the example embodiment of the secure database 610, shipping table header 444A may include a site record number 444A(1), a user (or group) ID 444A(2), a series of reference fields 444A(3)-444A(6), validation tags 444A(7)-444A(8), and a check value field 444A(9). The fields 444A(3)-444A(6) reference certain recent IDs that designate lists of shipping records 445 within shipping table 444. For example, field 444A(3) may reference to a "first" shipping record representing a completed outgoing shipment of an administrative object, and field 444A(4) may reference to a "last" shipping record representing a completed outgoing shipment of an administrative object. In this example, "first" and "last" may, if desired, refer to time or order of shipment as one example. Similarly, fields 444A(5) and 444A(6) may reference to "first" and "last" shipping records for scheduled outgoing shipments. Validation tag 444A(7) may provide validation from a name services record within name services record table 452 associated with the user (group) ID in the header. This permits access from the shipping record back to the

name services record that describes the sender of the object described by the shipping records. Validation tag 444A(8) provides validation for a "first" outgoing shipping record referenced by one or more of pointers 444A(3)-444A(6). Other validation tags may be provided for validation of scheduled shipping record(s).

Shipping record 444(1) shown includes a site record number 445(1)(A). It also includes first and last scheduled shipment date/times 445(1)(B), 445(1)(C) providing a window of time used for scheduling administrative object shipments. Field 445(1)(D) may specify an actual date/time of a completed shipment of an administrative object. Field 445(1)(E) provides an ID of an administrative object shipped or to be shipped, and thus identifies which administrative object within object storage 728 pertains to this particular shipping record. A reference field 445(1)(G) references a name services record within name services record table 452 specifying the actual or intended recipient of the administrative object shipped or to be shipped. This information within name services record table 452 may, for example, provide routing information sufficient to permit outgoing administrative objects manager 754 shown in Figure 12 to inform object switch 734 to ship the administrative object to the intended recipient. A field 445(1)(H) may specify (e.g., using a series of bit flags) the purpose of the administrative object shipment, and a field

445(1)(I) may specify the status of the shipment. Reference fields 445(1)(J), 445(1)(K) may reference "previous" and "next" shipping records 445 in a linked list (in the preferred embodiment, there may be two linked lists, one for completed shipping records and the other for scheduled shipping records). Fields 445(1)(L) - 445(1)(P) may provide validation tags respectively from header 444A, to a record within administrative event log 442 pointed to by pointer 445(1)(F); to the name services record referenced by field 445(1)(G); from the previous record referenced by 445(1)(J); and to the next record referenced by field 445(1)(K). A check value field 445(1)(Q) may be used for validating shipping record 445.

Figure 28 shows an example of one possible detailed format for a receiving table 446. In one embodiment, receiving table 446 has a structure that is similar to the structure of the shipping table 444 shown in Figure 27. Thus, for example, receiving table 446 may include a header 446a and a plurality of receiving records 447, each receiving record including details about a particular reception or scheduled reception of an administrative object. Receiving table 446 may include two linked lists, one for completed receptions and another for schedule receptions. Receiving table records 447 may each reference an entry within name services record table 452 specifying an administrative object sender, and may each point to an entry within

administrative event log 442. Receiving records 447 may also include additional details about scheduled and/or completed reception (e.g., scheduled or actual date/time of reception, purpose of reception and status of reception), and they may each include validation tags for validating references to other secure database records.

Figure 29 shows an example of a detailed format for an administrative event log 442. In the preferred embodiment, administrative event log 442 includes an event log record 442(1) . . . 442(N) for each shipped administrative object and for each received administrative object. Each administrative event log record may include a header 443a and from 1 to N sub-records 442(J)(1) . . . 442(J)(N). In the preferred embodiment, header 443a may include a site record number field 443A(1), a record length field 443A(2), an administrative object ID field 443A(3), a field 443A(4) specifying a number of events, a validation tag 443A(5) from shipping table 444 or receiving table 446, and a check sum field 443A(6). The number of events specified in field 443A(4) corresponds to the number of sub-records 442(J)(1) . . . 442(J)(N) within the administrative event log record 442(J). Each of these sub-records specifies information about a particular "event" affected or corresponding to the administrative object specified within field 443(A)(3). Administrative events are retained in the administrative event log 442 to permit the

reconstruction (and preparation for construction or processing) of the administrative objects that have been sent from or received by the system. This permits lost administrative objects to be reconstructed at a later time.

Each sub-record may include a sub-record length field 442(J)(1)(a), a data area length field 442(J)(1)(b), an event ID field 442(J)(1)(c), a record type field 442(J)(1)(d), a record ID field 442(J)(1)(e), a data area field 442(J)(1)(f), and a check value field 442(J)(1)(g). The data area 442(J)(1)(f) may be used to indicate which information within secure database 610 is affected by the event specified in the event ID field 442(J)(1)(c), or what new secure database item(s) were added, and may also specify the outcome of the event.

The object registration table 460 in the preferred embodiment includes a record corresponding to each VDE object 300 within object storage (repository) 728. When a new object arrives or is detected (e.g., by redirector 684), a preferred embodiment electronic appliance 600 "registers" the object by creating an appropriate object registration record and storing it in the object registration table 460. In the preferred embodiment, the object registration table stores information that is user-independent, and depends only on the objects that are registered at a given VDE electronic appliance 600. Registration activities

are typically managed by a REGISTER method associated with an object.

In the example, subject table 462 associates users (or groups of users) with registered objects. The example subject table 462 performs the function of an access control list by specifying which users are authorized to access which registered VDE objects 300.

As described above, secure database 610 stores at least one PERC 808 corresponding to each registered VDE object 300. PERCS 808 specify a set of rights that may be exercised to use or access the corresponding VDE object 300. The preferred embodiment allows user to "customize" their access rights by selecting a subset of rights authorized by a corresponding PERC 808 and/or by specifying parameters or choices that correspond to some or all of the rights granted by PERC 808. These user choices are set forth in a user rights table 464 in the preferred embodiment. User rights table (URT) 464 includes URT records, each of which corresponds to a user (or group of users). Each of these URT records specifies user choices for a corresponding VDE object 300. These user choices may, either independently or in combination with a PERC 808, reference one or more methods 1000 for exercising the rights granted to the user by the PERC

808 in a way specified by the choices contained within the URT record.

Figure 30 shows an example of how these various tables may interact with one another to provide a secure database lookup mechanism. Figure 30 shows object registration table 460 as having a plurality of object registration records 460(1), 460(2), These records correspond to VDE objects 300(1), 300(2), . . . stored within object repository 728. Figure 31 shows an example format for an object registration record 460 provided by the preferred embodiment. Object registration record 460(N) may include the following fields:

- site record number field 466(1)
- object type field 466(2)
- creator ID field 466(3)
- object ID field 466(4)
- a reference field 466(5) that references subject table 462
- an attribute field 466(6)
- a minimum registration interval field 466(7)
- a tag 466(8) to a subject table record, and
- a check value field 466(9).

The site record number field 466(1) specifies the site record number for this object registration record 460(N). In one embodiment of secure database 610, each record stored within

the secure database is identified by a site record number. This site record number may be used as part of a database lookup process in order to keep track of all of the records within the secure database 610.

Object type field 466(2) may specify the type of registered VDE object 300 (e.g., a content object, an administrative object, etc.).

Creator ID field 466(3) in the example may identify the creator of the corresponding VDE object 300.

Object ID field 466(4) in the example uniquely identifies the registered VDE object 300.

Reference field 466(5) in the preferred embodiment identifies a record within the subject table 462. Through use of this reference, electronic appliance 600 may determine all users (or user groups) listed in subject table 462 authorized to access the corresponding VDE object 300. Tag 466(8) is used to validate that the subject table records accessed using field 466(5) is the proper record to be used with the object registration record 460(N).

Attribute field 466(6) may store one or more attributes or attribute flags corresponding to VDE object 300.

Minimum registration interval field 466(7) may specify how often the end user may re-register as a user of the VDE object 300 with a clearinghouse service, VDE administrator, or VDE provider. One reason to prevent frequent re-registration is to foreclose users from reusing budget quantities in traveling objects until a specified amount of time has elapsed. The minimum registration interval field 466(7) may be left unused when the object owner does not wish to restrict re-registration.

Check value field 466(9) contains validation information used for detecting corruption or modification of record 460(N) to ensure security and integrity of the record. In the preferred embodiment, many or all of the fields within record 460(N) (as with other records within the secure database 610) may be fully or partially encrypted and/or contain fields that are stored redundantly in each record (once in unencrypted form and once in encrypted form). Encrypted and unencrypted versions of the same fields may be cross checked at various times to detect corruption or modification of the records.

As mentioned above, reference field 466(5) references subject table 462, and in particular, references one or more

user/object records 460(M) within the subject table. Figure 32 shows an example of a format for a user/object record 462(M) provided by the example. Record 462(M) may include a header 468 and a subject record portion 470. Header 468 may include a field 468(6) referencing a "first" subject record 470 contained within the subject registration table 462. This "first" subject record 470(1) may, in turn, include a reference field 470(5) that references a "next" subject record 470(2) within the subject registration table 462, and so on. This "linked list" structure permits a single object registration record 460(N) to reference to from one to N subject records 470.

Subject registration table header 468 in the example includes a site record number field 468(1) that may uniquely identify the header as a record within secure database 610. Header 468 may also include a creator ID field 468(2) that may be a copy of the content of the object registration table creator ID field 466(3). Similarly, subject registration table header 468 may include an object ID field 468(5) that may be a copy of object ID field 466(4) within object registration table 460. These fields 468(2), 468(5) make user/object registration records explicitly correspond to particular VDE objects 300.

Header 468 may also include a tag 468(7) that permits validation. In one example arrangement, the tag 468(7) within

the user/object registration header 468 may be the same as the tag 466(8) within the object registration record 460(N) that points to the user/object registration header. Correspondence between these tags 468(7) and 466(8) permits validation that the object registration record and user/object registration header match up.

User/object header 468 also includes an original distributor ID field 468(3) indicating the original distributor of the corresponding VDE object 300, and the last distributor ID field 468(4) that indicates the last distributor within the chain of handling of the object prior to its receipt by electronic appliance 600.

Header 468 also includes a tag 468(8) allowing validation between the header and the "first" subject record 470(1) which field 468(6) references

Subject record 470(1) includes a site record number 472(1), a user (or user group) ID field 472(2), a user (or user group) attributes field 472(3), a field 472(4) referencing user rights table 464, a field 472(5) that references to the "next" subject record 470(2) (if there is one), a tag 472(6) used to validate with the header tag 468(8), a tag 472(7) used to validate with a corresponding tag in the user rights table record referenced by field 472(4), a tag 472(9) used to validate with a tag in the "next"

subject record referenced to by field 472(5) and a check value field 472(9).

User or user group ID 472(2) identifies a user or a user group authorized to use the object identified in field 468(5). Thus, the fields 468(5) and 472(2) together form the heart of the access control list provided by subject table 462. User attributes field 472(3) may specify attributes pertaining to use/access to object 300 by the user or user group specified in fields 472(2). Any number of different users or user groups may be added to the access control list (each with a different set of attributes 472(3)) by providing additional subject records 470 in the "linked list" structure.

Subject record reference field 472(4) references one or more records within user rights table 464. Figure 33 shows an example of a preferred format for a user rights table record 464(k). User rights record 464(k) may include a URT header 474, a record rights header 476, and a set of user choice records 478. URT header 474 may include a site record number field, a field 474(2) specifying the number of rights records within the URT record 464(k), a field 474(3) referencing a "first" rights record (i.e., to rights record header 476), a tag 474(4) used to validate the lookup from the subject table 462, a tag 474(5) used to

validate the lookup to the rights record header 476, and a check value field 474(6).

Rights record header 476 in the preferred embodiment may include site record number field 476(1), a right ID field 476(2), a field 476(3) referencing the "next" rights record 476(2), a field 476(4) referencing a first set of user choice records 478(1), a tag 476(5) to allow validation with URT header tag 474(5), a tag 476(6) to allow validation with a user choice record tag 478(6), and a check value field 476(7). Right ID field 476(2) may, for example, specify the type of right conveyed by the rights record 476(e.g., right to use, right to distribute, right to read, right to audit, etc.).

The one or more user choice records 478 referenced by rights record header 476 sets forth the user choices corresponding to access and/or use of the corresponding VDE object 300. There will typically be a rights record 476 for each right authorized to the corresponding user or user group. These rights govern use of the VDE object 300 by that user or user group. For instance, the user may have an "access" right, and an "extraction" right, but not a "copy" right. Other rights controlled by rights record 476 (which is derived from PERC 808 using a REGISTER method in the preferred embodiment) include distribution rights, audit rights, and pricing rights. When an

object 300 is registered with the electronic appliance 600 and is registered with a particular user or user group, the user may be permitted to select among various usage methods set forth in PERC 808. For instance, a VDE object 300 might have two required meter methodologies: one for billing purposes, and one for accumulating data concerning the promotional materials used by the user. The user might be given the choice of a variety of meter/billing methods, such as: payment by VISA or MasterCard; choosing between billing based upon the quantity of material retrieved from an information database, based on the time of use, and/or both. The user might be offered a discount on time and/or quantity billing if he is willing to allow certain details concerning his retrieval of content to be provided to third parties (e.g., for demographic purposes). At the time of registration of an object and/or user for the object, the user would be asked to select a particular meter methodology as the "active metering method" for the first acquired meter. A VDE distributor might narrow the universe of available choices for the user to a subset of the original selection array stipulated by PERC 808. These user selection and configuration settings are stored within user choice records 480(1), 480(2), 480(N). The user choice records need not be explicitly set forth within user rights table 464; instead, it is possible for user choice records 480 to refer (e.g., by site reference number) to particular VDE methods and/or information parameterizing those methods. Such reference by user choice

records 480 to method 1000 should be validated by validation tags contained within the user choice records. Thus, user choice records 480 in the preferred embodiment may select one or more methods 1000 for use with the corresponding VDE object 300 (as is shown in Figure 27). These user choice records 480 may themselves fully define the methods 1000 and other information used to build appropriate components assemblies 690 for implementing the methods. Alternatively, the user/object record 462 used to reference the user rights record 464 may also reference the PERC 808 corresponding to VDE object 300 to provide additional information needed to build the component assembly 690 and/or otherwise access the VDE object 300. For example, PERC 808 may be accessed to obtain MDEs 1202 pertaining to the selected methods, private body and/or rights keys for decrypting and/or encrypting object contents, and may also be used to provide a checking capability ensuring that the user rights record conveys only those rights authorized by a current authorization embodied within a PERC.

In one embodiment provided by the present invention, a conventional database engine may be used to store and organize secure database 610, and the encryption layers discussed above may be "on top of" the conventional database structure. However, if such a conventional database engine is unable to organize the records in secure database 610 and support the

security considerations outlined above, then electronic appliance 600 may maintain separate indexing structures in encrypted form. These separate indexing structures can be maintained by SPE 503. This embodiment would require SPE 503 to decrypt the index and search decrypted index blocks to find appropriate "site record IDs" or other pointers. SPE 503 might then request the indicated record from the conventional database engine. If the record ID cannot be checked against a record list, SPE 503 might be required to ask for the data file itself so it can retrieve the desired record. SPE 503 would then perform appropriate authentication to ensure that the file has not been tampered with and that the proper block is returned. SPE 503 should not simply pass the index to the conventional database engine (unless the database engine is itself secure) since this would allow an incorrect record to be swapped for the requested one.

Figure 34 is an example of how the site record numbers described above may be used to access the various data structures within secure database 610. In this example, secure database 610 further includes a site record table 482 that stores a plurality of site record numbers. Site record table 482 may store what is in effect a "master list" of all records within secure database 610. These site record numbers stored by site record table 482 permit any record within secure database 610 to be accessed. Thus, some of the site records within site record table

482 may index records with an object registration table 460, other site record numbers within the site record table may index records within the user/object table 462, still other site record numbers within the site record table may access records within URT 464, and still other site record numbers within the site record table may access PERCs 808. In addition, each of method cores 1000' may also include a site record number so they may be accessed by site record table 482.

Figure 34A shows an example of a site record 482(j) within site record table 482. Site record 482(j) may include a field 484(1) indicating the type of record, a field 484(2) indicating the owner or creator of the record, a "class" field 484(3) and an "instance" field 484(4) providing additional information about the record to which the site record 482(j) points; a specific descriptor field 484(5) indicating some specific descriptor (e.g., object ID) associated with the record; an identification 484(6) of the table or other data structure which the site record references; a reference and/or offset within that data structure indicating where the record begins; a validation tag 484(8) for validating the record being looked up, and a check value field 484(9). Fields 484(6) and 484(7) together may provide the mechanism by which the record referenced to by the site record 484(j) is actually physically located within the secure database 610.

Updating Secure Database 610

Figure 35 show an example of a process 1150 which can be used by a clearinghouse, VDE administrator or other VDE participant to update the secure database 610 maintained by an end user's electronic appliance 600. For example, the process 1500 shown in Figure 35 might be used to collect "audit trail" records within secure database 610 and/or provide new budgets and permissions (e.g., PERCs 808) in response to an end user's request.

Typically, the end user's electronic appliance 600 may initiate communications with a clearinghouse (Block 1152). This contact may, for example, be established automatically or in response to a user command. It may be initiated across the electronic highway 108, or across other communications networks such as a LAN, WAN, two-way cable or using portable media exchange between electronic appliances. The process of exchanging administrative information need not occur in a single "on line" session, but could instead occur over time based on a number of different one-way and/or two-way communications over the same or different communications means. However, the process 1150 shown in Figure 35 is a specific example where the end user's electronic appliance 600 and the other VDE participant (e.g., a clearinghouse) establish a two-way real-time

interactive communications exchange across a telephone line, network, electronic highway 108, etc.

The end user's electronic appliance 600 generally contacts a particular VDE administrator or clearinghouse. The identity of the particular clearinghouse is based on the VDE object 300 the user wishes to access or has already accessed. For example, suppose the user has already accessed a particular VDE object 300 and has run out of budget for further access. The user could issue a request which will cause her electronic appliance 600 to automatically contact the VDE administrator, distributor and/or financial clearinghouse that has responsibility for that particular object. The identity of the appropriate VDE participants to contact is provided in the example by information within UDEs 1200, MDEs 1202, the Object Registration Table 460 and/or Subject Table 462, for example. Electronic appliance 600 may have to contact multiple VDE participants (e.g., to distribute audit records to one participant, obtain additional budgets or other permissions from another participant, etc.). The contact 1152 may in one example be scheduled in accordance with the Figure 27 Shipping Table 444 and the Figure 29 Administrative Event Log 442.

Once contact is established, the end user's electronic appliance and the clearinghouse typically authenticate one

another and agree on a session key to use for the real-time information exchange (Block 1154). Once a secure connection is established, the end user's electronic appliance may determine (e.g., based on Shipping Table 444) whether it has any administrative object(s) containing audit information that it is supposed to send to the clearinghouse (decision Block 1156). Audit information pertaining to several VDE objects 300 may be placed within the same administrative object for transmission, or different administrative objects may contain audit information about different objects. Assuming the end user's electronic appliance has at least one such administrative object to send to this particular clearinghouse ("yes" exit to decision Block 1156), the electronic appliance sends that administrative object to the clearinghouse via the now-established secure real-time communications (Block 1158). In one specific example, a single administrative object may be sent an administrative object containing audit information pertaining to multiple VDE objects, with the audit information for each different object comprising a separate "event" within the administrative object.

The clearinghouse may receive the administrative object and process its contents to determine whether the contents are "valid" and "legitimate." For example, the clearinghouse may analyze the contained audit information to determine whether it indicates misuse of the applicable VDE object 300. The

clearinghouse may, as a result of this analysis, may generate one or more responsive administrative objects that it then sends to the end user's electronic appliance 600 (Block 1160). The end user's electronic appliance 600 may process events that update its secure database 610 and/or SPU 500 contents based on the administrative object received (Block 1162). For example, if the audit information received by the clearinghouse is legitimate, then the clearinghouse may send an administrative object to the end user's electronic appliance 600 requesting the electronic appliance to delete and/or compress the audit information that has been transferred. Alternatively or in addition, the clearinghouse may request additional information from the end-user electronic appliance 600 at this stage (e.g., retransmission of certain information that was corrupted during the initial transmission, transmission of additional information not earlier transmitted, etc.). If the clearinghouse detects misuse based on the received audit information, it may transmit an administrative object that revokes or otherwise modifies the end user's right to further access the associated VDE objects 300.

The clearinghouse may, in addition or alternatively, send an administrative object to the end user's electronic appliance 600 that instructs the electronic appliance to display one or more messages to the user. These messages may inform the user about certain conditions and/or they may request additional

information from the user. For example, the message may instruct the end user to contact the clearinghouse directly by telephone or otherwise to resolve an indicated problem, enter a PIN, or it may instruct the user to contact a new service company to re-register the associated VDE object. Alternatively, the message may tell the end user that she needs to acquire new usage permissions for the object, and may inform the user of cost, status and other associated information.

During the same or different communications exchange, the same or different clearinghouse may handle the end user's request for additional budget and/or permission pertaining to VDE object 300. For example, the end user's electronic appliance 600 may (e.g., in response to a user input request to access a particular VDE object 300) send an administrative object to the clearinghouse requesting budgets and/or other permissions allowing access (Block 1164). As mentioned above, such requests may be transmitted in the form of one or more administrative objects, such as, for example, a single administrative object having multiple "events" associated with multiple requested budgets and/or other permissions for the same or different VDE objects 300. The clearinghouse may upon receipt of such a request, check the end user's credit, financial records, business agreements and/or audit histories to determine whether the requested budgets and/or permissions should be given. The

clearinghouse may, based on this analysis, send one or more responsive administrative objects which cause the end user's electronic appliance 600 to update its secure database in response (Block 1166, 1168). This updating might, for example, comprise replacing an expired PERC 808 with a fresh one, modifying a PERC to provide additional (or lesser) rights, etc. Steps 1164-1168 may be repeated multiple times in the same or different communications session to provide further updates to the end user's secure database 610.

Figure 36 shows an example of how a new record or element may be inserted into secure database 610. The load process 1070 shown in Figure 35 checks each data element or item as it is loaded to ensure that it has not been tampered with, replaced or substituted. In the process 1070 shown in Figure 35, the first step that is performed is to check to see if the current user of electronic appliance 600 is authorized to insert the item into secure database 610 (block 1072). This test may involve, in the preferred embodiment, loading (or using already loaded) appropriate methods 1000 and other data structures such as UDEs 1200 into an SPE 503, which then authenticates user authorization to make the change to secure database 610 (block 1074). If the user is approved as being authorized to make the change to secure database 610, then SPE 503 may check the integrity of the element to be added to the secure database by

decrypting it (block 1076) and determining whether it has become damaged or corrupted (block 1078). The element is checked to ensure that it decrypts properly using a predetermined management file key, and the check value may be validated. In addition, the public and private header ID tags (if present) may be compared to ensure that the proper element has been provided and had not been substituted, and the unique element tag ID compared against the predetermined element tag. If any of these tests fail, the element may be automatically rejected, error corrected, etc. Assuming the element is found to have integrity, SPE 503 may re-encrypt the information (block 1080) using a new key for example (see Figure 37 discussion below). In the same process step an appropriate tag is preferably provided so that the information becomes encrypted within a security wrapper having appropriate tags contained therein (block 1082). SPE 503 may retain appropriate tag information so that it can later validate or otherwise authenticate the item when it is again read from secure database 610 (block 1084). The now-secure element within its security wrapper may then be stored within secure database 610.

Figure 37 shows an example of a process 1050 used in the preferred embodiment database to securely access an item stored in secure database 610. In the preferred embodiment, SPE 503 first accesses and reads in the item from secure database 610

records. SPE 503 reads this information from secure database 610 in encrypted form, and may "unwrap" it (block 1052) by decrypting it (block 1053) based on access keys internally stored within the protected memory of an SPU 500. In the preferred embodiment, this "unwrap" process 1052 involves sending blocks of information to encrypt/decrypt engine 522 along with a management file key and other necessary information needed to decrypt. Decrypt engine 522 may return "plaintext" information that SPE 503 then checks to ensure that the security of the object has not been breached and that the object is the proper object to be used (block 1054). SPE 503 may then check all correlation and access tags to ensure that the read-in element has not been substituted and to guard against other security threats (block 1054). Part of this "checking" process involves checking the tags obtained from the secure database 610 with tags contained within the secure memory or an SPU 500 (block 1056). These tags stored within SPU 500 may be accessed from SPU protected memory (block 1056) and used to check further the now-unwrapped object. Assuming this "checking" process 1054 does not reveal any improprieties (and block 1052 also indicates that the object has not become corrupted or otherwise damaged), SPE 503 may then access or otherwise use the item (block 1058). Once use of the item is completed, SPE 503 may need to store the item back into secure database 610 if it has changed. If the item has changed, SPE 503 will send the item in its changed form to

encrypt/decrypt engine 522 for encryption (block 1060), providing the appropriate necessary information to the encrypt/decrypt engine (e.g., the appropriate same or different management file key and data) so that the object is appropriately encrypted. A unique, new tag and/or encryption key may be used at this stage to uniquely tag and/or encrypt the item security wrapper (block 1062; see also detailed Figure 37 discussion below). SPE 503 may retain a copy of the key and/or tag within a protected memory of SPU 500 (block 1064) so that the SPE can decrypt and validate the object when it is again read from secure database 610.

The keys to decrypt secure database 610 records are, in the preferred embodiment, maintained solely within the protected memory of an SPU 500. Each index or record update that leaves the SPU 500 may be time stamped, and then encrypted with a unique key that is determined by the SPE 503. For example, a key identification number may be placed "in plain view" at the front of the records of secure database 610 so the SPE 503 can determine which key to use the next time the record is retrieved. SPE 503 can maintain the site ID of the record or index, the key identification number associated with it, and the actual keys in the list internal to the SPE. At some point, this internal list may fill up. At this point, SPE 503 may call a maintenance routine that re-encrypts items within secure database 610 containing

changed information. Some or all of the items within the data structure containing changed information may be read in, decrypted, and then re-encrypted with the same key. These items may then be issued the same key identification number. The items may then be written out of SPE 503 back into secure database 610. SPE 503 may then clear the internal list of item IDs and corresponding key identification numbers. It may then begin again the process of assigning a different key and a new key identification number to each new or changed item. By using this process, SPE 503 can protect the data structures (including the indexes) of secure database 610 against substitution of old items and against substitution of indexes for current items. This process also allows SPE 503 to validate retrieved item IDs against the encrypted list of expected IDs.

Figure 38 is a flowchart showing this process in more detail. Whenever a secure database 610 item is updated or modified, a new encryption key can be generated for the updated item. Encryption using a new key is performed to add security and to prevent misuse of backup copies of secure database 610 records. The new encryption key for each updated secure database 610 record may be stored in SPU 500 secure memory with an indication of the secure database record or record(s) to which it applies.

SPE 503 may generate a new encryption/decryption key for each new item it is going to store within secure database 610 (block 1086). SPE 503 may use this new key to encrypt the record prior to storing it in the secure database (block 1088). SPE 503 make sure that it retains the key so that it can later read and decrypt the record. Such decryption keys are, in the preferred embodiment, maintained within protected non-volatile memory (e.g., NVRAM 534b) within SPU 500. Since this protected memory has a limited size, there may not be enough room within the protected memory to store a new key. This condition is tested for by decision block 1090 in the preferred embodiment. If there is not enough room in memory for the new key (or some other event such as the number of keys stored in the memory exceeding a predetermined number, a timer has expired, etc.), then the preferred embodiment handles the situation by re-encrypting other records with secure database 610 with the same new key in order to reduce the number of (or change) encryption/decryption keys in use. Thus, one or more secure database 610 items may be read from the secure database (block 1092), and decrypted using the old key(s) used to encrypt them the last time they were stored. In the preferred embodiment, one or more "old keys" are selected, and all secure database items encrypted using the old key(s) are read and decrypted. These records may now be re-encrypted using the new key that was generated at block 1086 for the new record (block 1094). The old

key(s) used to decrypt the other record(s) may now be removed from the SPU protected memory (block 1096), and the new key stored in its place (block 1097). The old key(s) cannot be removed from secure memory by block 1096 unless SPE 503 is assured that all records within the secure database 610 that were encrypted using the old key(s) have been read by block 1092 and re-encrypted by block 1904 using the new key. All records encrypted (or re-encrypted) using the new key may now be stored in secure database 610 (block 1098). If decision block 1090 determines there is room within the SPU 500 protected memory to store the new key, then the operations of blocks 1092, 1094, 1096 are not needed and SPE 503 may instead simply store the new key within the protected memory (block 1097) and store the new encrypted records into secure database 610 (block 1098).

The security of secure database 610 files may be further improved by segmenting the records into "compartments." Different encryption/decryption keys may be used to protect different "compartments." This strategy can be used to limit the amount of information within secure database 610 that is encrypted with a single key. Another technique for increasing security of secure database 610 may be to encrypt different portions of the same records with different keys so that more than one key may be needed to decrypt those records.

Backup of Secure Database 610

Secure database 610 in the preferred embodiment is backed up at periodic or other time intervals to protect the information the secure database contains. This secure database information may be of substantial value to many VDE participants. Back ups of secure database 610 should occur without significant inconvenience to the user, and should not breach any security.

The need to back up secure database 610 may be checked at power on of electronic appliance 600, when SPE 503 is initially invoked, at periodic time intervals, and if "audit roll up" value or other summary services information maintained by SPE 503 exceeds a user set or other threshold, or triggered by criteria established by one or more content publishers and/or distributors and/or clearinghouse service providers and/or users. The user may be prompted to backup if she has failed to do so by or at some certain point in time or after a certain duration of time or quantity of usage, or the backup may proceed automatically without user intervention.

Referring to Figure 8, backup storage 668 and storage media 670 (e.g., magnetic tape) may be used to store backed up information. Of course, any non-volatile media (e.g., one or more

floppy diskettes, a writable optical diskette, a hard drive, or the like) may be used for backup storage 668.

There are at least two scenarios to backing up secure database 610. The first scenario is "site specific," and uses the security of SPU 500 to support restoration of the backed up information. This first method is used in case of damage to secure database 610 due for example to failure of secondary storage device 652, inadvertent user damage to the files, or other occurrences that may damage or corrupt some or all of secure database 610. This first, site specific scenario of back up assumes that an SPU 500 still functions properly and is available to restore backed up information.

The second back up scenario assumes that the user's SPU 500 is no longer operational and needs to be, or has been, replaced. This second approach permits an authorized VDE administrator or other authorized VDE participant to access the stored back up information in order to prevent loss of critical data and/or assist the user in recovering from the error.

Both of these scenarios are provided by the example of program control steps performed by ROS 602 shown in Figure 39. Figure 39 shows an example back up routine 1250 performed by an electronic appliance 600 to back up secure database 610 (and

other information) onto back up storage 668. Once a back up has been initiated, as discussed above, back up routine 1250 generates one or more back up keys (block 1252). Back up routine 1250 then reads all secure database items, decrypts each item using the original key used to encrypt them before they were stored in secure database 610 (block 1254). Since SPU 500 is typically the only place where the keys for decrypting this information within an instance of secure database 610 are stored, and since one of the scenarios provided by back up routine 1250 is that SPU 500 completely failed or is destroyed, back up routine 1250 performs this reading and decrypting step 1254 so that recovery from a backup is not dependent on knowledge of these keys within the SPU. Instead, back up routine 1250 encrypts each secure database 610 item with a newly generated back up key(s) (block 1256) and writes the encrypted item to back up store 668 (block 1258). This process continues until all items within secure database 610 have been read, decrypted, encrypted with a newly generated back up key(s), and written to the back up store (as tested for by decision block 1260).

The preferred embodiment also reads the summary services audit information stored within the protected memory of SPU 500 by SPE summary services manager 560, encrypts this information with the newly generated back up key(s), and writes

this summary services information to back up store 668 (block 1262).

Finally, back up routine 1250 saves the back up key(s) generated by block 1252 and used to encrypt in blocks 1256, 1262 onto back up store 668. It does this in two secure ways in order to cover both of the restoration scenarios discussed above. Back up routine 1250 may encrypt the back up key(s) (along with other information such as the time of back up and other appropriate information to identify the back up) with a further key or keys such that only SPU 500 can decrypt (block 1264). This encrypted information is then written to back up store 668 (block 1264). For example, this step may include multiple encryptions using one or more public keys with corresponding private keys known only to SPU 500. Alternatively, a second back up key generated by the SPU 500 and kept only in the SPU may be used for the final encryption in place of a public key. Block 1264 preferably includes multiple encryption in order to make it more difficult to attack the security of the back up by "cracking" the encryption used to protect the back up keys. Although block 1262 includes encrypted summary services information on the back up, it preferably does not include SPU device private keys, shared keys, SPU code and other internal security information to prevent this information from ever becoming available to users even in encrypted form.

The information stored by block 1264 is sufficient to allow the same SPU 500 that performed (or at least in part performed) back up routine 1250 to recover the backed up information. However, this information is useless to any device other than that same SPU because only that SPU knows the particular keys used to protect the back up keys. To cover the other possible scenario wherein the SPU 500 fails in a non-recoverable way, back up routine 1250 provides an additional step (block 1266) of saving the back up key(s) under protection of one or more further set of keys that may be read by an authorized VDE administrator. For example, block 1266 may encrypt the back up keys with an "download authorization key" received during initialization of SPU 500 from a VDE administrator. This encrypted version of back up keys is also written to back up store 668 (block 1266). It can be used to support restoration of the back up files in the event of an SPU 500 failure. More specifically, a VDE administrator that knows the download authorization (or other) keys(s) used by block 1266 may be able to recover the back up key(s) in the back up store 668 and proceed to restore the backed up secure database 610 to the same or different electronic appliance 600.

In the preferred embodiment, the information saved by routine 1250 in back up files can be restored only after receiving a back up authorization from an authorized VDE administrator.

In most cases, the restoration process will simply be a restoration of secure database 610 with some adjustments to account for any usage since the back up occurred. This may require the user to contact additional providers to transmit audit and billing data and receive new budgets to reflect activity since the last back up. Current summary services information maintained within SPU 500 may be compared to the summary services information stored on the back up to determine or estimate most recent usage activity.

In case of an SPU 500 failure, an authorized VDE administrator must be contacted to both initialize the replacement SPU 500 and to decrypt the back up files. These processes allow for both SPU failures and upgrades to new SPUs. In the case of restoration, the back up files are used to restore the necessary information to the user's system. In the case of upgrades, the back up files may be used to validate the upgrade process.

The back up files may in some instances be used to transfer management information between electronic appliances 600. However, the preferred embodiment may restrict some or all information from being transportable between electronic appliances with appropriate authorizations. Some or all of the

back up files may be packaged within an administrative object and transmitted for analysis, transportation, or other uses.

As a more detailed example of a need for restoration from back up files, suppose an electronic appliance 600 suffers a hard disk failure or other accident that wipes out or corrupts part or all of the secure database 610, but assume that the SPU 500 is still functional. SPU 500 may include all of the information (e.g., secret keys and the like) it needs to restore the secure database 610. However, ROS 602 may prevent secure database restoration until a restoration authorization is received from a VDE administrator. A restoration authorization may comprise, for example, a "secret value" that must match a value expected by SPE 503. A VDE administrator may, if desired, only provide this restoration authorization after, for example, summary services information stored within SPU 500 is transmitted to the administrator in an administrative object for analysis. In some circumstances, a VDE administrator may require that a copy (partial or complete) of the back up files be transmitted to it within an administrative object to check for indications of fraudulent activities by the user. The restoration process, once authorized, may require adjustment of restored budget records and the like to reflect activity since the last back up, as mentioned above.

Figure 40 is an example of program controlled "restore" routine 1268 performed by electronic appliance 600 to restore secure database 610 based on the back up provided by the routine shown in Figure 38. This restore may be used, for example, in the event that an electronic appliance 600 has failed but can be recovered or "reinitialized" through contact with a VDE administrator for example. Since the preferred embodiment does not permit an SPU 500 to restore from backup unless and until authorized by a VDE administrator, restore routine 1268 begins by establishing a secure communication with a VDE administrator that can authorize the restore to occur (block 1270). Once SPU 500 and the VDE administrator authenticate one another (part of block 1270), the VDE administrator may extract "work in progress" and summary values from the SPU 500's internal non-volatile memory (block 1272). The VDE administrator may use this extracted information to help determine, for example, whether there has been a security violation, and also permits a failed SPU 500 to effectively "dump" its contents to the VDE administrator to permit the VDE administrator to handle the contents. The SPU 500 may encrypt this information and provide it to the VDE administrator packaged in one or more administrative objects. The VDE administrator may then request a copy of some or all of the current backup of secure database 610 from the SPU 500 (block 1274). This information may be packaged by SPU 500 into one or

more administrative objects, for example, and sent to the VDE administrator. Upon receiving the information, the VDE administrator may read the summary services audit information from the backup volume (i.e., information stored by Figure 38 block 1262) to determine the summary values and other information stored at time of backup. The VDE administrator may also determine the time and date the backup was made by reading the information stored by Figure 38 block 1264.

The VDE administrator may at this point restore the summary values and other information within SPU 500 based on the information obtained by block 1272 and from the backup (block 1276). For example, the VDE administrator may reset SPU internal summary values and counters so that they are consistent with the last backup. These values may be adjusted by the VDE administrator based on the "work in progress" recovered by block 1272, the amount of time that has passed since the backup, etc. The goal may typically be to attempt to provide internal SPU values that are equal to what they would have been had the failure not occurred.

The VDE administrator may then authorize SPU 500 to recover its secure database 610 from the backup files (block 1278). This restoration process replaces all secure database 610 records with the records from the backup. The VDE

administrator may adjust these records as needed by passing commands to SPU 500 during or after the restoration process.

The VDE administrator may then compute bills based on the recovered values (block 1280), and perform other actions to recover from SPU downtime (block 1282). Typically, the goal is to bill the user and adjust other VDE 100 values pertaining to the failed electronic appliance 600 for usage that occurred subsequent to the last backup but prior to the failure. This process may involve the VDE administrator obtaining, from other VDE participants, reports and other information pertaining to usage by the electronic appliance prior to its failure and comparing it to the secure database backup to determine which usage and other events are not yet accounted for.

In one alternate embodiment, SPU 500 may have sufficient internal, non-volatile memory to allow it to store some or all of secure database 610. In this embodiment, the additional memory may be provided by additional one or more integrated circuits that can be contained within a secure enclosure, such as a tamper resistant metal container or some form of a chip pack containing multiple integrated circuit components, and which impedes and/or evidences tampering attempts, and/or disables a portion or all of SPU 500 or associated critical key and/or other control information in the event of tampering. The same back up

routine 1250 shown in Figure 38 may be used to back up this type of information, the only difference being that block 1254 may read the secure database item from the SPU internal memory and may not need to decrypt it before encrypting it with the back up key(s).

Event-Driven VDE Processes

As discussed above, processes provided by/under the preferred embodiment rights operating system (ROS) 602 may be "event driven." This "event driven" capability facilitates integration and extendibility.

An "event" is a happening at a point in time. Some examples of "events" are a user striking a key of a keyboard, arrival of a message or an object 300, expiration of a timer, or a request from another process.

In the preferred embodiment, ROS 602 responds to an "event" by performing a process in response to the event. ROS 602 dynamically creates active processes and tasks in response to the occurrence of an event. For example, ROS 602 may create and begin executing one or more component assemblies 690 for performing a process or processes in response to occurrence of an event. The active processes and tasks may terminate once ROS 602 has responded to the event. This ability to dynamically

create (and end) tasks in response to events provides great flexibility, and also permits limited execution resources such as those provided by an SPU 500 to perform a virtually unlimited variety of different processes in different contexts.

Since an "event" may be any type of happening, there are an unlimited number of different events. Thus, any attempt to categorize events into different types will necessarily be a generalization. Keeping this in mind, it is possible to categorize events provided/supported by the preferred embodiment into two broad categories:

- user-initiated events; and
- system-initiated events.

Generally, "user-initiated" events are happenings attributable to a user (or a user application). A common "user-initiated" event is a user's request (e.g., by pushing a keyboard button, or transparently using redirector 684) to access an object 300 or other VDE-protected information.

"System-initiated" events are generally happenings not attributable to a user. Examples of system initiated events include the expiration of a timer indicating that information should be backed to non-volatile memory, receipt of a message

from another electronic appliance 600, and a service call generated by another process (which may have been started to respond to a system-initiated event and/or a user-initiated event).

ROS 602 provided by the preferred embodiment responds to an event by specifying and beginning processes to process the event. These processes are, in the preferred embodiment, based on methods 1000. Since there are an unlimited number of different types of events, the preferred embodiment supports an unlimited number of different processes to process events. This flexibility is supported by the dynamic creation of component assemblies 690 from independently deliverable modules such as method cores 1000', load modules 1100, and data structures such as UDEs 1200. Even though any categorization of the unlimited potential types of processes supported/provided by the preferred embodiment will be a generalization, it is possible to generally classify processes as falling within two categories:

- processes relating to use of VDE protected information;
- and
- processes relating to VDE administration.

"Use" and "Administrative" Processes

"Use" processes relate in some way to use of VDE-protected information. Methods 1000 provided by the preferred

embodiment may provide processes for creating and maintaining a chain of control for use of VDE-protected information. One specific example of a "use" type process is processing to permit a user to open a VDE object 300 and access its contents. A method 1000 may provide detailed use-related processes such as, for example, releasing content to the user as requested (if permitted), and updating meters, budgets, audit trails, etc. Use-related processes are often user-initiated, but some use processes may be system-initiated. Events that trigger a VDE use-related process may be called "use events."

An "administrative" process helps to keep VDE 100 working. It provides processing that helps support the transaction management "infrastructure" that keeps VDE 100 running securely and efficiently. Administrative processes may, for example, provide processing relating to some aspect of creating, modifying and/or destroying VDE-protected data structures that establish and maintain VDE's chain of handling and control. For example, "administrative" processes may store, update, modify or destroy information contained within a VDE electronic appliance 600 secure database 610. Administrative processes also may provide communications services that establish, maintain and support secure communications between different VDE electronic appliances 600. Events that trigger administrative processes may be called "administrative events."

Reciprocal Methods

Some VDE processes are paired based on the way they interact together. One VDE process may "request" processing services from another VDE process. The process that requests processing services may be called a "request process." The "request" constitutes an "event" because it triggers processing by the other VDE process in the pair. The VDE process that responds to the "request event" may be called a "response process." The "request process" and "response process" may be called "reciprocal processes."

The "request event" may comprise, for example, a message issued by one VDE node electronic appliance 600 or process for certain information. A corresponding "response process" may respond to the "request event" by, for example, sending the information requested in the message. This response may itself constitute a "request event" if it triggers a further VDE "response process." For example, receipt of a message in response to an earlier-generated request may trigger a "reply process." This "reply process" is a special type of "response process" that is triggered in response to a "reply" from another "response process." There may be any number of "request" and "response" process pairs within a given VDE transaction.

A "request process" and its paired "response process" may be performed on the same VDE electronic appliance 600, or the two processes may be performed on different VDE electronic appliances. Communication between the two processes in the pair may be by way of a secure (VDE-protected) communication, an "out of channel" communication, or a combination of the two.

Figures 41a-41d are a set of examples that show how the chain of handling and control is enabled using "reciprocal methods." A chain of handling and control is constructed, in part, using one or more pairs of "reciprocal events" that cooperate in request-response manner. Pairs of reciprocal events may be managed in the preferred embodiment in one or more "reciprocal methods." As mentioned above, a "reciprocal method" is a method 1000 that can respond to one or more "reciprocal events." Reciprocal methods contain the two halves of a cooperative process that may be securely executed at physically and/or temporally distant VDE nodes. The reciprocal processes may have a flexibly defined information passing protocols and information content structure. The reciprocal methods may, in fact, be based on the same or different method core 1000' operating in the same or different VDE nodes 600. VDE nodes 600A and 600B shown in Figure 41a may be the same physical electronic appliance 600 or may be separate electronic appliances.

Figure 41a is an example of the operation of a single pair of reciprocal events. In VDE node 600A, method 1000a is processing an event that has a request that needs to be processed at VDE node 600B. The method 1000a (e.g., based on a component assembly 690 including its associated load modules 1100 and data) that responds to this "request" event is shown in Figure 41a as 1450. The process 1450 creates a request (1452) and, optionally, some information or data that will be sent to the other VDE node 1000b for processing by a process associated with the reciprocal event. The request and other information may be transmitted by any of the transport mechanisms described elsewhere in this disclosure.

Receipt of the request by VDE node 600b comprises a response event at that node. Upon receipt of the request, the VDE node 600b may perform a "reciprocal" process 1454 defined by the same or different method 1000b to respond to the response event. The reciprocal process 1454 may be based on a component assembly 690 (e.g., one or more load modules 1100, data, and optionally other methods present in the VDE node 600B).

Figure 41b extends the concepts presented in Figure 41a to include a response from VDE node 600B back to VDE node 600A. The process starts as described for Figure 41a through the receipt and processing of the request event and information 1452

by the response process 1454 in VDE node 600B. The response process 1454 may, as part of its processing, cooperate with another request process (1468) to send a response 1469 back to the initiating VDE node 600A. A corresponding reciprocal process 1470 provided by method 1000A may respond to and process this request event 1469. In this manner, two or more VDE nodes 600A, 600B may cooperate and pass configurable information and requests between methods 1000A, 1000B executing in the nodes. The first and second request-response sequences [(1450, 1452, 1454) and (1468, 1469, 1470)] may be separated by temporal and spatial distances. For efficiency, the request (1468) and response (1454) processes may be based on the same method 1000 or they may be implemented as two methods in the same or different method core 1000'. A method 1000 may be parameterized by an "event code" so it may provide different behaviors/results for different events, or different methods may be provided for different events.

Figure 41c shows the extension the control mechanism described in Figures 41a-41b to three nodes (600A, 600B, 600C). Each request-response pair operates in the manner as described for Figure 41b, with several pairs linked together to form a chain of control and handling between several VDE nodes 600A, 600B, 600C. This mechanism may be used to extend the chain of handling and control to an arbitrary number of VDE nodes using

any configuration of nodes. For example, VDE node 600C might communicate directly to VDE node 600A and communicate directly to VDE 600B, which in turn communicates with VDE node 600A. Alternately, VDE node 600C might communicate directly with VDE node 600A, VDE node 600A may communicate with VDE node 600B, and VDE node 600B may communicate with VDE node 600C.

A method 1000 may be parameterized with sets of events that specify related or cooperative functions. Events may be logically grouped by function (e.g., use, distribute), or a set of reciprocal events that specify processes that may operate in conjunction with each other. Figure 41d illustrates a set of "reciprocal events" that support cooperative processing between several VDE nodes 102, 106, 112 in a content distribution model to support the distribution of budget. The chain of handling and control, in this example, is enabled by using a set of "reciprocal events" specified within a BUDGET method. Figure 41d is an example of how the reciprocal event behavior within an example BUDGET method (1510) work in cooperation to establish a chain of handling and control between several VDE nodes. The example BUDGET method 1510 responds to a "use" event 1478 by performing a "use" process 1476 that defines the mechanism by which processes are budgeted. The BUDGET method 1510 might, for example, specify a use process 1476 that compares a

meter count to a budget value and fail the operation if the meter count exceeds the budget value. It might also write an audit trail that describes the results of said BUDGET decisions. Budget method 1510 may respond to a "distribute" event by performing a distribute process 1472 that defines the process and/or control information for further distribution of the budget. It may respond to a "request" event 1480 by performing a request process 1480 that specifies how the user might request use and/or distribution rights from a distributor. It may respond to a "response" event 1482 by performing a response process 1484 that specifies the manner in which a distributor would respond to requests from other users to whom they have distributed some (or all) of their budget to. It may respond to a "reply" event 1474 by performing a reply process 1475 that might specify how the user should respond to message regranting or denying (more) budget.

Control of event processing, reciprocal events, and their associated methods and method components is provided by PERCs 808 in the preferred embodiment. These PERCs (808) might reference administrative methods that govern the creation, modification, and distribution of the data structures and administrative methods that permit access, modification, and further distribution of these items. In this way, each link in the chain of handling and control might, for example, be able to

customize audit information, alter the budget requirements for using the content, and/or control further distribution of these rights in a manner specified by prior members along the distribution chain.

In the example shown in Figure 41d, a distributor at a VDE distributor node (106) might request budget from a content creator at another node (102). This request may be made in the context of a secure VDE communication or it may be passed in an "out-of-channel" communication (e.g. a telephone call or letter). The creator 102 may decide to grant budget to the distributor 106 and processes a distribute event (1452 in BUDGET method 1510 at VDE node 102). A result of processing the distribute event within the BUDGET method might be a secure communication (1454) between VDE nodes 102 and 106 by which a budget granting use and redistribute rights to the distributor 106 may be transferred from the creator 102 to the distributor. The distributor's VDE node 106 may respond to the receipt of the budget information by processing the communication using the reply process 1475B of the BUDGET method 1510. The reply event processing 1475B might, for example, install a budget and PERC 808 within the distributor's VDE 106 node to permit the distributor to access content or processes for which access is control at least in part by the budget and/or PERC. At some

point, the distributor 106 may also desire to use the content to which she has been granted rights to access.

After registering to use the content object, the user 112 would be required to utilize an array of "use" processes 1476C to, for example, open, read, write, and/or close the content object as part of the use process.

Once the distributor 106 has used some or all of her budget, she may desire to obtain additional budget. The distributor 106 might then initiate a process using the BUDGET method request process (1480B). Request process 1480B might initiate a communication (1482AB) with the content creator VDE node 102 requesting more budget and perhaps providing details of the use activity to date (e.g., audit trails). The content creator 102 processes the 'get more budget' request event 1482AB using the response process (1484A) within the creator's BUDGET method 1510A. Response process 1484A might, for example, make a determination if the use information indicates proper use of the content, and/or if the distributor is credit worthy for more budget. The BUDGET method response process 1484A might also initiate a financial transaction to transfer funds from the distributor to pay for said use, or use the distribute process 1472A to distribute budget to the distributor 106. A response to the distributor 106 granting more budget (or denying more

budget) might be sent immediately as a response to the request communication 1482AB, or it might be sent at a later time as part of a separate communication. The response communication, upon being received at the distributor's VDE node 106, might be processed using the reply process 1475B within the distributor's copy of the BUDGET method 1510B. The reply process 1475B might then process the additional budget in the same manner as described above.

The chain of handling and control may, in addition to posting budget information, also pass control information that governs the manner in which said budget may be utilized. For example, the control information specified in the above example may also contain control information describing the process and limits that apply to the distributor's redistribution of the right to use the creator's content object. Thus, when the distributor responds to a budget request from a user (a communication between a user at VDE node 112 to the distributor at VDE node 106 similar in nature to the one described above between VDE nodes 106 and 102) using the distribute process 1472B within the distributor's copy of the BUDGET method 1510B, a distribution and request/response/reply process similar to the one described above might be initiated.

Thus, in this example a single method can provide multiple dynamic behaviors based on different "triggering" events. For example, single BUDGET method 1510 might support any or all of the events listed below:

Event Type	Event	Process Description
"Use" Events	use budget	Use budget.
Request Events Processed by User Node Request Process 1480c	request more budget	Request more money for budget.
	request audit by auditor #1	Request that auditor #1 audit the budget use.
	request budget deletion	Request that budget be deleted from system.
	request method updated	Update method used for auditing.
	request to change auditors	Change from auditor 1 to auditor 2, or vice versa.
	request different audit interval	Change time interval between audits.
	request ability to provide budget copies	Request ability to provide copies of a budget.
	request ability to distribute budget	Request ability to distribute a budget to other users.
	request account status	Request information on current status of an account.
	Request New Method	Request new method.
	Request Method Update	Request update of method.
	Request Method Deletion	Request deletion of method.
	Response Events Processed by User Node Request Process 1480C	receive more budget
receive method update		Update method.
receive auditor change		Change from one auditor to another.
receive change to audit interval		Change interval between audits.

Event Type	Event	Process Description
	receive budget deletion	Delete budget.
	provide audit to auditor #1	Forward audit information to auditor #1.
	provide audit to auditor #2	Forward audit information to auditor #2.
	receive account status	Provide account status.
	Receive New	Receive new budget.
	Receive Method Update	Receive updated information.
	Receive More	Receive more for budget.
	Sent Audit	Send audit information.
	Perform Deletion	Delete information.
	"Distribute" Events	Create New
Provide More		Provide more for budget.
Audit		Perform audit.
Delete		Delete information.
Reconcile		Reconcile budget and auditing.
Copy		Copy budget.
Distribute		Distribute budget.
Method Modification		Modify method.
Display Method		Display requested method.
"Request" Events Processed by Distributor Node Request Process 1484B	Delete	Delete information.
	Get New	Get new budget.
	Get More	Get more for budget.
	Get Updated	Get updated information.
	Get Audited	Get audit information.
"Response Events" Processed by Distributor Node Request Process 1484B	Provide New to user	Provide new budget to user.
	Provide More to user	Provide more budget to user.
	Provide Update to user	Provided updated budget to user.

Event Type	Event	Process Description
	Audit user	Audit a specified user.
	Delete user's method	Delete method belonging to user.

Examples of Reciprocal Method Processes

A. BUDGET

Figures 42a, 42b, 42c and 42d, respectively, are flowcharts of example process control steps performed by a representative example of BUDGET method 2250 provided by the preferred embodiment. In the preferred embodiment, BUDGET method 2250 may operate in any of four different modes:

- use (see Figure 42a)
- administrative request (see Figure 42b)
- administrative response (see Figure 42c)
- administrative reply (see Figure 42d).

In general, the "use" mode of BUDGET method 2250 is invoked in response to an event relating to the use of an object or its content. The "administrative request" mode of BUDGET method 2250 is invoked by or on behalf of the user in response to some user action that requires contact with a VDE financial provider, and basically its task is to send an administrative request to the VDE financial provider. The "administrative response" mode of BUDGET method 2250 is performed at the VDE financial provider in response to receipt of an administrative request sent from a VDE node to the VDE financial provider by the

"administrative request" invocation of BUDGET method 2250 shown in Figure 42b. The "administrative response" invocation of BUDGET method 2250 results in the transmission of an administrative object from VDE financial provider to the VDE user node. Finally, the "administrative reply" invocation of BUDGET method 2250 shown in Figure 42d is performed at the user VDE node upon receipt of the administrative object sent by the "administrative response" invocation of the method shown in Figure 42c.

In the preferred embodiment, the same BUDGET method 2250 performs each of the four different step sequences shown in Figures 42a-42d. In the preferred embodiment, different event codes may be passed to the BUDGET method 2250 to invoke these various different modes. Of course, it would be possible to use four separate BUDGET methods instead of a single BUDGET method with four different "dynamic personalities," but the preferred embodiment obtains certain advantages by using the same BUDGET method for each of these four types of invocations.

Looking at Figure 42a, the "use" invocation of BUDGET method 2250 first primes the Budget Audit Trail (blocks 2252, 2254). It then obtains the DTD for the Budget UDE, which it uses to obtain and read the Budget UDE blocks 2256-2262).

BUDGET method 2250 in this "use" invocation may then determine whether a Budget Audit date has expired, and terminate if it has ("yes" exit to decision block 2264; blocks 2266, 2268). So long as the Budget Audit date has not expired, the method may then update the Budget using the atomic element and event counts (and possibly other information) (blocks 2270, 2272), and may then save a Budget User Audit record in a Budget Audit Trail UDE (blocks 2274, 2276) before terminating (at terminate point 2278).

Looking at Figure 42b, the first six steps (blocks 2280-2290) may be performed by the user VDE node in response to some user action (e.g., request to access new information, request for a new budget, etc.). This "administrative request" invocation of BUDGET method 2250 may prime an audit trail (blocks 2280, 2282). The method may then place a request for administrative processing of an appropriate Budget onto a request queue (blocks 2284, 2286). Finally, the method may save appropriate audit trail information (blocks 2288, 2290). Sometime later, the user VDE node may prime a communications audit trail (blocks 2292, 2294), and may then write a Budget Administrative Request into an administrative object (block 2296). This step may obtain information from the secure database as needed from such sources such as, for example, Budget UDE; Budget Audit Trail

UDE(s); and Budget Administrative Request Record(s) (block 2298).

Block 2296 may then communicate the administrative object to a VDE financial provider, or alternatively, block 2296 may pass administrative object to a separate communications process or method that arranges for such communications to occur. If desired, method 2250 may then save a communications audit trail (blocks 2300, 2302) before terminating (at termination point 2304).

Figure 42c is a flowchart of an example of process control steps performed by the example of BUDGET method 2250 provided by the preferred embodiment operating in an "administrative response" mode. Steps shown in Figure 42c would, for example, be performed by a VDE financial provider who has received an administrative object containing a Budget administrative request as created (and communicated to a VDE administrator for example) by Figure 42b (block 2296).

Upon receiving the administrative object, BUDGET method 2250 at the VDE financial provider site may prime a budget communications and response audit trail (blocks 2306, 2308), and may then unpack the administrative object and retrieve the budget request(s), audit trail(s) and record(s) it

contains (block 2310). This information retrieved from the administrative object may be written by the VDE financial provider into its secure database (block 2312). The VDE financial provider may then retrieve the budget request(s) and determine the response method it needs to execute to process the request (blocks 2314, 2316). BUDGET method 2250 may send the event(s) contained in the request record(s) to the appropriate response method and may generate response records and response requests based on the RESPONSE method (block 2318). The process performed by block 2318 may satisfy the budget request by writing appropriate new response records into the VDE financial provider's secure database (block 2320). BUDGET method 2250 may then write these Budget administrative response records into an administrative object (blocks 2322, 2324), which it may then communicate back to the user node that initiated the budget request. BUDGET method 2250 may then save communications and response processing audit trail information into appropriate audit trail UDE(s) (blocks 2326, 2328) before terminating (at termination point 2330).

Figure 42d is a flowchart of an example of program control steps performed by a representative example of BUDGET method 2250 operating in an "administrative reply" mode. Steps shown in Figure 42d might be performed, for example, by a VDE user node upon receipt of an administrative object containing budget-

related information. BUDGET method 2250 may first prime a Budget administrative and communications audit trail (blocks 2332, 2334). BUDGET method 2250 may then extract records and requests from a received administrative object and write the reply record to the VDE secure database (blocks 2336, 2338). The VDE user node may then save budget administrative and communications audit trail information in an appropriate audit trail UDE(s) (blocks 2340, 2341).

Sometime later, the VDE user node may retrieve the reply record from the secure database and determine what method is required to process it (blocks 2344, 2346). The VDE user node may, optionally, prime an audit trail (blocks 2342, 2343) to record the results of the processing of the reply event. The BUDGET method 2250 may then send event(s) contained in the reply record(s) to the REPLY method, and may generate/update the secure database records as necessary to, for example, insert new budget records, delete old budget records and/or apply changes to budget records (blocks 2348, 2350). BUDGET method 2250 may then delete the reply record from the secure data base (blocks 2352, 2353) before writing the audit trail (if required) (blocks 2354m 2355) terminating (at terminate point 2356).

B. REGISTER

Figures 43a-43d are flowcharts of an example of program control steps performed by a representative example of a REGISTER method 2400 provided by the preferred embodiment. In this example, the REGISTER method 2400 performs the example steps shown in Figure 43a when operating in a "use" mode, performs the example steps shown in Figure 43b when operating in an "administrative request" mode, performs the steps shown in Figure 43c when operating in an "administrative response" mode, and performs the steps shown in Figure 43d when operating in an "administrative reply" mode.

The steps shown in Figure 43a may be, for example, performed at a user VDE node in response to some action by or on behalf of the user. For example the user may ask to access an object that has not yet been (or is not now) properly registered to her. In response to such a user request, the REGISTER method 2400 may prime a Register Audit Trail UDE (blocks 2402, 2404) before determining whether the object being requested has already been registered (decision block 2406). If the object has already been registered ("yes" exit to decision block 2406), the REGISTER method may terminate (at termination point 2408). If the object is not already registered ("no" exit to decision block 2406), then REGISTER method 2400 may access the VDE node

secure database PERC 808 and/or Register MDE (block 2410). REGISTER method 2400 may extract an appropriate Register Record Set from this PERC 808 and/or Register MDE (block 2412), and determine whether all of the required elements are present that are needed to register the object (decision block 2414). If some piece(s) is missing ("no" exit to decision block 2414), REGISTER method 2400 may queue a Register request record to a communication manager and then suspend the REGISTER method until the queued request is satisfied (blocks 2416, 2418). Block 2416 may have the effect of communicating a register request to a VDE distributor, for example. When the request is satisfied and the register request record has been received (block 2420), then the test of decision block 2414 is satisfied ("yes" exit to decision block 2414), and REGISTER method 2400 may proceed. At this stage, the REGISTER method 2400 may allow the user to select Register options from the set of method options allowed by PERC 808 accessed at block 2410 (block 2422). As one simple example, the PERC 808 may permit the user to pay by VISA or MasterCard but not by American Express; block 2422 may display a prompt asking the user to select between paying using her VISA card and paying using her MasterCard (block 2424). The REGISTER method 2400 preferably validates the user selected registration options and requires the user to select different options if the initial user options were invalid (block 2426, "no" exit to decision block 2428).

Once the user has made all required registration option selections and those selections have been validated ("yes" exit to decision block 2428), the REGISTER method 2400 may write an User Registration Table (URT) corresponding to this object and this user which embodies the user registration selections made by the user along with other registration information required by PERC 808 and/or the Register MDE (blocks 2430, 2432). REGISTER method 2400 may then write a Register audit record into the secure database (blocks 2432, 2434) before terminating (at terminate point 2436).

Figure 43b shows an example of an "administrative request" mode of REGISTER method 2400. This Administrative Request Mode may occur on a VDE user system to generate an appropriate administrative object for communication to a VDE distributor or other appropriate VDE participant requesting registration information. Thus, for example, the steps shown in Figure 43b may be performed as part of the "queue register request record" block 2416 shown in Figure 43a. To make a Register administrative request, REGISTER method 2400 may first prime a communications audit trail (blocks 2440, 2442), and then access the secure database to obtain data about registration (block 2444). This secure database access may, for example, allow the owner and/or publisher of the object being registered to find out demographic, user or other information about the user.

As a specific example, suppose that the object being registered is a spreadsheet software program. The distributor of the object may want to know what other software the user has registered. For example, the distributor may be willing to give preferential pricing if the user registers a "suite" of multiple software products distributed by the same distributor. Thus, the sort of information solicited by a "user registration" card enclosed with most standard software packages may be solicited and automatically obtained by the preferred embodiment at registration time. In order to protect the privacy rights of the user, REGISTER method 2400 may pass such user-specific data through a privacy filter that may be at least in part customized by the user so the user can prevent certain information from being revealed to the outside world (block 2446). The REGISTER method 2400 may write the resulting information along with appropriate Register Request information identifying the object and other appropriate parameters into an administrative object (blocks 2448, 2450). REGISTER method 2400 may then pass this administrative object to a communications handler. REGISTER method 2400 may then save a communications audit trail (blocks 2452, 2454) before terminating (at terminate point 2456).

Figure 43c includes REGISTER method 2400 steps that may be performed by a VDE distributor node upon receipt of Register Administrative object sent by block 2448, Figure 43b.

REGISTER method 2400 in this "administrative response" mode may prime appropriate audit trails (blocks 2460, 2462), and then may unpack the received administrative object and write the associated register request(s) configuration information into the secure database (blocks 2464, 2466). REGISTER method 2400 may then retrieve the administrative request from the secure database and determine which response method to run to process the request (blocks 2468, 2470). If the user fails to provide sufficient information to register the object, REGISTER method 2400 may fail (blocks 2472, 2474). Otherwise, REGISTER method 2400 may send event(s) contained in the appropriate request record(s) to the appropriate response method, and generate and write response records and response requests (e.g., PERC(s) and/or UDEs) to the secure database (blocks 2476, 2478). REGISTER method 2400 may then write the appropriate Register administrative response record into an administrative object (blocks 2480, 2482). Such information may include, for example, one or more replacement PERC(s) 808, methods, UDE(s), etc. (block 2482). This enables, for example, a distributor to distribute limited right permissions giving users only enough information to register an object, and then later, upon registration, replacing the limited right permissions with wider permissioning scope granting the user more complete access to the objects. REGISTER method 2400 may then save

the communications and response processing audit trail (blocks 2484, 2486), before terminating (at terminate point 2488).

Figure 43d shows steps that may be performed by the VDE user node upon receipt of the administrative object generated/transmitted by Figure 43c block 2480. The steps shown in Figure 43d are very similar to those shown in Figure 42d for the BUDGET method administrative reply process.

C. AUDIT

Figures 44a-44c are flowcharts of examples of program control steps performed by a representative example of an AUDIT method 2520 provided by the preferred embodiment. As in the examples above, the AUDIT method 2520 provides three different operational modes in this preferred embodiment example: Figure 44a shows the steps performed by the AUDIT method in an "administrative request" mode; Figure 44b shows steps performed by the method in the "administrative response" mode; and Figure 44c shows the steps performed by the method in an "administrative reply" mode.

The AUDIT method 2520 operating in the "administrative request" mode as shown in Figure 44a is typically performed, for example, at a VDE user node based upon some request by or on behalf of the user. For example, the user may have requested an

audit, or a timer may have expired that initiates communication of audit information to a VDE content provider or other VDE participant. In the preferred embodiment, different audits of the same overall process may be performed by different VDE participants. A particular "audit" method 2520 invocation may be initiated for any one (or all) of the involved VDE participants. Upon invocation of AUDIT method 2520, the method may prime an audit administrative audit trail (thus, in the preferred embodiment, the audit processing may itself be audited) (blocks 2522, 2524). The AUDIT method 2520 may then queue a request for administrative processing (blocks 2526, 2528), and then may save the audit administrative audit trail in the secure database (blocks 2530, 2532). Sometime later, AUDIT method 2520 may prime a communications audit trail (blocks 2534, 2536), and may then write Audit Administrative Request(s) into one or more administrative object(s) based on specific UDE, audit trail UDE(s), and/or administrative record(s) stored in the secure database (blocks 2538, 2540). The AUDIT method 2520 may then save appropriate information into the communications audit trail (blocks 2542, 2544) before terminating (at terminate point 2546).

Figure 44b shows example steps performed by a VDE content provider, financial provider or other auditing VDE node upon receipt of the administrative object generated and

communicated by Figure 44a block 2538. The AUDIT method 2520 in this "administrative response" mode may first prime an Audit communications and response audit trail (blocks 2550, 2552), and may then unpack the received administrative object and retrieve its contained Audit request(s) audit trail(s) and audit record(s) for storage into the secured database (blocks 2554, 2556). AUDIT method 2520 may then retrieve the audit request(s) from the secure database and determine the response method to run to process the request (blocks 2558, 2560). AUDIT method 2520 may at this stage send event(s) contained in the request record(s) to the appropriate response method, and generate response record(s) and requests based on this method (blocks 2562, 2564). The processing block 2562 may involve a communication to the outside world.

For example, AUDIT method 2520 at this point could call an external process to perform, for example, an electronic funds transfer against the user's bank account or some other bank account. The AUDIT administrative response can, if desired, call an external process that interfaces VDE to one or more existing computer systems. The external process could be passed the user's account number, PIN, dollar amount, or any other information configured in, or associated with, the VDE audit trail being processed. The external process can communicate with non-VDE hosts and use the information passed to it as part of

these communications. For example, the external process could generate automated clearinghouse (ACH) records in a file for submittal to a bank. This mechanism would provide the ability to automatically credit or debit a bank account in any financial institution. The same mechanism could be used to communicate with the existing credit card (e.g. VISA) network by submitting VDE based charges against the charge account.

Once the appropriate Audit response record(s) have been generated, AUDIT method 2520 may write an Audit administrative record(s) into an administrative object for communication back to the VDE user node that generated the Audit request (blocks 2566, 2568). The AUDIT method 2520 may then save communications and response processing audit information in appropriate audit trail(s) (blocks 2570, 2572) before terminating (at terminate point 2574).

Figure 44c shows an example of steps that may be performed by the AUDIT method 2520 back at the VDE user node upon receipt of the administrative object generated and sent by Figure 44b, block 2566. The steps 2580-2599 shown in Figure 44c are similar to the steps shown in Figure 43d for the REGISTER method 2400 in the "administrative reply" mode. Briefly, these steps involve receiving and extracting appropriate response records from the administrative object (block 2584), and

then processing the received information appropriately to update secure database records and perform any other necessary actions (blocks 2595, 2596).

Examples of Event-Driven Content-Based Methods

VDE methods 1000 are designed to provide a very flexible and highly modular approach to secure processing. A complete VDE process to service a "use event" may typically be constructed as a combination of methods 1000. As one example, the typical process for reading content or other information from an object 300 may involve the following methods:

- an EVENT method
- a METER method
- a BILLING method
- a BUDGET method.

Figure 45 is an example of a sequential series of methods performed by VDE 100 in response to an event. In this example, when an event occurs, an EVENT method 402 may "qualify" the event to determine whether it is significant or not. Not all events are significant. For example, if the EVENT method 1000 in a control process dictates that usage is to be metered based upon number of pages read, then user request "events" for reading less than a page of information may be ignored. In another example, if a system event represents a request to read a certain number

of bytes, and the EVENT method 1000 is part of a control process designed to meter paragraphs, then the EVENT method may evaluate the read request to determine how many paragraphs are represented in the bytes requested. This process may involve mapping to "atomic elements" to be discussed in more detail below.

EVENT method 402 filters out events that are not significant with regard to the specific control method involved. EVENT method 402 may pass on qualified events to a METER process 1404, which meters or discards the event based on its own particular criteria.

In addition, the preferred embodiment provides an optimization called "precheck." EVENT method/process 402 may perform this "precheck" based on metering, billing and budget information to determine whether processing based on an event will be allowed. Suppose, for example, that the user has already exceeded her budget with respect to accessing certain information content so that no further access is permitted. Although BUDGET method 408 could make this determination, records and processes performed by BUDGET method 404 and/or BILLING method 406 might have to be "undone" to, for example, prevent the user from being charged for an access that was actually denied. It may be more efficient to perform a "precheck"

within EVENT method 402 so that fewer transactions have to be "undone."

METER method 404 may store an audit record in a meter "trail" UDE 1200, for example, and may also record information related to the event in a meter UDE 1200. For example, METER method 404 may increment or decrement a "meter" value within a meter UDE 1200 each time content is accessed. The two different data structures (meter UDE and meter trail UDE) may be maintained to permit record keeping for reporting purposes to be maintained separately from record keeping for internal operation purposes, for example.

Once the event is metered by METER method 404, the metered event may be processed by a BILLING method 406. BILLING method 406 determines how much budget is consumed by the event, and keeps records that are useful for reconciliation of meters and budgets. Thus, for example, BILLING method 406 may read budget information from a budget UDE, record billing information in a billing UDE, and write one or more audit records in a billing trail UDE. While some billing trail information may duplicate meter and/or budget trail information, the billing trail information is useful, for example, to allow a content creator 102 to expect a payment of a certain size, and serve as a reconciliation check to reconcile meter trail information sent to

creator 102 with budget trail information sent to, for example, an independent budget provider.

BILLING method 406 may then pass the event on to a BUDGET method 408. BUDGET method 408 sets limits and records transactional information associated with those limits. For example, BUDGET method 408 may store budget information in a budget UDE, and may store an audit record in a budget trail UDE. BUDGET method 408 may result in a "budget remaining" field in a budget UDE being decremented by an amount specified by BILLING method 406.

The information content may be released, or other action taken, once the various methods 402, 404, 406, 408 have processed the event.

As mentioned above, PERCs 808 in the preferred embodiment may be provided with "control methods" that in effect "oversee" performance of the other required methods in a control process. Figure 46 shows how the required methods/processes 402, 404, 406, and 408 of Figure 45 can be organized and controlled by a control method 410. Control method 410 may call, dispatch events, or otherwise invoke the other methods 402, 404, 406, 408 and otherwise supervise the processing performed in response to an "event."

Control methods operate at the level of control sets 906 within PERCs 808. They provide structure, logic, and flow of control between disparate acquired methods 1000. This mechanism permits the content provider to create any desired chain of processing, and also allows the specific chain of processing to be modified (within permitted limits) by downstream redistributors. This control structure concept provides great flexibility.

Figure 47 shows an example of an "aggregate" method 412 which collects METER method 404, BUDGET method 406 and BILLING method 408 into an "aggregate" processing flow. Aggregate method 412 may, for example, combine various elements of metering, budgeting and billing into a single method 1000. Aggregate method 412 may provide increased efficiency as a result of processing METER method 404, BUDGET method 406 and BILLING method 408 aggregately, but may decrease flexibility because of decreased modularity.

Many different methods can be in effect simultaneously. Figure 48 shows an example of preferred embodiment event processing using multiple METER methods 404 and multiple BUDGET methods 1408. Some events may be subject to many different required methods operating independently or cumulatively. For example, in the example shown in Figure 48,

meter method 404a may maintain meter trail and meter information records that are independent from the meter trail and meter information records maintained by METER method 404b. Similarly, BUDGET method 408a may maintain records independently of those records maintained by BUDGET method 408b. Some events may bypass BILLING method 408 while nevertheless being processed by meter method 404a and BUDGET method 408a. A variety of different variations are possible.

REPRESENTATIVE EXAMPLES OF VDE METHODS

Although methods 1000 can have virtually unlimited variety and some may even be user-defined, certain basic "use" type methods are preferably used in the preferred embodiment to control most of the more fundamental object manipulation and other functions provided by VDE 100. For example, the following high level methods would typically be provided for object manipulation:

- OPEN method
- READ method
- WRITE method
- CLOSE method.

An OPEN method is used to control opening a container so its contents may be accessed. A READ method is used to control

the access to contents in a container. A WRITE method is used to control the insertion of contents into a container. A CLOSE method is used to close a container that has been opened.

Subsidiary methods are provided to perform some of the steps required by the OPEN, READ, WRITE and/or CLOSE methods. Such subsidiary methods may include the following:

- ACCESS method
- PANIC method
- ERROR method
- DECRYPT method
- ENCRYPT method
- DESTROY content method
- INFORMATION method
- OBSCURE method
- FINGERPRINT method
- EVENT method.
- CONTENT method
- EXTRACT method
- EMBED method
- METER method
- BUDGET method
- REGISTER method
- BILLING method
- AUDIT method

An ACCESS method may be used to physically access content associated with an opened container (the content can be anywhere). A PANIC method may be used to disable at least a portion of the VDE node if a security violation is detected. An ERROR method may be used to handle error conditions. A DECRYPT method is used to decrypt encrypted information. An ENCRYPT method is used to encrypt information. A DESTROY content method is used to destroy the ability to access specific content within a container. An INFORMATION method is used to provide public information about the contents of a container. An OBSCURE method is used to devalue content read from an opened container (e.g., to write the word "SAMPLE" over a displayed image). A FINGERPRINT method is used to mark content to show who has released it from the secure container. An event method is used to convert events into different events for response by other methods.

Open

Figure 49 is a flowchart of an example of preferred embodiment process control steps for an example of an OPEN method 1500. Different OPEN methods provide different detailed steps. However, the OPEN method shown in Figure 49 is a representative example of a relatively full-featured "open" method provided by the preferred embodiment. Figure 49 shows a macroscopic view of the OPEN method. Figures 49a-49f are

together an example of detailed program controlled steps performed to implement the method shown in Figure 49.

The OPEN method process starts with an "open event." This open event may be generated by a user application, an operating system intercept or various other mechanisms for capturing or intercepting control. For example, a user application may issue a request for access to a particular content stored within the VDE container. As another example, another method may issue a command.

In the example shown, the open event is processed by a control method 1502. Control method 1502 may call other methods to process the event. For example, control method 1502 may call an EVENT method 1504, a METER method 1506, a BILLING method 1508, and a BUDGET method 1510. Not all OPEN control methods necessarily call of these additional methods, but the OPEN method 1500 shown in Figure 49 is a representative example.

Control method 1502 passes a description of the open event to EVENT method 1504. EVENT method 1504 may determine, for example, whether the open event is permitted and whether the open event is significant in the sense that it needs to be processed by METER method 1506, BILLING method 1508,

and/or BUDGET method 1510. EVENT method 1504 may maintain audit trail information within an audit trail UDE, and may determine permissions and significance of the event by using an Event Method Data Element (MDE). EVENT method 1504 may also map the open event into an "atomic element" and count that may be processed by METER method 1506, BILLING method 1508, and/or BUDGET method 1510.

In OPEN method 1500, once EVENT method 1504 has been called and returns successfully, control method 1502 then may call METER method 1506 and pass the METER method, the atomic element and count returned by EVENT method 1504. METER method 1506 may maintain audit trail information in a METER method Audit Trail UDE, and may also maintain meter information in a METER method UDE. In the preferred embodiment, METER method 1506 returns a meter value to control method 1502 assuming successful completion.

In the preferred embodiment, control method 1502 upon receiving an indication that METER method 1506 has completed successfully, then calls BILLING method 1508. Control method 1502 may pass to BILLING method 1508 the meter value provided by METER method 1506. BILLING method 1508 may read and update billing information maintained in a BILLING method map MDE, and may also maintain and update audit trail

in a BILLING method Audit Trail UDE. BILLING method 1508 may return a billing amount and a completion code to control method 1502.

Assuming BILLING method 1508 completes successfully, control method 1502 may pass the billing value provided by BILLING method 1508 to BUDGET method 1510. BUDGET method 1510 may read and update budget information within a BUDGET method UDE, and may also maintain audit trail information in a BUDGET method Audit Trail UDE. BUDGET method 1510 may return a budget value to control method 1502, and may also return a completion code indicating whether the open event exceeds the user's budget (for this type of event).

Upon completion of BUDGET method 1510, control method 1502 may create a channel and establish read/use control information in preparation for subsequent calls to the READ method.

Figures 49a-49f are a more detailed description of the OPEN method 1500 example shown in Figure 49. Referring to Figure 49a, in response to an open event, control method 1502 first may determine the identification of the object to be opened and the identification of the user that has requested the object to be opened (block 1520). Control method 1502 then determines

whether the object to be opened is registered for this user (decision block 1522). It makes this determination at least in part in the preferred embodiment by reading the PERC 808 and the User Rights Table (URT) element associated with the particular object and particular user determined by block 1520 (block 1524). If the user is not registered for this particular object ("no" exit to decision block 1522), then control method 1502 may call the REGISTER method for the object and restart the OPEN method 1500 once registration is complete (block 1526). The REGISTER method block 1526 may be an independent process and may be time independent. It may, for example, take a relatively long time to complete the REGISTER method (say if the VDE distributor or other participant responsible for providing registration wants to perform a credit check on the user before registering the user for this particular object).

Assuming the proper URT for this user and object is present such that the object is registered for this user ("yes" exit to decision block 1522), control method 1502 may determine whether the object is already open for this user (decision block 1528). This test may avoid creating a redundant channel for opening an object that is already open. Assuming the object is not already open ("no" exit to decision block 1528), control method 1502 creates a channel and binds appropriate open control elements to it (block 1530). It reads the appropriate open control

elements from the secure database (or the container, such as, for example, in the case of a travelling object), and "binds" or "links" these particular appropriate control elements together in order to control opening of the object for this user. Thus, block 1530 associates an event with one or more appropriate method core(s), appropriate load modules, appropriate User Data Elements, and appropriate Method Data Elements read from the secure database (or the container) (block 1532). At this point, control method 1502 specifies the open event (which started the OPEN method to begin with), the object ID and user ID (determined by block 1520), and the channel ID of the channel created by block 1530 to subsequent EVENT method 1504, METER method 1506, BILLING method 1508 and BUDGET method 1510 to provide a secure database "transaction" (block 1536). Before doing so, control method 1502 may prime an audit process (block 1533) and write audit information into an audit UDE (block 1534) so a record of the transaction exists even if the transaction fails or is interfered with.

The detail steps performed by EVENT method 1504 are set forth on Figure 49b. EVENT method 1504 may first prime an event audit trail if required (block 1538) which may write to an EVENT Method Audit Trail UDE (block 1540). EVENT method 1504 may then perform the step of mapping the open event to an atomic element number and event count using a map MDE (block 1542). The EVENT method map MDE may be read from the

secure database (block 1544). This mapping process performed by block 1542 may, for example, determine whether or not the open event is meterable, billable, or budgetable, and may transform the open event into some discrete atomic element for metering, billing and/or budgeting. As one example, block 1542 might perform a one-to-one mapping between open events and "open" atomic elements, or it may only provide an open atomic element for every fifth time that the object is opened. The map block 1542 preferably returns the open event, the event count, the atomic element number, the object ID, and the user ID. This information may be written to the EVENT method Audit Trail UDE (block 1546, 1548). In the preferred embodiment, a test (decision block 1550) is then performed to determine whether the EVENT method failed. Specifically, decision block 1550 may determine whether an atomic element number was generated. If no atomic element number was generated (e.g., meaning that the open event is not significant for processing by METER method 1506, BILLING method 1508 and/or BUDGET method 1510), then EVENT method 1504 may return a "fail" completion code to control method 1502 ("no" exit to decision block 1550).

Control method 1502 tests the completion code returned by EVENT method 1504 to determine whether it failed or was successful (decision block 1552). If the EVENT method failed ("no" exit to decision block 1552), control method 1502 may "roll

back™ the secure database transaction (block 1554) and return itself with an indication that the OPEN method failed (block 1556). In this context, "rolling back" the secure database transaction means, for example, "undoing" the changes made to audit trail UDE by blocks 1540, 1548. However, this "roll back" performed by block 1554 in the preferred embodiment does not "undo" the changes made to the control method audit UDE by blocks 1532, 1534.

Assuming the EVENT method 1504 completed successfully, control method 1502 then calls the METER method 1506 shown on Figure 49c. In the preferred embodiment, METER method 1506 primes the meter audit trail if required (block 1558), which typically involves writing to a METER method audit trail UDE (block 1560). METER method 1506 may then read a METER method UDE from the secure database (block 1562), modify the meter UDE by adding an appropriate event count to the meter value contained in the meter UDE (block 1564), and then writing the modified meter UDE back to the secure database (block 1562). In other words, block 1564 may read the meter UDE, increment the meter count it contains, and write the changed meter UDE back to the secure database. In the preferred embodiment, METER method 1506 may then write meter audit trail information to the METER method audit trail UDE if required (blocks 1566, 1568). METER method 1506

preferably next performs a test to determine whether the meter increment succeeded (decision block 1570). METER method 1506 returns to control method 1502 with a completion code (e.g., succeed or fail) and a meter value determined by block 1564.

Control method 1502 tests whether the METER method succeeded by examining the completion code, for example (decision block 1572). If the METER method failed ("no" exit to decision block 1572), then control method 1502 "rolls back" a secure database transaction (block 1574), and returns with an indication that the OPEN method failed (block 1576). Assuming the METER method succeeded ("yes" exit to decision block 1572), control method 1502 calls the BILLING method 1508 and passes it the meter value provided by METER method 1506.

An example of steps performed by BILLING method 1508 is set forth in Figure 49d. BILLING method 1508 may prime a billing audit trail if required (block 1578) by writing to a BILLING method Audit Trail UDE within the secure database (block 1580). BILLING method 1508 may then map the atomic element number, count and meter value to a billing amount using a BILLING method map MDE read from the secure database (blocks 1582, 1584). Providing an independent BILLING method map MDE containing, for example, price list information, allows separately deliverable pricing for the billing process. The

resulting billing amount generated by block 1582 may be written to the BILLING method Audit Trail UDE (blocks 1586, 1588), and may also be returned to control method 1502. In addition, BILLING method 1508 may determine whether a billing amount was properly selected by block 1582 (decision block 1590). In this example, the test performed by block 1590 generally requires more than mere examination of the returned billing amount, since the billing amount may be changed in unpredictable ways as specified by BILLING method map MDE. Control then returns to control method 1502, which tests the completion code provided by BILLING method 1508 to determine whether the BILLING method succeeded or failed (block 1592). If the BILLING method failed ("no" exit to decision block 1592), control method 1502 may "roll back" the secure database transaction (block 1594), and return an indication that the OPEN method failed (block 1596). Assuming the test performed by decision block 1592 indicates that the BILLING method succeeded ("yes" exit to decision block 1592), then control method 1502 may call BUDGET method 1510.

Other BILLING methods may use site, user and/or usage information to establish, for example, pricing information. For example, information concerning the presence or absence of an object may be used in establishing "suite" purchases, competitive discounts, etc. Usage levels may be factored into a BILLING

method to establish price breaks for different levels of usage. A currency translation feature of a BILLING method may allow purchases and/or pricing in many different currencies. Many other possibilities exist for determining an amount of budget consumed by an event that may be incorporated into BILLING methods.

An example of detailed control steps performed by BUDGET method 1510 is set forth in Figure 49e. BUDGET method 1510 may prime a budget audit trail if required by writing to a budget trail UDE (blocks 1598, 1600). BUDGET method 1510 may next perform a billing operation by adding a billing amount to a budget value (block 1602). This operation may be performed, for example, by reading a BUDGET method UDE from the secure database, modifying it, and writing it back to the secure database (block 1604). BUDGET method 1510 may then write the budget audit trail information to the BUDGET method Audit Trail UDE (blocks 1606, 1608). BUDGET method 1510 may finally, in this example, determine whether the user has run out of budget by determining whether the budget value calculated by block 1602 is out of range (decision block 1610). If the user has run out of budget ("yes" exit to decision block 1610), the BUDGET method 1510 may return a "fail completion" code to control method 1502. BUDGET method 1510 then returns to control method 1502, which tests whether the BUDGET method

completion code was successful (decision block 1612). If the BUDGET method failed ("no" exit to decision block 1612), control method 1502 may "roll back" the secure database transaction and itself return with an indication that the OPEN method failed (blocks 1614, 1616). Assuming control method 1502 determines that the BUDGET method was successful, the control method may perform the additional steps shown on Figure 49f. For example, control method 1502 may write an open audit trail if required by writing audit information to the audit UDE that was primed at block 1532 (blocks 1618, 1620). Control method 1502 may then establish a read event processing (block 1622), using the User Right Table and the PERC associated with the object and user to establish the channel (block 1624). This channel may optionally be shared between users of the VDE node 600, or may be used only by a specified user.

Control method 1502 then, in the preferred embodiment, tests whether the read channel was established successfully (decision block 1626). If the read channel was not successfully established ("no" exit to decision block 1626), control method 1502 "rolls back" the secured database transaction and provides an indication that the OPEN method failed (blocks 1628, 1630). Assuming the read channel was successfully established ("yes" exit to decision block 1626), control method 1502 may "commit" the secure database transaction (block 1632). This step of

"committing" the secure database transaction in the preferred embodiment involves, for example, deleting intermediate values associated with the secure transaction that has just been performed and, in one example, writing changed UDEs and MDEs to the secure database. It is generally not possible to "roll back" a secure transaction once it has been committed by block 1632. Then, control method 1502 may "tear down" the channel for open processing (block 1634) before terminating (block 1636). In some arrangements, such as multi-tasking VDE node environments, the open channel may be constantly maintained and available for use by any OPEN method that starts. In other implementations, the channel for open processing may be rebuilt and restarted each time an OPEN method starts.

Read

Figure 50, 50a-50f show examples of process control steps for performing a representative example of a READ method 1650. Comparing Figure 50 with Figure 49 reveals that the same overall high level processing may typically be performed for READ method 1650 as was described in connection with OPEN method 1500. Thus, READ method 1650 may call a control method 1652 in response to a read event, the control method in turn invoking an EVENT method 1654, a METER method 1656, a BILLING method 1658 and a BUDGET method 1660. In the preferred embodiment, READ control method 1652 may request

methods to fingerprint and/or obscure content before releasing the decrypted content.

Figures 50a-50e are similar to Figures 49a-49e. Of course, even though the same user data elements may be used for both the OPEN method 1500 and the READ method 1650, the method data elements for the READ method may be completely different, and in addition, the user data elements may provide different auditing, metering, billing and/or budgeting criteria for read as opposed to open processing.

Referring to Figure 50f, the READ control method 1652 must determine which key to use to decrypt content if it is going to release decrypted content to the user (block 1758). READ control method 1652 may make this key determination based, in part, upon the PERC 808 for the object (block 1760). READ control method 1652 may then call an ACCESS method to actually obtain the encrypted content to be decrypted (block 1762). The content is then decrypted using the key determined by block 1758 (block 1764). READ control method 1652 may then determine whether a "fingerprint" is desired (decision block 1766). If fingerprinting of the content is desired ("yes" exit of decision block 1766), READ control method 1652 may call the FINGERPRINT method (block 1768). Otherwise, READ control method 1652 may determine whether it is desired to obscure the

decrypted content (decision block 1770). If so, READ control method 1652 may call an OBSCURE method to perform this function (block 1772). Finally, READ control method 1652 may commit the secure database transaction (block 1774), optionally tear down the read channel (not shown), and terminate (block 1776).

Write

Figures 51, 51a-51f are flowcharts of examples of process control steps used to perform a representative example of a WRITE method 1780 in the preferred embodiment. WRITE method 1780 uses a control method 1782 to call an EVENT method 1784, METER method 1786, BILLING method 1788, and BUDGET method 1790 in this example. Thus, writing information into a container (either by overwriting information already stored in the container or adding new information to the container) in the preferred embodiment may be metered, billed and/or budgeted in a manner similar to the way opening a container and reading from a container can be metered, billed and budgeted. As shown in Figure 51, the end result of WRITE method 1780 is typically to encrypt content, update the container table of contents and related information to reflect the new content, and write the content to the object.

Figure 51a for the WRITE control method 1782 is similar to Figure 49a and Figure 50a for the OPEN control method and the READ control method, respectively. However, Figure 51b is slightly different from its open and read counterparts. In particular, block 1820 is performed if the WRITE EVENT method 1784 fails. This block 1820 updates the EVENT method map MDE to reflect new data. This is necessary to allow information written by block 1810 to be read by Figure 51b READ method block 1678 based on the same (but now updated) EVENT method map MDE.

Looking at Figure 51f, once the EVENT, METER, BILLING and BUDGET methods have returned successfully to WRITE control method 1782, the WRITE control method writes audit information to Audit UDE (blocks 1890, 1892), and then determines (based on the PERC for the object and user and an optional algorithm) which key should be used to encrypt the content before it is written to the container (blocks 1894, 1896). CONTROL method 1782 then encrypts the content (block 1898) possibly by calling an ENCRYPT method, and writes the encrypted content to the object (block 1900). CONTROL method 1782 may then update the table of contents (and related information) for the container to reflect the newly written information (block 1902), commit the secure database transaction (block 1904), and return (block 1906).

Close

Figure 52 is a flowchart of an example of process control steps to perform a representative example of a CLOSE method 1920 in the preferred embodiment. CLOSE method 1920 is used to close an open object. In the preferred embodiment, CLOSE method 1920 primes an audit trail and writes audit information to an Audit UDE (blocks 1922, 1924). CLOSE method 1920 then may destroy the current channel(s) being used to support and/or process one or more open objects (block 1926). As discussed above, in some (e.g., multi-user or multi-tasking) installations, the step of destroying a channel is not needed because the channel may be left operating for processing additional objects for the same or different users. CLOSE method 1920 also releases appropriate records and resources associated with the object at this time (block 1926). The CLOSE method 1920 may then write an audit trail (if required) into an Audit UDE (blocks 1928, 1930) before completing.

Event

Figure 53a is a flowchart of example process control steps provided by a more general example of an EVENT method 1940 provided by the preferred embodiment. Examples of EVENT methods are set forth in Figures 49b, 50b and 51b and are described above. EVENT method 1940 shown in Figure 53a is somewhat more generalized than the examples above. Like the

EVENT method examples above, EVENT method 1940 receives an identification of the event along with an event count and event parameters. EVENT method 1940 may first prime an EVENT audit trail (if required) by writing appropriate information to an EVENT method Audit Trail UDE (blocks 1942, 1944). EVENT method 1940 may then obtain and load an EVENT method map DTD from the secure database (blocks 1946, 1948). This EVENT method map DTD describes, in this example, the format of the EVENT method map MDE to be read and accessed immediately subsequently (by blocks 1950, 1952). In the preferred embodiment, MDEs and UDEs may have any of various different formats, and their formats may be flexibly specified or changed dynamically depending upon the installation, user, etc. The DTD, in effect, describes to the EVENT method 1940 how to read from the EVENT method map MDE. DTDs are also used to specify how methods should write to MDEs and UDEs, and thus may be used to implement privacy filters by, for example, preventing certain confidential user information from being written to data structures that will be reported to third parties.

Block 1950 ("map event to atomic element # and event count using a Map MDE") is in some sense the "heart" of EVENT method 1940. This step "maps" the event into an "atomic element number" to be responded to by subsequently called methods. An example of process control steps performed by a

somewhat representative example of this "mapping" step 1950 is shown in Figure 53b.

The Figure 53b example shows the process of converting a READ event that is associated with requesting byte range 1001-1500 from a specific piece of content into an appropriate atomic element. The example EVENT method mapping process (block 1950 in Figure 53a) can be detailed as the representative process shown in Figure 53b.

EVENT method mapping process 1950 may first look up the event code (READ) in the EVENT method MDE (1952) using the EVENT method map DTD (1948) to determine the structure and contents of the MDE. A test might then be performed to determine if the event code was found in the MDE (1956), and if not ("No" branch), the EVENT method mapping process may terminate (1958) without mapping the event to an atomic element number and count. If the event was found in the MDE ("Yes" branch), the EVENT method mapping process may then compare the event range (e.g., bytes 1001-1500) against the atomic element to event range mapping table stored in the MDE (block 1960). The comparison might yield one or more atomic element numbers or the event range might not be found in the mapping table. The result of the comparison might then be tested (block 1962) to determine if any atomic element numbers

were found in the table. If not ("No" branch), the EVENT method mapping process may terminate without selecting any atomic element numbers or counts (1964). If the atomic element numbers were found, the process might then calculate the atomic element count from the event range (1966). In this example, the process might calculate the number of bytes requested by subtracting the upper byte range from the lower byte range (e.g., $1500 - 1001 + 1 = 500$). The example EVENT method mapping process might then terminate (block 1968) and return the atomic element number(s) and counts.

EVENT method 1940 may then write an EVENT audit trail if required to an EVENT method Audit Trail UDE (block 1970, 1972). EVENT method 1940 may then prepare to pass the atomic element number and event count to the calling CONTROL method (or other control process) (at exit point 1978). Before that, however, EVENT method 1940 may test whether an atomic element was selected (decision block 1974). If no atomic element was selected, then the EVENT method may be failed (block 1974). This may occur for a number of reasons. For example, the EVENT method may fail to map an event into an atomic element if the user is not authorized to access the specific areas of content that the EVENT method MDE does not describe. This mechanism could be used, for example, to distribute customized versions of a piece of content and control access to the various

versions in the content object by altering the EVENT method MDE delivered to the user. A specific use of this technology might be to control the distribution of different language (e.g., English, French, Spanish) versions of a piece of content.

Billing

Figure 53c is a flowchart of an example of process control steps performed by a BILLING method 1980. Examples of BILLING methods are set forth in Figures 49d, 50d, and 51d and are described above. BILLING method 1980 shown in Figure 53c is somewhat more generalized than the examples above. Like the BILLING method examples above, BILLING method 1980 receives a meter value to determine the amount to bill. BILLING method 1980 may first prime a BILLING audit trail (if required) by writing appropriate information to the BILLING method Audit Trail UDE (blocks 1982, 1984). BILLING method 1980 may then obtain and load a BILLING method map DTD from the secure database (blocks 1985, 1986), which describes the BILLING method map MDE (e.g., a price list, table, or parameters to the billing amount calculation algorithm) that should be used by this BILLING method. The BILLING method map MDE may be delivered either as part of the content object or as a separately deliverable component that is combined with the control information at registration.

The BILLING method map MDE in this example may describe the pricing algorithm that should be used in this BILLING method (e.g., bill \$0.001 per byte of content released). Block 1988 ("Map meter value to billing amount") functions in the same manner as block 1950 of the EVENT method; it maps the meter value to a billing value. Process step 1988 may also interrogate the secure database (as limited by the privacy filter) to determine if other objects or information (e.g., user information) are present as part of the BILLING method algorithm.

BILLING method 1980 may then write a BILLING audit trail if required to a BILLING method Audit Trail UDE (block 1990, 1992), and may prepare to return the billing amount to the calling CONTROL method (or other control process). Before that, however, BILLING method 1980 may test whether a billing amount was determined (decision block 1994). If no billing amount was determined, then the BILLING method may be failed (block 1996). This may occur if the user is not authorized to access the specific areas of the pricing table that the BILLING method MDE describes (e.g., you may purchase not more than \$100.00 of information from this content object).

Access

Figure 54 is a flowchart of an example of program control steps performed by an ACCESS method 2000. As described above, an ACCESS method may be used to access content embedded in an object 300 so it can be written to, read from, or otherwise manipulated or processed. In many cases, the ACCESS method may be relatively trivial since the object may, for example, be stored in a local storage that is easily accessible. However, in the general case, an ACCESS method 2000 must go through a more complicated procedure in order to obtain the object. For example, some objects (or parts of objects) may only be available at remote sites or may be provided in the form of a real-time download or feed (e.g., in the case of broadcast transmissions). Even if the object is stored locally to the VDE node, it may be stored as a secure or protected object so that it is not directly accessible to a calling process. ACCESS method 2000 establishes the connections, routings, and security requisites needed to access the object. These steps may be performed transparently to the calling process so that the calling process only needs to issue an access request and the particular ACCESS method corresponding to the object or class of objects handles all of the details and logistics involved in actually accessing the object.

ACCESS method 2000 may first prime an ACCESS audit trail (if required) by writing to an ACCESS Audit Trail UDE (blocks 2002, 2004). ACCESS method 2000 may then read and load an ACCESS method DTD in order to determine the format of an ACCESS MDE (blocks 2006, 2008). The ACCESS method MDE specifies the source and routing information for the particular object to be accessed in the preferred embodiment. Using the ACCESS method DTD, ACCESS method 2000 may load the correction parameters (e.g., by telephone number, account ID, password and/or a request script in the remote resource dependent language).

ACCESS method 2000 reads the ACCESS method MDE from the secure database, reads it in accordance with the ACCESS method DTD, and loads encrypted content source and routing information based on the MDE (blocks 2010, 2012). This source and routing information specifies the location of the encrypted content. ACCESS method 2000 then determines whether a connection to the content is available (decision block 2014). This "connection" could be, for example, an on-line connection to a remote site, a real-time information feed, or a path to a secure/protected resource, for example. If the connection to the content is not currently available ("No" exit of decision block 2014), then ACCESS method 2000 takes steps to open the connection (block 2016). If the connection fails (e.g.,

because the user is not authorized to access a protected secure resource), then the ACCESS method 2000 returns with a failure indication (termination point 2018). If the open connection succeeds, on the other hand, then ACCESS method 2000 obtains the encrypted content (block 2020). ACCESS method 2000 then writes an ACCESS audit trail if required to the secure database ACCESS method Audit Trail UDE (blocks 2022, 2024), and then terminates (terminate point 2026).

Decrypt and Encrypt

Figure 55a is a flowchart of an example of process control steps performed by a representative example of a DECRYPT method 2030 provided by the preferred embodiment. DECRYPT method 2030 in the preferred embodiment obtains or derives a decryption key from an appropriate PERC 808, and uses it to decrypt a block of encrypted content. DECRYPT method 2030 is passed a block of encrypted content or a pointer to where the encrypted block is stored. DECRYPT 2030 selects a key number from a key block (block 2032). For security purposes, a content object may be encrypted with more than one key. For example, a movie may have the first 10 minutes encrypted using a first key, the second 10 minutes encrypted with a second key, and so on. These keys are stored in a PERC 808 in a structure called a "key block." The selection process involves determining the correct key to use from the key block in order to decrypt the content. The

process for this selection is similar to the process used by EVENT methods to map events into atomic element numbers. DECRYPT method 2030 may then access an appropriate PERC 808 from the secure database 610 and loads a key (or "seed") from a PERC (blocks 2034, 2036). This key information may be the actual decryption key to be used to decrypt the content, or it may be information from which the decryption key may be at least in part derived or calculated. If necessary, DECRYPT method 2030 computes the decryption key based on the information read from PERC 808 at block 2034 (block 2038). DECRYPT method 2030 then uses the obtained and/or calculated decryption key to actually decrypt the block of encrypted information (block 2040). DECRYPT method 2030 outputs the decrypted block (or the pointer indicating where it may be found), and terminates (termination point 2042).

Figure 55b is a flowchart of an example of process control steps performed by a representative example of an ENCRYPT method 2050. ENCRYPT method 2050 is passed as an input, a block of information to encrypt (or a pointer indicating where it may be found). ENCRYPT method 2050 then may determine an encryption key to use from a key block (block 2052). The encryption key selection makes a determination if a key for a specific block of content to be written already exists in a key block stored in PERC 808. If the key already exists in the key block,

then the appropriate key number is selected. If no such key exists in the key block, a new key is calculated using an algorithm appropriate to the encryption algorithm. This key is then stored in the key block of PERC 808 so that DECRYPT method 2030 may access the key in order to decrypt the content stored in the content object. ENCRYPT method 2050 then accesses the appropriate PERC to obtain, derive and/or compute an encryption key to be used to encrypt the information block (blocks 2054, 2056, 2058, which are similar to Figure 55a blocks 2034, 2036, 2038). ENCRYPT method 2050 then actually encrypts the information block using the obtained and/or derived encryption key (block 2060) and outputs the encrypted information block or a pointer where it can be found before terminating (termination point 2062).

Content

Figure 56 is a flowchart of an example of process control steps performed by a representative of a CONTENT method 2070 provided by the preferred embodiment. CONTENT method 2070 in the preferred embodiment builds a "synopsis" of protected content using a secure process. For example, CONTENT method 2070 may be used to derive unsecure ("public") information from secure content. Such derived public information might include, for example, an abstract, an index, a table of contents, a directory of files, a schedule when content may be available, or excerpts such as for example, a movie "trailer."

CONTENT method 2070 begins by determining whether the derived content to be provided must be derived from secure contents, or whether it is already available in the object in the form of static values (decision block 2070). Some objects may, for example, contain prestored abstracts, indexes, tables of contents, etc., provided expressly for the purpose of being extracted by the CONTENT method 2070. If the object contains such static values ("static" exit to decision block 2072), then CONTENT method 2070 may simply read this static value content information from the object (block 2074), optionally decrypt, and release this content description (block 2076). If, on the other hand, CONTENT method 2070 must derive the synopsis/content description from the secure object ("derived" exit to decision block 2072), then the CONTENT method may then securely read information from the container according to a synopsis algorithm to produce the synopsis (block 2078).

Extract and Embed

Figure 57a is a flowchart of an example of process control steps performed by a representative example of an EXTRACT method 2080 provided by the preferred embodiment. EXTRACT method 2080 is used to copy or remove content from an object and place it into a new object. In the preferred embodiment, the EXTRACT method 2080 does not involve any release of content, but rather simply takes content from one container and places it

into another container, both of which may be secure. Extraction of content differs from release in that the content is never exposed outside a secure container. Extraction and Embedding are complementary functions; extract takes content from a container and creates a new container containing the extracted content and any specified control information associated with that content. Embedding takes content that is already in a container and stores it (or the complete object) in another container directly and/or by reference, integrating the control information associated with existing content with those of the new content.

EXTRACT method 2080 begins by priming an Audit UDE (blocks 2082, 2084). EXTRACT method then calls a BUDGET method to make sure that the user has enough budget for (and is authorized to) extract content from the original object (block 2086). If the user's budget does not permit the extraction ("no" exit to decision block 2088), then EXTRACT method 2080 may write a failure audit record (block 2090), and terminate (termination point 2092). If the user's budget permits the extraction ("yes" exit to decision block 2088), then the EXTRACT method 2080 creates a copy of the extracted object with specified rules and control information (block 2094). In the preferred embodiment, this step involves calling a method that actually controls the copy. This step may or may not involve decryption

and encryption, depending on the particular the PERC 808 associated with the original object, for example. EXTRACT method 2080 then checks whether any control changes are permitted by the rights authorizing the extract to begin with (decision block 2096). In some cases, the extract rights require an exact copy of the PERC 808 associated with the original object (or a PERC included for this purpose) to be placed in the new (destination) container ("no" exit to decision block 2096). If no control changes are permitted, then extract method 2080 may simply write audit information to the Audit UDE (blocks 2098, 2100) before terminating (terminate point 2102). If, on the other hand, the extract rights permit the user to make control changes ("yes" to decision block 2096), then EXTRACT method 2080 may call a method or load module that solicits new or changed control information (e.g., from the user, the distributor who created/granted extract rights, or from some other source) from the user (blocks 2104, 2106). EXTRACT method 2080 may then call a method or load module to create a new PERC that reflects these user-specified control information (block 2104). This new PERC is then placed in the new (destination) object, the auditing steps are performed, and the process terminates.

Figure 57b is an example of process control steps performed by a representative example of an EMBED method 2110 provided by the preferred embodiment. EMBED method

2110 is similar to EXTRACT method 2080 shown in Figure 57a. However, the EMBED method 2110 performs a slightly different function—it writes an object (or reference) into a destination container. Blocks 2112-2122 shown in Figure 57b are similar to blocks 2082-2092 shown in Figure 57a. At block 2124, EMBED method 2110 writes the source object into the destination container, and may at the same time extract or change the control information of the destination container. One alternative is to simply leave the control information of the destination container alone, and include the full set of control information associated with the object being embedded in addition to the original container control information. As an optimization, however, the preferred embodiment provides a technique whereby the control information associated with the object being embedded are "abstracted" and incorporated into the control information of the destination container. Block 2124 may call a method to abstract or change this control information. EMBED method 2110 then performs steps 2126-2130 which are similar to steps 2096, 2104, 2106 shown in Figure 57a to allow the user, if authorized, to change and/or specify control information associated with the embedded object and/or destination container. EMBED method 2110 then writes audit information into an Audit UDE (blocks 2132, 2134), before terminating (at termination point 2136).

Obscure

Figure 58a is a flowchart of an example of process control steps performed by a representative example of an OBSCURE method 2140 provided by the preferred embodiment. OBSCURE method 2140 is typically used to release secure content in devalued form. For example, OBSCURE method 2140 may release a high resolution image in a lower resolution so that a viewer can appreciate the image but not enjoy its full value. As another example, the OBSCURE method 2140 may place an obscuring legend (e.g., "COPY," "PROOF," etc.) across an image to devalue it. OBSCURE method 2140 may "obscure" text, images, audio information, or any other type of content.

OBSCURE method 2140 first calls an EVENT method to determine if the content is appropriate and in the range to be obscured (block 2142). If the content is not appropriate for obscuring, the OBSCURE method terminates (decision block 2144 "no" exit, terminate point 2146). Assuming that the content is to be obscured ("yes" exit to decision block 2144), then OBSCURE method 2140 determines whether it has previously been called to obscure this content (decision block 2148). Assuming the OBSCURE 2140 has not previously called for this object/content ("yes" exit to decision block 2148), the OBSCURE method 2140 reads an appropriate OBSCURE method MDE from the secure database and loads an obscure formula and/or pattern

from the MDE (blocks 2150, 2152). The OBSCURE method 2140 may then apply the appropriate obscure transform based on the patterns and/or formulas loaded by block 2150 (block 2154). The OBSCURE method then may terminate (terminate block 2156).

Fingerprint

Figure 58b is a flowchart of an example of process control steps performed by a representative example of a FINGERPRINT method 2160 provided by the preferred embodiment. FINGERPRINT method 2160 in the preferred embodiment operates to "mark" released content with a "fingerprint" identification of who released the content and/or check for such marks. This allows one to later determine who released unsecured content by examining the content. FINGERPRINT method 2160 may, for example, insert a user ID within a datastream representing audio, video, or binary format information. FINGERPRINT method 2160 is quite similar to OBSCURE method 2140 shown in Figure 58a except that the transform applied by FINGERPRINT method block 2174 "fingerprints" the released content rather than obscuring it.

Figure 58c shows an example of a "fingerprinting" procedure 2160 that inserts into released content "fingerprints" 2161 that identify the object and/or property and/or the user that

requested the released content and/or the date and time of the release and/or other identification criteria of the released content.

Such fingerprints 2161 can be "buried" -- that is inserted in a manner that hides the fingerprints from typical users, sophisticated "hackers," and/or from all users, depending on the file format, the sophistication and/or variety of the insertion algorithms, and on the availability of original, non-fingerprinted content (for comparison for reverse engineering of algorithm(s)). Inserted or embedded fingerprints 2161, in a preferred embodiment, may be at least in part encrypted to make them more secure. Such encrypted fingerprints 2161 may be embedded within released content provided in "clear" (plaintext) form.

Fingerprints 2161 can be used for a variety of purposes including, for example, the often related purposes of proving misuse of released materials and proving the source of released content. Software piracy is a particularly good example where fingerprinting can be very useful. Fingerprinting can also help to enforce content providers' rights for most types of electronically delivered information including movies, audio recordings, multimedia, information databases, and traditional "literary" materials. Fingerprinting is a desirable alternative or addition to copy protection.

Most piracy of software applications, for example, occurs not with the making of an illicit copy by an individual for use on another of the individual's own computers, but rather in giving a copy to another party. This often starts a chain (or more accurately a pyramid) of illegal copies, as copies are handed from individual to individual. The fear of identification resulting from the embedding of a fingerprint 2161 will likely dissuade most individuals from participating, as many currently do, in widespread, "casual" piracy. In some cases, content may be checked for the presence of a fingerprint by a fingerprint method to help enforce content providers' rights.

Different fingerprints 2161 can have different levels of security (e.g., one fingerprint 2161(1) could be readable/identifiable by commercial concerns, while another fingerprint 2161(2) could be readable only by a more trusted agency. The methods for generating the more secure fingerprint 2161 might employ more complex encryption techniques (e.g., digital signatures) and/or obscuring of location methodologies. Two or more fingerprints 2161 can be embedded in different locations and/or using different techniques to help protect fingerprinted information against hackers. The more secure fingerprints might only be employed periodically rather than each time content release occurs, if the technique used to provide a more secure fingerprint involves an undesired amount of

additional overhead. This may nevertheless be effective since a principal objective of fingerprinting is deterrence—that is the fear on the part of the creator of an illicit copy that the copying will be found out.

For example, one might embed a copy of a fingerprint 2161 which might be readily identified by an authorized party—for example a distributor, service personal, client administrator, or clearinghouse using a VDE electronic appliance 600. One might embed one or more additional copies or variants of a fingerprint 2161 (e.g., fingerprints carrying information describing some or all relevant identifying information) and this additional one or more fingerprints 2161 might be maintained in a more secure manner.

Fingerprinting can also protect privacy concerns. For example, the algorithm and/or mechanisms needed to identify the fingerprint 2161 might be available only through a particularly trusted agent.

Fingerprinting 2161 can take many forms. For example, in an image, the color of every N pixels (spread across an image, or spread across a subset of the image) might be subtly shifted in a visually unnoticeable manner (at least according to the normal, unaided observer). These shifts could be interpreted by analysis

of the image (with or without access to the original image), with each occurrence or lack of occurrence of a shift in color (or greyscale) being one or more binary "on or off" bits for digital information storage. The N pixels might be either consistent, or alternatively, pseudo-random in order (but interpretable, at least in part, by a object creator, object provider, client administrator, and/or VDE administrator).

Other modifications of an image (or moving image, audio, etc.) which provide a similar benefit (that is, storing information in a form that is not normally noticeable as a result of a certain modification of the source information) may be appropriate, depending on the application. For example, certain subtle modifications in the frequency of stored audio information can be modified so as to be normally unnoticeable to the listener while still being readable with the proper tools. Certain properties of the storage of information might be modified to provide such slight but interpretable variations in polarity of certain information which is optically stored to achieve similar results. Other variations employing other electronic, magnetic, and/or optical characteristic may be employed.

Content stored in files that employ graphical formats, such as Microsoft Windows word processing files, provide significant opportunities for "burying" a fingerprint 2161. Content that

includes images and/or audio provides the opportunity to embed fingerprints 2161 that may be difficult for unauthorized individuals to identify since, in the absence of an "unfingerprinted" original for purposes of comparison, minor subtle variations at one or more time instances in audio frequencies, or in one or more video images, or the like, will be in themselves undiscernible given the normally unknown nature of the original and the large amounts of data employed in both image and sound data (and which is not particularly sensitive to minor variations). With formatted text documents, particularly those created with graphical word processors (such as Microsoft Windows or Apple MacIntosh word processors and their DOS and Unix equivalents), fingerprints 2161 can normally be inserted unobtrusively into portions of the document data representation that are not normally visible to the end user (such as in a header or other non-displayed data field).

Yet another form of fingerprinting, which may be particularly suitable for certain textual documents, would employ and control the formation of characters for a given font. Individual characters may have a slightly different visual formation which connotes certain "fingerprint" information. This alteration of a given character's form would be generally undiscernible, in part because so many slight variations exist in versions of the same font available from different suppliers, and

in part because of the smallness of the variation. For example, in a preferred embodiment, a program such as Adobe Type Align could be used which, in its off-the-shelf versions, supports the ability of a user to modify font characters in a variety of ways. The mathematical definition of the font character is modified according to the user's instructions to produce a specific set of modifications to be applied to a character or font. Information content could be used in an analogous manner (as an alternative to user selections) to modify certain or all characters too subtly for user recognition under normal circumstances but which nevertheless provide appropriate encoding for the fingerprint 2161. Various subtly different versions of a given character might be used within a single document so as to increase the ability to carry transaction related font fingerprinted information.

Some other examples of applications for fingerprinting might include:

1. In software programs, selecting certain interchangeable code fragments in such a way as to produce more or less identical operation, but on analysis, differences that detail fingerprint information.
2. With databases, selecting to format certain fields, such as dates, to appear in different ways.

3. In games, adjusting backgrounds, or changing order of certain events, including noticeable or very subtle changes in timing and/or ordering of appearance of game elements, or slight changes in the look of elements of the game.

Fingerprinting method 2160 is typically performed (if at all) at the point at which content is released from a content object 300. However, it could also be performed upon distribution of an object to "mark" the content while still in encrypted form. For example, a network-based object repository could embed fingerprints 2161 into the content of an object before transmitting the object to the requester, the fingerprint information could identify a content requester/end user. This could help detect "spoof" electronic appliances 600 used to release content without authorization.

Destroy

Figure 59 is a flowchart of an example of process control steps performed by a representative performed by a DESTROY method 2180 provided by the preferred embodiment. DESTROY method 2180 removes the ability of a user to use an object by destroying the URT the user requires to access the object. In the preferred embodiment, a DESTROY method 2180 may first write audit information to an Audit UDE (blocks 2182, 2184).

DESTROY method 2180 may then call a WRITE and/or ACCESS method to write information which will corrupt (and thus destroy) the header and/or other important parts of the object (block 2186). DESTROY method 2180 may then mark one or more of the control structures (e.g., the URT) as damaged by writing appropriate information to the control structure (blocks 2188, 2190). DESTROY method 2180, finally, may write additional audit information to Audit UDE (blocks 2192, 2194) before terminating (terminate point 2196).

Panic

Figure 60 is a flowchart of an example of process control steps performed by a representative example of a PANIC method 2200 provided by the preferred embodiment. PANIC method 2200 may be called when a security violation is detected. PANIC method 2200 may prevent the user from further accessing the object currently being accessed by, for example, destroying the channel being used to access the object and marking one or more of the control structures (e.g., the URT) associated with the user and object as damaged (blocks 2206, and 2208-2210, respectively). Because the control structure is damaged, the VDE node will need to contact an administrator to obtain a valid control structure(s) before the user may access the same object again. When the VDE node contacts the administrator, the administrator may request information sufficient to satisfy itself

that no security violation occurred, or if a security violation did occur, take appropriate steps to ensure that the security violation is not repeated.

Meter

Figure 61 is a flowchart of an example of process control steps performed by a representative example of a METER method provided by the preferred embodiment. Although METER methods were described above in connection with Figures 49, 50 and 51, the METER method 2220 shown in Figure 61 is possibly a somewhat more representative example. In the preferred embodiment, METER method 2220 first primes an Audit Trail by accessing a METER Audit Trail UDE (blocks 2222, 2224). METER method 2220 may then read the DTD for the Meter UDE from the secure database (blocks 2226, 2228). METER method 2220 may then read the Meter UDE from the secure database (blocks 2230, 2232). METER method 2220 next may test the obtained Meter UDE to determine whether it has expired (decision block 2234). In the preferred embodiment, each Meter UDE may be marked with an expiration date. If the current date/time is later than the expiration date of the Meter UDE ("yes" exit to decision block 2234), then the METER method 2220 may record a failure in the Audit Record and terminate with a failure condition (block 2236, 2238).

Assuming the Meter UDE is not yet expired, the meter method 2220 may update it using the atomic element and event count passed to the METER method from, for example, an EVENT method (blocks 2239, 2240). The METER method 2220 may then save the Meter Use Audit Record in the Meter Audit Trail UDE (blocks 2242, 2244), before terminating (at terminate point 2246).

Additional Security Features Provided by the Preferred Embodiment

VDE 100 provided by the preferred embodiment has sufficient security to help ensure that it cannot be compromised short of a successful "brute force attack," and so that the time and cost to succeed in such a "brute force attack" substantially exceeds any value to be derived. In addition, the security provided by VDE 100 compartmentalizes the internal workings of VDE so that a successful "brute force attack" would compromise only a strictly bounded subset of protected information, not the entire system.

The following are among security aspects and features provided by the preferred embodiment:

- security of PPE 650 and the processes it performs
- security of secure database 610

- security of encryption/decryption performed by PPE 650
- key management; security of encryption/decryption keys and shared secrets
- security of authentication/external communications
- security of secure database backup
- secure transportability of VDE internal information between electronic appliances 600
- security of permissions to access VDE secure information
- security of VDE objects 300
- integrity of VDE security.

Some of these security aspects and considerations are discussed above. The following provides an expanded discussion of preferred embodiment security features not fully addressed elsewhere.

Management of Keys and Shared Secrets

VDE 100 uses keys and shared secrets to provide security. The following key usage features are provided by the preferred embodiment:

- different cryptosystem/key types
- secure key length
- key generation

- key "convolution" and key "aging."

Each of these types are discussed below.

A. Public-Key and Symmetric Key Cryptosystems

The process of disguising or transforming information to hide its substance is called encryption. Encryption produces "ciphertext." Reversing the encryption process to recover the substance from the ciphertext is called "decryption." A cryptographic algorithm is the mathematical function used for encryption and decryption.

Most modern cryptographic algorithms use a "key." The "key" specifies one of a family of transformations to be provided. Keys allow a standard, published and tested cryptographic algorithm to be used while ensuring that specific transformations performed using the algorithm are kept secret. The secrecy of the particular transformations thus depends on the secrecy of the key, not on the secrecy of the algorithm.

There are two general forms of key-based algorithms, either or both of which may be used by the preferred embodiment PPE 650:

- symmetric; and
- public-key ("PK").

Symmetric algorithms are algorithms where the encryption key can be calculated from the decryption key and vice versa. In many such systems, the encryption and decryption keys are the same. The algorithms, also called "secret-key", "single key" or "shared secret" algorithms, require a sender and receiver to agree on a key before ciphertext produced by a sender can be decrypted by a receiver. This key must be kept secret. The security of a symmetric algorithm rests in the key: divulging the key means that anybody could encrypt and decrypt information in such a cryptosystem. See Schneier, Applied Cryptography at Page 3. Some examples of symmetric key algorithms that the preferred embodiment may use include DES, Skipjack/Clipper, IDEA, RC2, and RC4.

In public-key cryptosystems, the key used for encryption is different from the key used for decryption. Furthermore, it is computationally infeasible to derive one key from the other. The algorithms used in these cryptosystems are called "public key" because one of the two keys can be made public without endangering the security of the other key. They are also sometimes called "asymmetric" cryptosystems because they use different keys for encryption and decryption. Examples of public-key algorithms include RSA, El Gamal and LUC.

The preferred embodiment PPE 650 may operate based on only symmetric key cryptosystems, based on public-key cryptosystems, or based on both symmetric key cryptosystems and public-key cryptosystems. VDE 100 does not require any specific encryption algorithms; the architecture provided by the preferred embodiment may support numerous algorithms including PK and/or secret key (non PK) algorithms. In some cases, the choice of encryption/decryption algorithm will be dependent on a number of business decisions such as cost, market demands, compatibility with other commercially available systems, export laws, etc.

Although the preferred embodiment is not dependent on any particular type of cryptosystem or encryption/decryption algorithm(s), the preferred example uses PK cryptosystems for secure communications between PPEs 650, and uses secret key cryptosystems for "bulk" encryption/decryption of VDE objects 300. Using secret key cryptosystems (e.g., DES implementations using multiple keys and multiple passes, Skipjack, RC2, or RC4) for "bulk" encryption/decryption provides efficiencies in encrypting and decrypting large quantities of information, and also permits PPEs 650 without PK-capability to deal with VDE objects 300 in a variety of applications. Using PK cryptosystems for communications may provide advantages such as eliminating reliance on secret shared external communication keys to

establish communications, allowing for a challenge/response that doesn't rely on shared internal secrets to authenticate PPEs 650, and allowing for a publicly available "certification" process without reliance on shared secret keys.

Some content providers may wish to restrict use of their content to PK implementations. This desire can be supported by making the availability of PK capabilities, and the specific nature or type of PK capabilities, in PPEs 650 a factor in the registration of VDE objects 300, for example, by including a requirement in a REGISTER method for such objects in the form of a load module that examines a PPE 650 for specific or general PK capabilities before allowing registration to continue.

Although VDE 100 does not require any specific algorithm, it is highly desirable that all PPEs 650 are capable of using the same algorithm for bulk encryption/decryption. If the bulk encryption/decryption algorithm used for encrypting VDE objects 300 is not standardized, then it is possible that not all VDE electronic appliances 600 will be capable of handling all VDE objects 300. Performance differences will exist between different PPEs 650 and associated electronic appliances 600 if standardized bulk encryption/decryption algorithms are not implemented in whole or in part by hardware-based encrypt/decrypt engine 522, and instead are implemented in

software. In order to support algorithms that are not implemented in whole or in part by encrypt/decrypt engine 522, a component assembly that implements such an algorithm must be available to a PPE 650.

B. Key Length

Increased key length may increase security. A "brute-force" attack of a cryptosystem involves trying every possible key. The longer the key, the more possible keys there are to try. At some key length, available computation resources will require an impractically large amount of time for a "brute force" attacker to try every possible key.

VDE 100 provided by the preferred embodiment accommodates and can use many different key lengths. The length of keys used by VDE 100 in the preferred embodiment is determined by the algorithm(s) used for encryption/decryption, the level of security desired, and throughput requirements. Longer keys generally require additional processing power to ensure fast encryption/decryption response times. Therefore, there is a tradeoff between (a) security, and (b) processing time and/or resources. Since a hardware-based PPE encrypt/decrypt engine 522 may provide faster processing than software-based encryption/decryption, the hardware-based approach may, in general, allow use of longer keys.

The preferred embodiment may use a 1024 bit modulus (key) RSA cryptosystem implementation for PK encryption/decryption, and may use 56-bit DES for "bulk" encryption/decryption. Since the 56-bit key provided by standard DES may not be long enough to provide sufficient security for at least the most sensitive VDE information, multiple DES encryptions using multiple passes and multiple DES keys may be used to provide additional security. DES can be made significantly more secure if operated in a manner that uses multiple passes with different keys. For example, three passes with 2 or 3 separate keys is much more secure because it effectively increases the length of the key. RC2 and RC4 (alternatives to DES) can be exported for up to 40-bit key sizes, but the key size probably needs to be much greater to provide even DES level security. The 80-bit key length provided by NSA's Skipjack may be adequate for most VDE security needs.

The capability of downloading code and other information dynamically into PPE 650 allows key length to be adjusted and changed dynamically even after a significant number of VDE electronic appliances 600 are in use. The ability of a VDE administrator to communicate with each PPE 650 efficiently makes such after-the-fact dynamic changes both possible and cost-effective. New or modified cryptosystems can be downloaded into existing PPEs 650 to replace or add to the cryptosystem

repertoire available within the PPE, allowing older PPEs to maintain compatibility with newer PPEs and/or newly released VDE objects 300 and other VDE-protected information. For example, software encryption/decryption algorithms may be downloaded into PPE 650 at any time to supplement the hardware-based functionality of encrypt/decrypt engine 522 by providing different key length capabilities. To provide increased flexibility, PPE encrypt/decrypt engine 522 may be configured to anticipate multiple passes and/or variable and/or longer key lengths. In addition, it may be desirable to provide PPEs 650 with the capability to internally generate longer PK keys.

C. Key Generation

Key generation techniques provided by the preferred embodiment permit PPE 650 to generate keys and other information that are "known" only to it.

The security of encrypted information rests in the security of the key used to encrypt it. If a cryptographically weak process is used to generate keys, the entire security is weak. Good keys are random bit strings so that every possible key in the key space is equally likely. Therefore, keys should in general be derived from a reliably random source, for example, by a cryptographically secure pseudo-random number generator seeded from such a source. Examples of such key generators are

described in Schneier, Applied Cryptography (John Wiley and Sons, 1994), chapter 15. If keys are generated outside a given PPE 650 (e.g., by another PPE 650), they must be verified to ensure they come from a trusted source before they can be used. "Certification" may be used to verify keys.

The preferred embodiment PPE 650 provides for the automatic generation of keys. For example, the preferred embodiment PPE 650 may generate its own public/private key pair for use in protecting PK-based external communications and for other reasons. A PPE 650 may also generate its own symmetric keys for various purposes during and after initialization. Because a PPE 650 provides a secure environment, most key generation in the preferred embodiment may occur within the PPE (with the possible exception of initial PPE keys used at manufacturing or installation time to allow a PPE to authenticate initial download messages to it).

Good key generation relies on randomness. The preferred embodiment PPE 650 may, as mentioned above in connection with Figure 9, include a hardware-based random number generator 542 with the characteristics required to generate reliable random numbers. These random numbers may be used to "seed" a cryptographically strong pseudo-random number generator (e.g., DES operated in Output Feedback Mode) for

generation of additional key values derived from the random seed. In the preferred embodiment, random number generator 542 may consist of a "noise diode" or other physically-based source of random values (e.g., radioactive decay).

If no random number generator 542 is available in the PPE 650, the SPE 503 may employ a cryptographic algorithm (e.g., DES in Output Feedback Mode) to generate a sequence of pseudo-random values derived from a secret value protected within the SPE. Although these numbers are pseudo-random rather than truly random, they are cryptographically derived from a value unknown outside the SPE 503 and therefore may be satisfactory in some applications.

In an embodiment incorporating an HPE 655 without an SPE 503, the random value generator 565 software may derive reliably random numbers from unpredictable external physical events (e.g., high-resolution timing of disk I/O completions or of user keystrokes at an attached keyboard 612).

Conventional techniques for generating PK and non-PK keys based upon such "seeds" may be used. Thus, if performance and manufacturing costs permit, PPE 650 in the preferred embodiment will generate its own public/private key pair based on such random or pseudo-random "seed" values. This key pair

may then be used for external communications between the PPE 650 that generated the key pair and other PPEs that wish to communicate with it. For example, the generating PPE 650 may reveal the public key of the key pair to other PPEs. This allows other PPEs 650 using the public key to encrypt messages that may be decrypted only by the generating PPE (the generating PPE is the only PPE that "knows" the corresponding "private key"). Similarly, the generating PPE 650 may encrypt messages using its private key that, when decrypted successfully by other PPEs with the generating PPE's public key, permit the other PPEs to authenticate that the generating PPE sent the message.

Before one PPE 650 uses a public key generated by another PPE, a public key certification process should be used to provide authenticity certificates for the public key. A public-key certificate is someone's public key "signed" by a trustworthy entity such as an authentic PPE 650 or a VDE administrator. Certificates are used to thwart attempts to convince a PPE 650 that it is communicating with an authentic PPE when it is not (e.g., it is actually communicating with a person attempting to break the security of PPE 650). One or more VDE administrators in the preferred embodiment may constitute a certifying authority. By "signing" both the public key generated by a PPE 650 and information about the PPE and/or the corresponding VDE electronic appliance 600 (e.g., site ID, user ID, expiration

date, name, address, etc.), the VDE administrator certifying authority can certify that information about the PPE and/or the VDE electronic appliance is correct and that the public key belongs to that particular VDE mode.

Certificates play an important role in the trustedness of digital signatures, and also are important in the public-key authentication communications protocol (to be discussed below). In the preferred embodiment, these certificates may include information about the trustedness/level of security of a particular VDE electronic appliance 600 (e.g., whether or not it has a hardware-based SPE 503 or is instead a less trusted software emulation type HPE 655) that can be used to avoid transmitting certain highly secure information to less trusted/secure VDE installations.

Certificates can also play an important role in decommissioning rogue users and/or sites. By including a site and/or user ID in a certificate, a PPE can evaluate this information as an aspect of authentication. For example, if a VDE administrator or clearinghouse encounters a certificate bearing an ID (or other information) that meets certain criteria (e.g., is present on a list of decommissioned and/or otherwise suspicious users and/or sites), they may choose to take actions based on those criteria such as refusing to communicate,

communicating disabling information, notifying the user of the condition, etc. Certificates also typically include an expiration date to ensure that certificates must be replaced periodically, for example, to ensure that sites and/or users must stay in contact with a VDE administrator and/or to allow certification keys to be changed periodically. More than one certificate based on different keys may be issued for sites and/or users so that if a given certification key is compromised, one or more "backup" certificates may be used. If a certification key is compromised, A VDE administrator may refuse to authenticate based on certificates generated with such a key, and send a signal after authenticating with a "backup" certificate that invalidates all use of the compromised key and all certificates associated with it in further interactions with VDE participants. A new one or more "backup" certificates and keys may be created and sent to the authenticated site/user after such a compromise.

If multiple certificates are available, some of the certificates may be reserved as backups. Alternatively or in addition, one certificate from a group of certificates may be selected (e.g., by using RNG 542) in a given authentication, thereby reducing the likelihood that a certificate associated with a compromised certification key will be used. Still alternatively, more than one certificate may be used in a given authentication.

To guard against the possibility of compromise of the certification algorithm (e.g., by an unpredictable advance in the mathematical foundations on which the algorithm is based), distinct algorithms may be used for different certificates that are based on different mathematical foundations.

Another technique that may be employed to decrease the probability of compromise is to keep secret (in protected storage in the PPE 650) the "public" values on which the certificates are based, thereby denying an attacker access to values that may aid in the attack. Although these values are nominally "public," they need be known only to those components which actually validate certificates (i.e., the PPE 650).

In the preferred embodiment, PPE 650 may generate its own certificate, or the certificate may be obtained externally, such as from a certifying authority VDE administrator. Irrespective of where the digital certificate is generated, the certificate is eventually registered by the VDE administrator certifying authority so that other VDE electronic appliances 600 may have access to (and trust) the public key. For example, PPE 650 may communicate its public key and other information to a certifying authority which may then encrypt the public key and other information using the certifying authority's private key. Other installations 600 may trust the "certificate" because it can

be authenticated by using the certifying authority's public key to decrypt it. As another example, the certifying authority may encrypt the public key it receives from the generating PPE 650 and use it to encrypt the certifying authority's private key. The certifying authority may then send this encrypted information back to the generating PPE 650. The generating PPE 650 may then use the certifying authority's private key to internally create a digital certificate, after which it may destroy its copy of the certifying authority's private key. The generating PPE 650 may then send out its digital certificate to be stored in a certification repository at the VDE administrator (or elsewhere) if desired. The certificate process can also be implemented with an external key pair generator and certificate generator, but might be somewhat less secure depending on the nature of the secure facility. In such a case, a manufacturing key should be used in PPE 650 to limit exposure to the other keys involved.

A PPE 650 may need more than one certificate. For example, a certificate may be needed to assure other users that a PPE is authentic, and to identify the PPE. Further certificates may be needed for individual users of a PPE 650. These certificates may incorporate both user and site information or may only include user information. Generally, a certifying authority will require a valid site certificate to be presented prior to creating a certificate for a given user. Users may each require

their own public key/private key pair in order to obtain certificates. VDE administrators, clearinghouses, and other participants may normally require authentication of both the site (PPE 650) and of the user in a communication or other interaction. The processes described above for key generation and certification for PPEs 650 may also be used to form site/user certificates or user certificates.

Certificates as described above may also be used to certify the origin of load modules 1100 and/or the authenticity of administrative operations. The security and assurance techniques described above may be employed to decrease the probability of compromise for any such certificate (including certificates other than the certificate for a VDE electronic appliance 600's identity).

D. Key Aging and Convolution

PPE 650 also has the ability in the preferred embodiment to generate secret keys and other information that is shared between multiple PPEs 650. In the preferred embodiment, such secret keys and other information may be shared between multiple VDE electronic appliances 600 without requiring the shared secret information to ever be communicated explicitly between the electronic appliances. More specifically, PPE 650 uses a technique called "key convolution" to derive keys based on

a deterministic process in response to seed information shared between multiple VDE electronic appliances 600. Since the multiple electronic appliances 600 "know" what the "seed" information is and also "know" the deterministic process used to generate keys based on this information, each of the electronic appliances may independently generate the "true key." This permits multiple VDE electronic appliances 600 to share a common secret key without potentially compromising its security by communicating it over an insecure channel.

No encryption key should be used for an indefinite period. The longer a key is used, the greater the chance that it may be compromised and the greater the potential loss if the key is compromised but still in use to protect new information. The longer a key is used, the more information it may protect and therefore the greater the potential rewards for someone to spend the effort necessary to break it. Further, if a key is used for a long time, there may be more ciphertext available to an attacker attempting to break the key using a ciphertext-based attack. See Schneier at 150-151. Key convolution in the preferred embodiment provides a way to efficiently change keys stored in secure database 610 on a routine periodic or other basis while simplifying key management issues surrounding the change of keys. In addition, key convolution may be used to provide "time

aged keys" (discussed below) to provide "expiration dates" for key usage and/or validity.

Figure 62 shows an example implementation of key convolution in the preferred embodiment. Key convolution may be performed using a combination of a site ID 2821 and the high-order bits of the RTC 528 to yield a site-unique value "V" that is time-dependent on a large scale (e.g., hours or days). This value "V" may be used as the key for an encryption process 2871 that transforms a convolution seed value 2861 into a "current convolution key" 2862. The seed value 2861 may be a universe-wide or group-wide shared secret value, and may be stored in secure key storage (e.g., protected memory within PPE 650). The seed value 2861 is installed during the manufacturing process and may be updated occasionally by a VDE administrator. There may be a plurality of seed values 2861 corresponding to different sets of objects 300.

The current convolution key 2862 represents an encoding of the site ID 2821 and current time. This transformed value 2862 may be used as a key for another encryption process 2872 to transform the stored key 810 in the object's PERC 808 into the true private body key 2863 for the object's contents.

The "convolution function" performed by blocks 2861, 2871 may, for example, be a one-way function that can be performed independently at both the content creator's site and at the content user's site. If the content user does not use precisely the same convolution function and precisely the same input values (e.g., time and/or site and/or other information) as used by the content creator, then the result of the convolution function performed by the content user will be different from the content creator's result. If the result is used as a symmetrical key for encryption by the content creator, the content user will not be able to decrypt unless the content user's result is the same as the result of the content creator.

The time component for input to the key convolution function may be derived from RTC 528 (care being taken to ensure that slight differences in RTC synchronization between VDE electronic appliances will not cause different electronic appliances to use different time components). Different portions of the RTC 528 output may be used to provide keys with different valid durations, or some tolerance can be built into the process to try several different key values. For example, a "time granularity" parameter can be adjusted to provide time tolerance in terms of days, weeks, or any other time period. As one example, if the "time granularity" is set to 2 days, and the tolerance is ± 2 days, then three real-time input values can be

tried as input to the convolution algorithm. Each of the resulting key values may be tried to determine which of the possible keys is actually used. In this example, the keys will have only a 4 day life span.

Figure 63 shows how an appropriate convoluted key may be picked in order to compensate for skew between the user's RTC 528 and the producer's RTC 528. A sequence of convolution keys 2862 (a-e) may be generated by using different input values 2881(a-e), each derived from the site ID 2821 and the RTC 528 value plus or minus a differential (e.g., -2 days, -1 days, no delta, +1 days, +2 days). The convolution steps 2871(a-e) are used to generate the sequence of keys 2862(a-e).

Meanwhile, the creator site may use the convolution step 2871(z) based on his RTC 528 value (adjusted to correspond to the intended validity time for the key) to generate a convoluted key 2862(z), which may then be used to generate the content key 2863 in the object's PERC 808. To decrypt the object's content, the user site may use each of its sequence of convolution keys 2862 (a-e) to attempt to generate the master content key 810. When this is attempted, as long as the RTC 538 of the creator site is within acceptable tolerance of the RTC 528 at the user site, one of keys 2862(a-e) will match key 2862(z) and the decryption

will be successful. In this example, matching is determined by validity of decrypted output, not by direct comparison of keys.

Key convolution as described above need not use both site ID and time as a value. Some keys may be generated based on current real time, other keys might be generated on site ID, and still other keys might be generated based on both current real-time and site ID.

Key convolution can be used to provide "time-aged" keys. Such "time-aged" keys provide an automatic mechanism for allowing keys to expire and be replaced by "new" keys. They provide a way to give a user time-limited rights to make time-limited use of an object, or portions of an object, without requiring user re-registration but retaining significant control in the hands of the content provider or administrator. If secure database 610 is sufficiently secure, similar capabilities can be accomplished by checking an expiration date/time associated with a key, but this requires using more storage space for each key or group of keys.

In the preferred embodiment, PERCs 808 can include an expiration date and/or time after which access to the VDE-protected information they correspond to is no longer authorized. Alternatively or in addition, after a duration of time related to

some aspect of the use of the electronic appliance 600 or one or more VDE objects 300, a PERC 808 can force a user to send audit history information to a clearinghouse, distributor, client administrator, or object creator in order to regain or retain the right to use the object(s). The PERC 808 can enforce such time-based restrictions by checking/enforcing parameters that limit key usage and/or availability past time of authorized use. "Time aged" keys may be used to enforce or enhance this type of time-related control of access to VDE protected information.

"Time aged" keys can be used to encrypt and decrypt a set of information for a limited period of time, thus requiring re-registration or the receipt of new permissions or the passing of audit information, without which new keys are not provided for user use. Time aged keys can also be used to improve system security since one or more keys would be automatically replaced based on the time ageing criteria—and thus, cracking secure database 610 and locating one or more keys may have no real value. Still another advantage of using time aged keys is that they can be generated dynamically—thereby obviating the need to store decryption keys in secondary and/or secure memory.

A "time aged key" in the preferred embodiment is not a "true key" that can be used for encryption/decryption, but rather is a piece of information that a PPE 650, in conjunction with

other information, can use to generate a "true key." This other information can be time-based, based on the particular "ID" of the PPE 650, or both. Because the "true key" is never exposed but is always generated within a secure PPE 650 environment, and because secure PPEs are required to generate the "true key," VDE 100 can use "time aged" keys to significantly enhance security and flexibility of the system.

The process of "aging" a key in the preferred embodiment involves generating a time-aged "true key" that is a function of: (a) a "true key," and (b) some other information (e.g., real time parameters, site ID parameters, etc.) This information is combined/transformed (e.g., using the "key convolution" techniques discussed above) to recover or provide a "true key." Since the "true key" can be recovered, this avoids having to store the "true key" within PERC 808, and allow different "true keys" to correspond to the same information within PERC 808. Because the "true key" is not stored in the PERC 808, access to the PERC does not provide access to the information protected by the "true key." Thus, "time aged" keys allows content creators/providers to impose a limitation (e.g., site based and/or time based) on information access that is, in a sense, "external of" or auxiliary to the permissioning provided by one or more PERCs 808. For example, a "time aged" key may enforce an additional time limitation on access to certain protected information, this

additional time limitation being independent of any information or permissioning contained within the PERC 808 and being instead based on one or more time and/or site ID values.

As one example, time-aged decryption keys may be used to allow the purchaser of a "trial subscription" of an electronically published newspaper to access each edition of the paper for a period of one week, after which the decryption keys will no longer work. In this example, the user would need to purchase one or more new PERCs 808, or receive an update to an existing one or more permissions records, to access editions other than the ones from that week. Access to those other editions which might be handled with a totally different pricing structure (e.g., a "regular" subscription rate as opposed to a free or minimal "trial" subscription rate).

In the preferred embodiment, time-aged-based "true keys" can be generated using a one-way or invertible "key convolution" function. Input parameters to the convolution function may include the supplied time-aged keys; user and/or site specific values; a specified portion (e.g., a certain number of high order bits) of the time value from an RTC 528 (if present) or a value derived from such time value in a predefined manner; and a block or record identifier that may be used to ensure that each time aged key is unique. The output of the "key convolution" function

may be a "true key" that is used for decryption purposes until discarded. Running the function with a time-aged key and inappropriate time values typically yields a useless key that will not decrypt.

Generation of a new time aged key can be triggered based on some value of elapsed, absolute or relative time (e.g., based on a real time value from a clock such as RTC 528). At that time, the convolution would produce the wrong key and decryption could not occur until the time-aged key is updated. The criteria used to determine when a new "time aged key" is to be created may itself be changed based on time or some other input variable to provide yet another level of security. Thus, the convolution function and/or the event invoking it may change, shift or employ a varying quantity as a parameter.

One example of the use of time-aged keys is as follows:

- 1) A creator makes a "true" key, and encrypts content with it.
- 2) A creator performs a "reverse convolution" to yield a "time aged key" using, as input parameters to the "reverse convolution":
 - a) the "true" key,

- b) a time parameter (e.g., valid high-order time bits of RTC 528), and
 - c) optional other information (e.g., site ID and/or user ID).
- 3) The creator distributes the "time-aged key" to content users (the creator may also need to distribute the convolution algorithm and/or parameters if she is not using a convolution algorithm already available to the content users' PPE 650).
- 4) The content user's PPE 650 combines:
- a) "time-aged" key
 - b) high-order time bits
 - c) required other information (same as 2c).

It performs a convolution function (i.e., the inverse of "reverse convolution" algorithm in step (2) above) to obtain the "true" key. If the supplied time and/or other information is "wrong," the convolution function will not yield the "true" key, and therefore content cannot be decrypted.

Any of the key blocks associated with VDE objects 300 or other items can be either a regular key block or a time-aged key block, as specified by the object creator during the object

configuration process, or where appropriate, a distributor or client administrator.

"Time aged" keys can also be used as part of protocols to provide secure communications between PPEs 650. For example, instead of providing "true" keys to PPE 650 for communications, VDE 100 may provide only "partial" communication keys to the PPE. These "partial" keys may be provided to PPE 650 during initialization, for example. A predetermined algorithm may produce "true keys" for use to encrypt/decrypt information for secure communications. The predetermined algorithm can "age" these keys the same way in all PPEs 650, or PPEs 650 can be required to contact a VDE administrator at some predetermined time so a new set of partial communications keys can be downloaded to the PPEs. If the PPE 650 does not generate or otherwise obtain "new" partial keys, then it will be disabled from communicating with other PPEs (a further, "fail safe" key may be provided to ensure that the PPE can communicate with a VDE administrator for reinitialization purposes). Two sets of partial keys can be maintained within a PPE 650 to allow a fixed amount of overlap time across all VDE appliances 600. The older of the two sets of partial keys can be updated periodically.

The following additional types of keys (to be discussed below) can also be "aged" in the preferred embodiment:

individual message keys (i.e., keys used for a particular message),
administrative, stationary and travelling object shared keys,
secure database keys, and
private body and content keys.

Initial Installation Key Management

Figure 64 shows the flow of universe-wide, or "master," keys during creating of a PPE 650. In the preferred embodiment, the PPE 650 contains a secure non-volatile key storage 2802 (e.g. SPU 500 non-volatile RAM 534 B or protected storage maintained by HPE 655) that is initialized with keys generated by the manufacturer and by the PPE itself.

The manufacturer possesses (i.e., knows, and protects from disclosure or modification) one or more public key 2811/private key 2812 key pairs used for signing and validating site identification certificates 2821. For each site, the manufacturer generates a site ID 2821 and list of site characteristics 2822. In addition, the manufacturer possesses the public keys 2813, 2814 for validating load modules and initialization code downloads. To enhance security, there may be a plurality of such certification keys, and each PPE 650 may be initialized using only a subset of such keys of each type.

As part of the initialization process, the PPE 650 may generate internally or the manufacturer may generate and supply, one or more pairs of site-specific public keys 2815 and private keys 2816. These are used by the PPE 650 to prove its identity. Similarly, site-specific database key(s) 2817 for the site are generated, and if needed (i.e., if a Random Number Generator 542 is not available), a random initialization seed 2818 is generated.

The initialization may begin by generating site ID 2821 and characteristics 2822 and the site public key 2815/private key 2816 pair(s). These values are combined and may be used to generate one or more site identity certificates 2823. The site identity certificates 2823 may be generated by the public key generation process 2804, and may be stored both in the PPE's protected key storage 2802 and in the manufacturer's VDE site certificate database 2803.

The certification process 2804 may be performed either by the manufacturer or internally to the PPE 650. If performed by the PPE 650, the PPE will temporarily receive the identity certification private key(s) 2812, generate the certificate 2823, store the certificate in local key storage 2802 and transmit it to the manufacturer, after which the PPE 650 must erase its copy of the identity certification private key(s) 2812.

Subsequently, initialization may require generation, by the PPE 650 or by the manufacturer, of site-specific database key(s) 2817 and of site-specific seed value(s) 2818, which are stored in the key storage 2802. In addition, the download certification key(s) 2814 and the load module certification key(s) 2813 may be supplied by the manufacturer and stored in the key storage 2802. These may be used by the PPE 650 to validate all further communications with external entities.

At this point, the PPE 650 may be further initialized with executable code and data by downloading information certified by the load module key(s) 2813 and download key(s) 2814. In the preferred embodiment, these keys may be used to digitally sign data to be loaded into the PPE 650, guaranteeing its validity, and additional key(s) encrypted using the site-specific public key(s) 2815 may be used to encrypt such data and protect it from disclosure.

Installation and Update Key Management

Figure 65 illustrates an example of further key installation either by the manufacturer or by a subsequent update by a VDE administrator. The manufacturer or administrator may supply initial or new values for private header key(s) 2831, external communication key(s) 2832, administrative object keys 2833, or other shared key(s) 2834. These keys may be universe-wide in

the same sense as the global certification keys 2811, 2813, and 2814, or they may be restricted to use within a defined group of VDE instances.

To perform this installation, the installer retrieves the destination site's identity certificate(s) 2823, and from that extracts the site public key(s) 2815. These key(s) may be used in an encryption process 2841 to protect the keys being installed. The key(s) being installed are then transmitted inside the destination site's PPE 650. Inside the PPE 650, the decryption process 2842 may use the site private key(s) 2816 to decrypt the transmission. The PPE 650 then stores the installed or updated keys in its key storage 2802.

Object-Specific Key Use

Figures 66 and 67 illustrate the use of keys in protecting data and control information associated with VDE objects 300.

Figure 66 shows use of a stationary content object 850 whose control information is derived from an administrative object 870. The objects may be received by the PPE 650 (e.g., by retrieval from an object repository 728 over a network or retrieved from local storage). The administrative object decryption process 2843 may use the private header key(s) 2815 to decrypt the administrative object 870, thus retrieving the

PERC 808 governing access to the content object 850. The private body key(s) 810 may then be extracted from the PERC 808 and used by the content decryption process 2845 to make the content available outside the PPE 650. In addition, the database key(s) 2817 may be used by the encryption process 2844 to prepare the PERC for storage outside the PPE 650 in the secure database 610. In subsequent access to the content object 850, the PERC 808 may be retrieved from the secure database 610, decrypted with database key(s) 2817, and used directly, rather than being extracted from administrative object 870.

Figure 67 shows the similar process involving a traveling object 860. The principal distinction between Figures 66 and 67 is that the PERC 808 is stored directly within the traveling object 860, and therefore may be used immediately after the decryption process 2843 to provide a private header key(s) 2831. This private header key 2831 is used to process content within the traveling object 860.

Secret-Key Variations

Figures 64 through 67 illustrate the preferred public-key embodiment, but may also be used to help understand the secret-key versions. In secret-key embodiments, the certification process and the public key encryptions/decryptions are replaced with private-key encryptions, and the public key/private-key

pairs are replaced with individual secret keys that are shared between the PPE 650 instance and the other parties (e.g., the load module supplier(s), the PPE manufacturer). In addition, the certificate generation process 2804 is not performed in secret-key embodiments, and no site identity certificates 2823 or VDE certificate database 2803 exist.

Key Types

The detailed descriptions of key types below further explain secret-key embodiments; this summary is not intended as a complete description. The preferred embodiment PPE 650 can use different types of keys and/or different "shared secrets" for different purposes. Some key types apply to a Public-Key/Secret Key implementation, other keys apply to a Secret Key only

implementation, and still other key types apply to both. The following table lists examples of various key and "shared secret" information used in the preferred embodiment, and where this information is used and stored:

Key/Secret Information Type	Used in PK or Non-PK	Example Storage Location(s)
Master Key(s) (may include some of the specific keys mentioned below)	Both	PPE Manufacturing facility VDE administrator
Manufacturing Key	Both (PK optional)	PPE (PK case) Manufacturing facility
Certification key pair	PK	PPE Certification repository
Public/private key pair	PK	PPE Certification repository (Public Key only)
Initial secret key	Non-PK	PPE
PPE manufacturing ID	Non-PK	PPE
Site ID, shared code, shared keys and shared secrets	Both	PPE
Download authorization key	Both	PPE VDE administrator
External communication keys and other info	Both	PPE Secure Database
Administrative object keys	Both	Permission record
Stationary object keys	Both	Permission record
Traveling object shared keys	Both	Permission record
Secure database keys	Both	PPE
Private body keys	Both	Secure database Some objects
Content keys	Both	Secure database Some objects
Authorization shared secrets	Both	Permission record
Secure Database Back up keys	Both	PPE Secure database

Master Keys

A "master" key is a key used to encrypt other keys. An initial or "master" key may be provided within PPE 650 for communicating other keys in a secure way. During initialization of PPE 650, code and shared keys are downloaded to the PPE. Since the code contains secure convolution algorithms and/or coefficients, it is comparable to a "master key." The shared keys may also be considered "master keys."

If public-key cryptography is used as the basis for external communication with PPE 650, then a master key is required during the PPE Public-key pair certification process. This master key may be, for example, a private key used by the manufacturer or VDE administrator to establish the digital certificate (encrypted public key and other information of the PPE), or it may, as another example, be a private key used by a VDE administrator to encrypt the entries in a certification repository. Once certification has occurred, external communications between PPEs 650 may be established using the certificates of communicating PPEs.

If shared secret keys are used as the basis for external communications, then an initial secret key is required to establish external communications for PPE 650 initialization. This initial secret key is a "master key" in the sense that it is

used to encrypt other keys. A set of shared partial external communications keys (see discussion above) may be downloaded during the PPE initialization process, and these keys are used to establish subsequent external PPE communications.

Manufacturing Key

A manufacturing key is used at the time of PPE manufacture to prevent knowledge by the manufacturing staff of PPE-specific key information that is downloaded into a PPE at initialization time. For example, a PPE 650 that operates as part of the manufacturing facility may generate information for download into the PPE being initialized. This information must be encrypted during communication between the PPEs 650 to keep it confidential, or otherwise the manufacturing staff could read the information. A manufacturing key is used to protect the information. The manufacturing key may be used to protect various other keys downloaded into the PPE such as, for example, a certification private key, a PPE public/private key pair, and/or other keys such as shared secret keys specific to the PPE. Since the manufacturing key is used to encrypt other keys, it is a "master key."

A manufacturing key may be public-key based, or it may be based on a shared secret. Once the information is downloaded, the now-initialized PPE 650 can discard (or simply not use) the

manufacturing key. A manufacturing key may be hardwired into PPE 650 at manufacturing time, or sent to the PPE as its first key and discarded after it is no longer needed. As indicated in the table above and in the preceding discussion, a manufacturing key is not required if PK capabilities are included in the PPE.

Certification Key Pair

A certification key pair may be used as part of a "certification" process for PPEs 650 and VDE electronic appliances 600. This certification process in the preferred embodiment may be used to permit a VDE electronic appliance to present one or more "certificates" authenticating that it (or its key) can be trusted. As described above, this "certification" process may be used by one PPE 650 to "certify" that it is an authentic VDE PPE, it has a certain level of security and capability set (e.g., it is hardware based rather than merely software based), etc. Briefly, the "certification" process may involve using a certificate private key of a certification key pair to encrypt a message including another VDE node's public-key. The private key of a certification key pair is preferably used to generate a PPE certificate. It is used to encrypt a public-key of the PPE. A PPE certificate can either be stored in the PPE, or it may be stored in a certification repository.

Depending on the authentication technique chosen, the public key and the private key of a certification key pair may need to be protected. In the preferred embodiment, the certification public key(s) is distributed amongst PPEs such that they may make use of them in decrypting certificates as an aspect of authentication. Since, in the preferred embodiment, this public key is used inside a PPE 650, there is no need for this public key to be available in plaintext, and in any event it is important that such key be maintained and transmitted with integrity (e.g., during initialization and/or update by a VDE administrator). If the certification public key is kept confidential (i.e., only available in plaintext inside the PPE 650), it may make cracking security much more difficult. The private key of a certification key pair should be kept confidential and only be stored by a certifying authority (i.e., should not be distributed).

In order to allow, in the preferred embodiment, the ability to differentiate installations with different levels/degrees of trustedness/security, different certification key pairs may be used (e.g., different certification keys may be used to certify SPEs 503 then are used to certify HPEs 655).

PPE Public/Private Key Pair

In the preferred embodiment, each PPE 650 may have its own unique "device" (and/or user) public/private key pair.

Preferably, the private key of this key pair is generated within the PPE and is never exposed in any form outside of the PPE. Thus, in one embodiment, the PPE 650 may be provided with an internal capability for generating key pairs internally. If the PPE generates its own public-key crypto-system key pairs internally, a manufacturing key discussed above may not be needed. If desired, however, for cost reasons a key pair may be exposed only at the time a PPE 650 is manufactured, and may be protected at that time using a manufacturing key. Allowing PPE 650 to generate its public key pair internally allows the key pair to be concealed, but may in some applications be outweighed by the cost of putting a public-key key pair generator into PPE 650.

Initial Secret Key

The initial secret key is used as a master key by a secret key only based PPE 650 to protect information downloaded into the PPE during initialization. It is generated by the PPE 650, and is sent from the PPE to a secure manufacturing database encrypted using a manufacturing key. The secure database sends back a unique PPE manufacturing ID encrypted using the initial secret key in response.

The initial secret key is likely to be a much longer key than keys used for "standard" encryption due to its special role in PPE initialization. Since the resulting decryption overhead occurs

only during the initialization process, multiple passes through the decryption hardware with selected portions of this key are tolerable.

PPE Manufacturing ID

The PPE manufacturing ID is not a "key," but does fall within the classic definition of a "shared secret." It preferably uniquely identifies a PPE 650 and may be used by the secure database 610 to determine the PPE's initial secret key during the PPE initialization process.

Site ID, Shared Code, Shared Keys and Shared Secrets

The VDE site ID along with shared code, keys and secrets are preferably either downloaded into PPE 650 during the PPE initialization process, or are generated internally by a PPE as part of that process. In the preferred embodiment, most or all of this information is downloaded.

The PPE site ID uniquely identifies the PPE 650. The site ID is preferably unique so as to uniquely identify the PPE 650 and distinguish that PPE from all other PPEs. The site ID in the preferred embodiment provides a unique address that may be used for various purposes, such as for example to provide "address privacy" functions. In some cases, the site ID may be the public key of the PPE 650. In other cases, the PPE site ID

may be assigned during the manufacturing and/or initialization process. In the case of a PPE 650 that is not public-key-capable, it would not be desirable to use the device secret key as the unique site ID because this would expose too many bits of the key—and therefore a different information string should be used as the site ID.

Shared code comprises those code fragments that provide at least a portion of the control program for the PPE 650. In the preferred embodiment, a basic code fragment is installed during PPE manufacturing that permits the PPE to bootstrap and begin the initialization process. This fragment can be replaced during the initialization process, or during subsequent download processing, with updated control logic.

Shared keys may be downloaded into PPE 650 during the initialization process. These keys may be used, for example, to decrypt the private headers of many object structures.

When PPE 650 is operating in a secret key only mode, the initialization and download processes may import shared secrets into the PPE 650. These shared secrets may be used during communications processes to permit PPEs 650 to authenticate the identity of other PPEs and/or users.

Download Authorization Key

The download authorization key is received by PPE 650 during the initialization download process. It is used to authorize further PPE 650 code updates, key updates, and may also be used to protect PPE secure database 610 backup to allow recovery by a VDE administrator (for example) if the PPE fails. It may be used along with the site ID, time and convolution algorithm to derive a site ID specific key. The download authorization key may also be used to encrypt the key block used to encrypt secure database 610 backups. It may also be used to form a site specific key that is used to enable future downloads to the PPE 650. This download authorization key is not shared among all PPEs 650 in the preferred embodiment; it is specific to functions performed by authorized VDE administrators.

External Communications Keys and Related Secret and Public Information

There are several cases where keys are required when PPEs 650 communicate. The process of establishing secure communications may also require the use of related public and secret information about the communicating electronic appliances 600. The external communication keys and other information are used to support and authenticate secure communications. These keys comprise a public-key pair in the

preferred embodiment although shared secret keys may be used alternatively or in addition.

Administrative Object Keys

In the preferred embodiment, an administrative object shared key may be used to decrypt the private header of an administrative object 870. In the case of administrative objects, a permissions record 808 may be present in the private header. In some cases, the permissions record 808 may be distributed as (or within) an administrative object that performs the function of providing a right to process the content of other administrative objects. The permissions record 808 preferably contains the keys for the private body, and the keys for the content that can be accessed would be budgets referenced in that permissions record 808. The administrative object shared keys may incorporate time as a component, and may be replaced when expired.

Stationary Object Keys

A stationary object shared key may be used to decrypt a private header of stationary objects 850. As explained above, in some cases a permissions record 808 may be present in the private header of stationary objects. If present, the permissions record 808 may contain the keys for the private body but will not contain the keys for the content. These shared keys may

incorporate time as a component, and may be replaced when expired.

Traveling Object Shared Keys

A traveling object shared key may be used to decrypt the private header of traveling objects 860. In the preferred embodiment, traveling objects contain permissions record 808 in their private headers. The permissions record 808 preferably contains the keys for the private body and the keys for the content that can be accessed as permitted by the permissions record 808. These shared keys may incorporate time as a component, and may be replaced when expired.

Secure Database Keys

PPE 650 preferably generates these secure database keys and never exposes them outside of the PPE. They are site-specific in the preferred embodiment, and may be "aged" as described above. As described above, each time an updated record is written to secure database 610, a new key may be used and kept in a key list within the PPE. Periodically (and when the internal list has no more room), the PPE 650 may generate a new key to encrypt new or old records. A group of keys may be used instead of a single key, depending on the size of the secure database 610.

Private Body Keys

Private body keys are unique to an object 300, and are not dependent on key information shared between PPEs 650. They are preferably generated by the PPE 650 at the time the private body is encrypted, and may incorporate real-time as a component to "age" them. They are received in permissions records 808, and their usage may be controlled by budgets.

Content Keys

Content Keys are unique to an object 300, and are not dependent on key information shared between PPEs 650. They are preferably generated by the PPE 650 at the time the content is encrypted. They may incorporate time as a component to "age" them. They are received in permissions records 808, and their usage may be controlled by budgets.

Authorization Shared Secrets

Access to and use of information within a PPE 650 or within a secure database 610 may be controlled using authorization "shared secrets" rather than keys. Authorization shared secrets may be stored within the records they authorize (permissions records 808, budget records, etc.). The authorization shared secret may be formulated when the corresponding record is created. Authorization shared secrets can be generated by an authorizing PPE 650, and may be

replaced when record updates occur. Authorization shared secrets have some characteristics associated with "capabilities" used in capabilities based operating systems. Access tags (described below) are an important set of authorization shared secrets in the preferred embodiment.

Backup Keys

As described above, the secure database 610 backup consists of reading all secure database records and current audit "roll ups" stored in both PPE 650 and externally. Then, the backup process decrypts and re-encrypts this information using a new set of generated keys. These keys, the time of the backup, and other appropriate information to identify the backup, may be encrypted multiple times and stored with the previously encrypted secure database files and roll up data within the backup files. These files may then all be encrypted using a "backup key" that is generated and stored within PPE 650. This backup key 500 may be used by the PPE to recover a backup if necessary. The backup keys may also be securely encrypted (e.g., using a download authentication key and/or a VDE administrator public key) and stored within the backup itself to permit a VDE administrator to recover the backup in case of PPE 650 failure.

Cryptographic Sealing

Sealing is used to protect the integrity of information when it may be subjected to modifications outside the control of the PPE 650, either accidentally or as an attack on the VDE security. Two specific applications may be the computation of check values for database records and the protection of data blocks that are swapped out of an SPE 500.

There are two types of sealing: keyless sealing, also known as cryptographic hashing, and keyed sealing. Both employ a cryptographically strong hash function, such as MD5 or SHA. Such a function takes an input of arbitrary size and yields a fixed-size hash, or "digest." The digest has the property that it is infeasible to compute two inputs that yield the same digest, and infeasible to compute one input that yields a specific digest value, where "infeasible" is with reference to a work factor based on the size of the digest value in bits. If, for example, a 256-bit hash function is to be called strong, it must require approximately on average 10^{38} (2^{128}) trials before a duplicated or specified digest value is likely to be produced.

Keyless seals may be employed as check values in database records (e.g., in PERC 808) and similar applications. A keyless seal may be computed based on the content of the body of the record, and the seal stored with the rest of the record. The

combination of seal and record may be encrypted to protect it in storage. If someone modifies the encrypted record without knowing the encryption key (either in the part representing the data or the part representing the seal), the decrypted content will be different, and the decrypted check value will not match the digest computed from the record's data. Even though the hash algorithm is known, it is not feasible to modify both the record's data and its seal to correspond because both are encrypted.

Keyed seals may be employed as protection for data stored outside a protected environment without encryption, or as a validity proof between two protected environments. A keyed seal is computed similarly to a keyless seal, except that a secret initial value is logically prefixed to the data being sealed. The digest value thus depends both on the secret and the data, and it is infeasible to compute a new seal to correspond to modified data even though the data itself is visible to an attacker. A keyed seal may protect data in storage with a single secret value, or may protect data in transit between two environments that share a single secret value.

The choice of keys or keyless seals depends on the nature of the data being protected and whether it is additionally protected by encryption.

Tagging

Tagging is particularly useful for supporting the secure storage of important component assembly and related information on secondary storage memory 652. Integrated use of information "tagging" and encryption strategies allows use of inexpensive mass storage devices to securely store information that, in part enables, limits and/or records the configuration, management and operation of a VDE node and the use of VDE protected content.

When encrypted or otherwise secured information is delivered into a user's secure VDE processing area (e.g., PPE 650), a portion of this information can be used as a "tag" that is first decrypted or otherwise unsecured and then compared to an expected value to confirm that the information represents expected information. The tag thus can be used as a portion of a process confirming the identity and correctness of received, VDE protected, information.

Three classes of tags that may be included in the control structures of the preferred embodiment:

- access tags
- validation tags
- correlation tags.

These tags have distinct purposes.

An access tag may be used as a "shared secret" between VDE protected elements and entities authorized to read and/or modify the tagged element(s). The access tag may be broken into separate fields to control different activities independently. If an access tag is used by an element such as a method core 1000', administrative events that affect such an element must include the access tag (or portion of the access tag) for the affected element(s) and assert that tag when an event is submitted for processing. If access tags are maintained securely (e.g., created inside a PPE 650 when the elements are created, and only released from PPE 650 in encrypted structures), and only distributed to authorized parties, modification of structures can be controlled more securely. Of course, control structures (e.g., PERCs 808) may further limit or qualify modifications or other actions expressed in administrative events.

Correlation tags are used when one element references another element. For example, a creator might be required by a budget owner to obtain permission and establish a business relationship prior to referencing their budget within the creator's PERCs. After such relationship was formed, the budget owner might transmit one or more correlation tags to the creator as one aspect of allowing the creator to produce PERCs that reference the budget owner's budget.

Validation tags may be used to help detect record substitution attempts on the part of a tamperer.

In some respects, these three classes of tags overlap in function. For example, a correlation tag mismatch may prevent some classes of modification attempts that would normally be prevented by an access tag mismatch before an access tag check is performed. The preferred embodiment may use this overlap in some cases to reduce overhead by, for example, using access tags in a role similar to validation tags as described above.

In general, tagging procedures involve changing, within SPE 503, encryption key(s), securing techniques(s), and/or providing specific, stored tag(s). These procedures can be employed with secure database 610 information stored on said inexpensive mass storage 652 and used within a hardware SPU 500 for authenticating, decrypting, or otherwise analyzing, using and making available VDE protected content and management database information. Normally, changing validation tags involves storing within a VDE node hardware (e.g., the PPE 650) one or more elements of information corresponding to the tagging changes. Storage of information outside of the hardware SPE's physically secure, trusted environment is a highly cost savings means of secure storage, and the security of important stored management database information is enhanced by this tagging of

information. Performing this tagging "change" frequently (for example, every time a given record is decrypted) prevents the substitution of "incorrect" information for "correct" information, since said substitution will not carry information which will match the tagging information stored within the hardware SPE during subsequent retrieval of the information.

Another benefit of information tagging is the use of tags to help enforce and/or verify information and/or control mechanisms in force between two or more parties. If information is tagged by one party, and then passed to another party or parties, a tag can be used as an expected value associated with communications and/or transactions between the two parties regarding the tagged information. For example, if a tag is associated with a data element that is passed by Party A to Party B, Party B may require Party A to prove knowledge of the correct value of at least a portion of a tag before information related to, and/or part of, said data element is released by Party B to Party A, or vice versa. In another example, a tag may be used by Party A to verify that information sent by Party B is actually associated with, and/or part of, a tagged data element, or vice versa.

Establishing A Secure, Authenticated, Communication Channel

From time to time, two parties (e.g., PPEs A and B), will need to establish a communication channel that is known by both

parties to be secure from eavesdropping, secure from tampering, and to be in use solely by the two parties whose identifies are correctly known to each other.

The following describes an example process for establishing such a channel and identifies how the requirements for security and authentication may be established and validated by the parties. The process is described in the abstract, in terms of the claims and belief each party must establish, and is not to be taken as a specification of any particular protocol. In particular, the individual sub-steps of each step are not required to be implemented using distinct operations; in practice, the establishment and validation of related proofs is often combined into a single operation.

The sub-steps need not be performed in the order detailed below, except to the extent that the validity of a claim cannot be proven before the claim is made by the other party. The steps may involve additional communications between the two parties than are implied by the enumerated sub-steps, as the "transmission" of information may itself be broken into sub-steps. Also, it is not necessary to protect the claims or the proofs from disclosure or modification during transmission. Knowledge of the claims (including the specific communication proposals and acknowledgements thereof) is not considered protected

information. Any modification of the proofs will cause the proofs to become invalid and will cause the process to fail.

Standard public-key or secret-key cryptographic techniques can be used to implement this process (e.g., X.509, Authenticated Diffie-Hellman, Kerberos). The preferred embodiment uses the three-way X.509 public key protocol steps.

The following may be the first two steps in the example process:

A. (*precursor step*): Establish means of creating validatable claims by A

B. (*precursor step*): Establish means of creating validatable claims by B

These two steps ensure that each party has a means of making claims that can be validated by the other party, for instance, by using a public key signature scheme in which both parties maintain a private key and make available a public key that itself is authenticated by the digital signature of a certification authority.

The next steps may be:

A (proposal step):

1. Determine B's identity

2. Acquire means of validating claims made by B
3. Create a unique identity for this specific proposed communication
4. Create a communication proposal identifying the parties and the specific communication
5. Create validatable proof of A's identity and the origin of the communication proposal
6. Deliver communication proposal and associated proof to B.

These steps establish the identity of the correspondent party B and proposes a communication. Because establishment of the communication will require validation of claims made by B, a means must be provided for A to validate such claims. Because the establishment of the communication must be unique to a specific requirement by A for communication, this communication proposal and all associated traffic must be unambiguously distinguishable from all other such traffic. Because B must validate the proposal as a legitimate proposal from A, a proof must be provided that the proposal is valid.

The next steps may be as follows:

B (acknowledgement step):

1. Extract A's identity from the communication proposal
2. Acquire means of validating claims made by A
3. Validate A's claim of identity and communication proposal origin
4. Determine the unique identification of the communication proposal
5. Determine that the communication proposal does not duplicate an earlier proposal
6. Create an acknowledgement identifying the specific communication proposal
7. Create validatable proof of B's identity and the origin of the acknowledgement
8. Deliver the acknowledgement and associated proof to A.

These steps establish that party B has received A's communication proposal and is prepared to act on it. Because B must validate the proposal, B must first determine its origin and validate its authenticity. B must ensure that its response is associated with a specific proposal, and that the proposal is not a replay. If B accepts the proposal, it must prove both B's own identity and that B has received a specific proposal.

The next steps may be:

A (establishment step):

1. Validate B's claim acknowledgement of A's specific proposal
2. Extract the identity of the specific communication proposal from the acknowledgement
3. Determine that the acknowledgement is associated with an outstanding communication proposal
4. Create unique session key to be used for the proposed communication
5. Create proof of session key's creation by A
6. Create proof of session key's association with the specific communication proposal
7. Create proof of receipt of B's acknowledgement
8. Protect the session key from disclosure in transmission
9. Protect the session key from modification in transmission
10. Deliver protected session key and all proofs to B.

These steps allows A to specify a session key to be associated with all further traffic related to A's specific communication proposal. A must create the key, prove that A created it, and prove that it is associated with the specific proposed communication. In addition, A must prove that the

session key is generated in response to B's acknowledgement of the proposal. The session key must be protected from disclosure of modification to ensure that an attacker cannot substitute a different value.

Transportability of VDE Installations Between PPEs 650

In a preferred embodiment, VDE objects 300 and other secure information may if appropriate, be transported from one PPE 650 to another securely using the various keys outlined above. VDE 100 uses redistribution of VDE administrative information to exchange ownership of VDE object 300, and to allow the portability of objects between electronic appliances 600.

The permissions record 808 of VDE objects 300 contains rights information that may be used to determine whether an object can be redistributed in whole, in part, or at all. If a VDE object 300 can be redistributed, then electronic appliance 600 normally must have a "budget" and/or other permissioning that allows it to redistribute the object. For example, an electronic appliance 600 authorized to redistribute an object may create an administrative object containing a budget or rights less than or equal to the budget or rights that it owns. Some administrative objects may be sent to other PPEs 650. A PPE 650 that receives one of the administrative objects may have the ability to use at least a portion of the budgets, or rights, to related objects.

Transfer of ownership of a VDE object 300 is a special case in which all of the permissions and/or budgets for a VDE object are redistributed to a different PPE 650. Some VDE objects may require that all object-related information be delivered (e.g., it's possible to "sell" all rights to the object). However, some VDE objects 300 may prohibit such a transfer. In the case of ownership transfer, the original providers for a VDE object 300 may need to be contacted by the new owner, informed of the transfer, and validated using an authorization shared secret that accompanies reauthorization, before transfer of ownership can be completed.

When an electronic appliance 600 receives a component assembly, an encrypted part of the assembly may contain a value that is known only to the party or PPE 650 that supplied the assembly. This value may be saved with information that must eventually be returned to the assembly supplier (e.g., audit, billing and related information). When a component supplier requests that information be reported, the value may be provided by the supplier so that the local electronic appliance 600 can check it against the originally supplied value to ensure that the request is legitimate. When a new component is received, the value may be checked against an old component to determine whether the new component is legitimate (e.g., the new value for use in the next report process may be included with the new component).

Integrity of VDE Security

There are many ways in which a PPE 650 might be compromised. The goal of the security provided by VDE 100 is to reduce the possibility that the system will be compromised, and minimize the adverse effects if it is compromised.

The basic cryptographic algorithms that are used to implement VDE 100 are assumed to be safe (cryptographically strong). These include the secret-key encryption of content, public-key signatures for integrity verification, public-key encryption for privacy between PPEs 650 or between a PPE and a VDE administrator, etc. Direct attack on these algorithms is assumed to be beyond the capabilities of an attacker. For domestic versions of VDE 100 some of this is probably a safe assumption since the basic building blocks for control information have sufficiently long keys and are sufficiently proven.

The following risks of threat or attacks may be significant:

- Unauthorized creation or modification of component assemblies (e.g., budgets)
- Unauthorized bulk disclosure of content
- Compromise of one or more keys
- Software emulation of a hardware PPE
- Substitution of older records in place of newer records

- Introduction of "rogue" (i.e., unauthentic) load modules
- Replay attacks
- Defeating "fingerprinting"
- Unauthorized disclosure of individual content items
- Redistribution of individual content items.

A significant potential security breach would occur if one or more encryption keys are compromised. As discussed above, however, the encryption keys used by VDE 100 are sufficiently varied and compartmentalized so that compromising one key would have only limited value to an attacker in most cases. For example, if a certification private key is exposed, an attacker could pass the challenge/response protocol as discussed above but would then confront the next level of security that would entail cracking either the initialization challenge/response or the external communication keys. If the initialization challenge/response security is also defeated, the initialization code and various initialization keys would also be exposed. However, it would still be necessary to understand the code and data to find the shared VDE keys and to duplicate the key-generation ("convolution") algorithms. In addition, correct real time clock values must be maintained by the spoof. If the attacker is able to accomplish all of this successfully, then all secure communications to the bogus PPE would be compromised.

An object would be compromised if communications related to the permissions record 808 of that object are sent to the bogus PPE.

Knowledge of the PPE download authorization key and the algorithms that are used to derive the key that encrypts the keys for backup of secured database 610 would compromise the entire secured database at a specific electronic appliance 600. However, in order to use this information to compromise content of VDE objects 300, an understanding of appropriate VDE internals would also be required. In a preferred embodiment, the private body keys and content keys stored in a secured database 610 are "aged" by including a time component. Time is convoluted with the stored values to derive the "true keys" needed to decrypt content. If this process is also compromised, then object content or methods would be revealed. Since a backup of secured database 610 is not ever restored to a PPE 650 in the preferred embodiment without the intervention of an authorized VDE administrator, a "bogus" PPE would have to be used to make use of this information.

External communication shared keys are used in the preferred embodiment in conjunction with a key convolution algorithm based on site ID and time. If compromised, all of the steps necessary to allow communications with PPEs 650 must also be known to take advantage of this knowledge. In addition,

at least one of the administrative object shared keys must be compromised to gain access to a decrypted permissions record 808.

Compromising an administrative object shared key has no value unless the "cracker" also has knowledge of external communication keys. All administrative objects are encrypted by unique keys exchanged using the shared external communications keys, site ID and time. Knowledge of PPE 650 internal details would be necessary to further decrypt the content of administrative objects.

The private header of a stationary object (or any other stationary object that uses the same shared key) if compromised, may provide the attacker with access to content until the shared key "ages" enough to no longer decrypt the private header. Neither the private body nor the content of the object is exposed unless a permissions record 808 for that object is also compromised. The private headers of these objects may remain compromised until the key "ages" enough to no longer decrypt the private header.

Secure database encryption keys in the preferred embodiment are frequently changing and are also site specific. The consequences of compromising a secured database 610 file or

a record depends on the information that has been compromised. For example, permissions record 808 contain keys for the public body and content of a VDE object 300. If a permissions record 808 is compromised, the aspects of that object protected by the keys provided by the permissions record are also compromised—if the algorithm that generates the "true keys" is also known. If a private body key becomes known, the private body of the object is compromised until the key "ages" and expires. If the "aging" process for that key is also compromised, the breach is permanent. Since the private body may contain methods that are shared by a number of different objects, these methods may also become compromised. When the breach is detected, all administrative objects that provide budgets and permissions record should update the compromised methods. Methods stored in secure database 610 are only replaced by more recent versions, so the compromised version becomes unusable after the update is completed.

If a content key becomes compromised, the portion of the content encrypted with the key is also compromised until the key "ages" and expires. If the "aging" process for that key also becomes compromised, then the breach becomes permanent. If multiple levels of encryption are used, or portions of the content are encrypted with different keys, learning a single key would be insufficient to release some or all of the content.

If an authorization shared secret (e.g., an access tag) becomes known, the record containing the secret may be modified by an authorized means if the "cracker" knows how to properly use the secret. Generally speaking, the external communications keys, the administrative object keys and the management file keys must also be "cracked" before a shared secret is useful. Of course, any detailed knowledge of the protocols would also be required to make use of this information.

In the preferred embodiment, PPE 650 may detect whether or not it has become compromised. For example, by comparing information stored in an SPE 503 (e.g., summary service information) with information stored in secure database 610 and/or transmitted to a VDE participant (e.g., a VDE clearinghouse), discrepancies may become evident. If PPE 650 (or a VDE administrator watching its activities or communicating with it) detects that it has been compromised, it may be updated with an initialization to use new code, keys and new encryption/decryption algorithms. This would limit exposure to VDE objects 300 that existed at the time the encryption scheme was broken. It is possible to require the PPE 650 to cease functioning after a certain period of time unless new code and key downloads occur. It is also possible to have VDE administrators force updates to occur. It is also likely that the

desire to acquire a new VDE object 300 will provide an incentive for users to update their PPEs 650 at regular time intervals.

Finally, the end-to-end nature of VDE applications, in which content 108 flows in one direction, generating reports and bills 118 in the other, makes it possible to perform "back-end" consistency checks. Such checks, performed in clearinghouses 116, can detect patterns of use that may or do indicate fraud (e.g., excessive acquisition of protected content without any corresponding payment, usage records without corresponding billing records). The fine grain of usage reporting and the ready availability of usage records and reports in electronic form enables sophisticated fraud detection mechanisms to be built so that fraud-related costs can be kept to an acceptable level.

PPE Initialization

Each PPE 650 needs to be initialized before it can be used. Initialization may occur at the manufacturer site, after the PPE 650 has been placed out in the field, or both. The manufacturing process for PPE 650 typically involves embedding within the PPE sufficient software that will allow the device to be more completely initialized at a later time. This manufacturing process may include, for example, testing the bootstrap loader and challenge-response software permanently stored within PPE 650, and loading the PPE's unique ID. These steps provide a

basic VDE-capable PPE 650 that may be further initialized (e.g., after it has been installed within an electronic appliance 600 and placed in the field). In some cases, the manufacturing and further initialization processes may be combined to produce "VDE ready" PPEs 650. This description elaborates on the summary presented above with respect to Figures 64 and 65.

Figure 68 shows an example of steps that may be performed in accordance with one preferred embodiment to initialize a PPE 650. Some of the steps shown in this flowchart may be performed at the manufacturing site, and some may be performed remotely through contact between a VDE administrator and the PPE 650. Alternatively, all of the steps shown in the diagram may be performed at the manufacturing site, or all of the steps shown may be performed through remote communications between the PPE 500 and a VDE administrator.

If the initialization process 1370 is being performed at the manufacturer, PPE 650 may first be attached to a testbed. The manufacturing testbed may first reset the PPE 650 (e.g., with a power on clear) (Block 1372). If this reset is being performed at the manufacturer, then the PPE 650 preferably executes a special testbed bootstrap code that completely tests the PPE operation from a software standpoint and fails if something is wrong with the PPE. A secure communications exchange may

then be established between the manufacturing testbed and the PPE 650 using an initial challenge-response interaction (Block 1374) that is preferably provided as part of the testbed bootstrap process. Once this secure communications has been established, the PPE 650 may report the results of the bootstrap tests it has performed to the manufacturing testbed. Assuming the PPE 650 has tested successfully, the manufacturing testbed may download new code into the PPE 650 to update its internal bootstrap code (Block 1376) so that it does not go through the testbed bootstrap process upon subsequent resets (Block 1376). The manufacturing testbed may then load new firmware into the PPE internal non-volatile memory in order to provide additional standard and/or customized capabilities (Block 1378). For example, the manufacturing testbed may preload PPE 650 with the load modules appropriate for the particular manufacturing lot. This step permits the PPE 500 to be customized at the factory for specific applications.

The manufacturing testbed may next load a unique device ID into PPE 650 (Block 1380). PPE 650 now carries a unique ID that can be used for further interactions.

Blocks 1372-1380R typically are, in the preferred embodiment, performed at the manufacturing site. Blocks 1374

and 1382-1388 may be performed either at the manufacturing site, after the PPE 650 has been deployed, or both.

To further initialize PPE 650, once a secure communications has been established between the PPE and the manufacturing testbed or a VDE administrator (Block 1374), any required keys, tags or certificates are loaded into PPE 650 (Block 1382). For example, the manufacturing test bed may load its information into PPE 650 so the PPE may be initialized at a later time. Some of these values may be generated internally within PPE 650. The manufacturing testbed or VDE administrator may then initialize the PPE real time clock 528 to the current real time value (Block 1384). This provides a time and date reference for the PPE 650. The manufacturing testbed or the VDE administrator may next initialize the summary values maintained internally to the PPE 500 (Block 1386). If the PPE 650 is already installed as part of an electronic appliance 600, the PPE may at this point initialize its secure database 610 (Block 1388).

Figure 69 shows an example of program control steps performed by PPE 650 as part of a firmware download process (See Figure 68, Block 1378). The PPE download process is used to load externally provided firmware and/or data elements into the PPE. Firmware loads may take two forms: permanent loads

for software that remains resident in the PPE 650; and transient loads for software that is being loaded for execution. A related process for storing into the secure database 610 is performed for elements that have been sent to a VDE electronic appliance 600.

PPE 650 automatically performs several checks to ensure that firmware being downloaded into the PPE has not been tampered with, replaced, or substituted before it was loaded. The download routine 1390 shown in the figure illustrates an example of such checks. Once the PPE 650 has received a new firmware item (Block 1392), it may check the item to ensure that it decrypts properly using the predetermined download or administrative object key (depending on the source of the element) (decision Block 1394). If the firmware decrypts properly ("yes" exits to decision Block 1394), the firmware as check valve may be calculated and compared against the check valve stored under the encryption wrapper of the firmware (decision Block 1396). If the two check summed values compare favorably ("yes" exit to decision Block 1396), then the PPE 650 may compare the public and private header identification tags associated with the firmware to ensure that the proper firmware was provided and had not been substituted (step not shown in the figure). Assuming this test also passes, the PPE 500 may calculate the digital signatures of the firmware (assuming digital signatures are supported by the PPE 650 and the firmware is "signed") and

may check the calculated signature to ensure that it compares favorably to the digital signatures under the firmware encryption wrapper (Blocks 1398, 1400). If any of these tests fail, then the download will be aborted ("fail" termination 1401).

Assuming all of the tests described above pass, then PPE 650 determines whether the firmware is to be stored within the PPE (e.g., an internal non-volatile memory), or whether it is to be stored in the secure database 610 (decision Block 1402). If the firmware is to be stored within the PPE ("yes" exit to decision Block 1402), then the PPE 500 may simply store the information internally (Block 1404). If the firmware is to be stored within the secure database 610 ("no" exit to decision Block 1402), then the firmware may be tagged with a unique PPE-specific tag designed to prevent record substitution (Block 1406), and the firmware may then be encrypted using the appropriate secure database key and released to the secure database 610 (Block 1408).

Networking SPUs 500 and/or VDE Electronic Appliances 600

In the context of many computers interconnected by a local or wide area network, it would be possible for one or a few of them to be VDE electronic appliances 600. For example, a VDE-capable server might include one or more SPUs 500. This centralized VDE server could provide all VDE services required within the network or it can share VDE service with VDE server

nodes; that is, it can perform a few, some, or most VDE service activities. For example, a user's non-VDE computer could issue a request over the network for VDE-protected content. In response to the request, the VDE server could comply by accessing the appropriate VDE object 300, releasing the requested content and delivering the content over the network 672 to the requesting user. Such an arrangement would allow VDE capabilities to be easily integrated into existing networks without requiring modification or replacement of the various computers and other devices connected to the networks.

For example, a VDE server having one or more protected processing environments 650 could communicate over a network with workstations that do not have a protected processing environment. The VDE server could perform all secure VDE processing, and release resulting content and other information to the workstations on the network. This arrangement would require no hardware or software modification to the workstations.

However, some applications may require greater security, flexibility and/or performance that may be obtained by providing multiple VDE electronic appliances 600 connected to the same network 672. Because commonly-used local area networks constitute an insecure channel that may be subject to tampering

and/or eavesdropping, it is desirable in most secure applications to protect the information communicated across the network. It would be possible to use conventional network security techniques to protect VDE-released content or other VDE information communicated across a network 672 between a VDE electronic appliance 600 and a non-VDE electronic appliance. However, advantages are obtained by providing multiple networked VDE electronic appliances 600 within the same system.

As discussed above in connection with Figure 8, multiple VDE electronic appliances 600 may communicate with one another over a network 672 or other communications path. Such networking of VDE electronic appliances 600 can provide advantages. Advantages include, for example, the possibility of centralizing VDE-resources, storing and/or archiving metering information on a server VDE and delivering information and services efficiently across the network 672 to multiple electronic appliances 600.

For example, in a local area network topology, a "VDE server" electronic appliance 600 could store VDE-protected information and make it available to one or more additional electronic appliances 600 or computers that may communicate with the server over network 672. As one example, an object

repository 728 storing VDE objects could be maintained at the centralized server, and each of many networked electronic appliance 600 users could access the centralized object repository over the network 672 as needed. When a user needs to access a particular VDE object 300, her electronic appliance 600 could issue a request over network 672 to obtain a copy of the object. The "VDE server" could deliver all or a portion of the requested object 300 in response to the request. Providing such a centralized object repository 728 would have the advantage of minimizing mass storage requirements local to each electronic appliance 600 connected to the network 672, eliminate redundant copies of the same information, ease information management burdens, provide additional physical and/or other security for particularly important VDE processes and/or information occurring at the server, where providing such security at VDE nodes may be commercially impractical for certain business models, etc.

It may also be desirable to centralize secure database 610 in a local area network topology. For example, in the context of a local area network, a secure database 610 server could be provided at a centralized location. Each of several electronic appliances 600 connected to a local area network 672 could issue requests for secure database 610 records over the network, and receive those records via the network. The records could be

provided over the network in encrypted form. "Keys" needed to decrypt the records could be shared by transmitting them across the network in secure communication exchanges. Centralizing secure database 610 in a network 672 has potential advantages of minimizing or eliminating secondary storage and/or other memory requirements for each of the networked electronic appliances 600, avoiding redundant information storage, allowing centralized backup services to be provided, easing information management burdens, etc.

One way to inexpensively and conveniently deploy multiple instances of VDE electronic appliances 600 across a network would be to provide network workstations with software defining an HPE 655. This arrangement requires no hardware modification of the workstations; an HPE 655 can be defined using software only. An SPE(s) 503 and/or HPE(s) 655 could also be provided within a VDE server. This arrangement has the advantage of allowing distributed VDE network processing without requiring workstations to be customized or modified (except for loading a new program(s) into them). VDE functions requiring high levels of security may be restricted to an SPU-based VDE server. "Secure" HPE-based workstations could perform VDE functions requiring less security, and could also coordinate their activities with the VDE server.

Thus, it may be advantageous to provide multiple VDE electronic appliances 600 within the same network. It may also be advantageous to provide multiple VDE electronic appliances 600 within the same workstation or other electronic appliance 600. For example, an electronic appliance 600 may include multiple electronic appliances 600 each of which have a SPU 500 and are capable of performing VDE functions.

For example, one or more VDE electronic appliances 600 can be used as input/output device(s) of a computer system. This may eliminate the need to decrypt information in one device and then move it in unencrypted form across some bus or other unsecured channel to another device such as a peripheral. If the peripheral device itself is a VDE electronic appliance 600 having a SPU 500, VDE-protected information may be securely sent to the peripheral across the insecure channel for processing (e.g., decryption) at the peripheral device. Giving the peripheral device the capability of handling VDE-protected information directly also increases flexibility. For example, the VDE electronic appliance 600 peripheral device may control VDE object 300 usage. It may, for example, meter the usage or other parameters associated with the information it processes, and it may gather audit trails and other information specific to the processing it performs in order to provide greater information gathering about VDE object usage. Providing multiple

cooperating VDE electronic appliances 600 may also increase performance by eliminating the need to move encrypted information to a VDE electronic appliance 600 and then move it again in unencrypted form to a non-VDE device. The VDE-protected information can be moved directly to its destination device which, if VDE-capable, may directly process it without requiring involvement by some other VDE electronic appliance 600.

Figure 70 shows an example of an arrangement 2630 comprising multiple VDE electronic appliances 600(1), 600(2), 600(3), . . . , 600(N). VDE electronic appliances 600(1) . . . 600(N) may communicate with one another over a communications path 2631 (e.g., the system bus of a work station, a telephone or other wire, a cable, a backplane, a network 672, or any other communications mechanism). Each of the electronic appliances 600 shown in the figure may have the same general architecture shown in Figure 8, i.e., they may each include a CPU (or microcontroller) 654, SPU 500, RAM 656, ROM 658, and system bus 653. Each of the electronic appliances 600 shown in the figure may have an interface/controller 2632 (which may be considered to be a particular kind of I/O controller 660 and/or communications controller 666 shown in Figure 8). This interface/controller 2632 provides an interface between the electronic appliance system bus 653 and an appropriate electrical

connector 2634. Electrical connectors 2634 of each of the respective electronic appliances 600(1), . . . 600(N) provide a connection to a common network 672 or other communication paths.

Although each of electronic appliances 600 shown in the figure may have a generally similar architecture, they may perform different specialized tasks. For example, electronic appliance 600(1) might comprise a central processing section of a workstation responsible for managing the overall operation of the workstation and providing computation resources. Electronic appliance 600(2) might be a mass storage device 620 for the same workstation, and could provide a storage mechanism 2636 that might, for example, read information from and write information to a secondary storage device 652. Electronic appliance 600(3) might be a display device 614 responsible for performing display tasks, and could provide a displaying mechanism 2638 such as a graphics controller and associated video or other display. Electronic appliance 600(N) might be a printer 622 that performs printing related tasks and could include, for example, a print mechanism 2640.

Each of electronic appliances 600(1), . . . 600(N) could comprise a different module of the same workstation device all contained within a common housing, or the different electronic

appliances could be located within different system components. For example, electronic appliance 600(2) could be disposed within a disk controller unit, electronic appliance 600(3) could be disposed within a display device 614 housing, and the electronic appliance 600(N) could be disposed within the housing of a printer 622. Referring back to Figure 7, scanner 626, modem 618, telecommunication means 624, keyboard 612 and/or voice recognition box 613 could each comprise a VDE electronic appliance 600 having its own SPU 500. Additional examples include RF or otherwise wireless interface controller, a serial interface controller, LAN controllers, MPEG (video) controllers, etc.

Because electronic appliances 600(1) . . . 600(N) are each VDE-capable, they each have the ability to perform encryption and/or decryption of VDE-protected information. This means that information communicated across network 672 or other communications path 2631 connecting the electronic appliances can be VDE-protected (e.g., it may be packaged in the form of VDE administrative and/or content objects and encrypted as discussed above). One of the consequences of this arrangement is that an eavesdropper who taps into communications path 2631 will not be able obtain information except in VDE-protected form. For example, information generated by electronic appliance 600 (1) to be printed could be packaged in a VDE content object 300

and transmitted over path 2631 to electronic appliance 600 (N) for printing. An attacker would gain little benefit from intercepting this information since it is transmitted in protected form; she would have to compromise electronic appliance 600(1) or 600(N) (or the SPU 500(1), 500(N)) in order to access this information in unprotected form.

Another advantage provided by the arrangement shown in the diagram is that each of electronic appliances 600(1), . . . 600(N) may perform their own metering, control and/or other VDE-related functions. For example, electronic appliance 600(N) may meter and/or perform any other VDE control functions related to the information to be printed, electronic appliance 600(3) may meter and/or perform any other VDE control functions related to the information to be displayed, electronic appliance 600(2) may meter and/or perform any other VDE control functions related to the information to be stored and/or retrieved from mass storage 620, and electronic appliance 600(1) may meter and/or perform any other VDE control functions related to the information it processes.

In one specific arrangement, each of electronic appliances 600(1), . . . 600(N) would receive a command that indicates that the information received by or sent to the electronic appliance is to use its SPU 500 to process the information to follow. For

example, electronic appliance 600(N) might receive a command that indicates that information it is about to receive for printing is in VDE-protected form (or the information that is sent to it may itself indicate this). Upon receiving this command or other information, electronic appliance 600(N) may decrypt the received information using SPU 500, and might also meter the information the SPU provides to the print mechanism 2644 for printing. An additional command might be sent to electronic appliance 600(N) to disable the decryption process or 600(N)'s VDE secure subsystem may determine that the information should not be decrypted and/or printed. Additional commands, for example, may exist to load encryption/decryption keys, load "limits," establish "fingerprinting" requirements, and read metered usage. These additional commands may be sent in encrypted or unencrypted form as appropriate.

Suppose, for example, that electronic appliance 600(1) produces information it wishes to have printed by a VDE-capable printer 622. SPU 500(1) could establish a secure communications across path 2631 with SPU 500(N) to provide a command instructing SPU 500(N) to decrypt the next block of data and store it as a decryption key and a limit. SPU 500(1) might then send a further command to SPU 500(N) to use the decryption key and associated limit to process any following encrypted print stream (or this command could be sent by CPU 654(1) to

microcontroller 654(N)). Electronic appliance 600(1) could then begin sending encrypted information on path 672 for decryption and printing by printer 622. Upon receipt of each new block of information by printer 622, SPU 500(N) might first check to ensure that the limit is greater than zero. SPU 500(N) could then increment a usage meter value it maintains, and decrement the limit value. If the limit value is non-zero, SPU 500(N) could decrypt the information it has received and provide it to print mechanism 2640 for printing. If the limit is zero, then SPU 500(N) would not send the received information to the print mechanism 2640, nor would it decrypt it. Upon receipt of a command to stop, printer 622 could revert to a "non-secure" mode in which it would print everything received by it across path 2631 without permitting VDE processing.

The SPU 500(N) associated with printer 622 need not necessarily be disposed within the housing of the printer, but could instead be placed within an I/O controller 660 for example (see Figure 8). This would allow at least some of the advantages similar to the ones discussed above to be provided without requiring a special VDE-capable printer 622. Alternatively, a SPU 500(N) could be provided both within printer 622 and within I/O controller 660 communicating with the printer to provide advantages in terms of coordinating I/O control and relieving processing burdens from the SPU 500 associated with the central

processing electronic appliance 600(1). When multiple VDE instances occur within an electronic appliance, one or more VDE secure subsystems may be "central" subsystems, that is "secondary" VDE instances may pass encrypted usage related information to one or more central secure subsystems so as to allow said central subsystem to directly control storage of said usage related information. Certain control information may also be centrally stored by a central subsystem and all or a portion of such information may be securely provided to the secondary secure subsystem upon its secure VDE request.

Portable Electronic Appliance

Electronic appliance 600 provided by the present invention may be portable. Figure 71 shows one example of a portable electronic appliance 2600. Portable appliance 2600 may include a portable housing 2602 that may be about the size of a credit card in one example. Housing 2602 may connect to the outside world through, for example, an electrical connector 2604 having one or more electrical contact pins (not shown). Connector 2604 may electrically connect an external bus interface 2606 internal to housing 2602 to a mating connector 2604a of a host system 2608. External bus interface 2606 may, for example, comprise a PCMCIA (or other standard) bus interface to allow portable appliance 2600 to interface with and communicate over a bus 2607 of host system 2608. Host 2608 may, for example, be almost

any device imaginable, such as a computer, a pay telephone, another VDE electronic appliance 600, a television, an arcade video game, or a washing machine, to name a few examples.

Housing 2602 may be tamper resistant. (See discussion above relating to tamper resistance of SPU barrier 502.)

Portable appliance 2600 in the preferred embodiment includes one or more SPUs 500 that may be disposed within housing 2602. SPU 500 may be connected to external bus interface 2606 by a bus 2610 internal to housing 2602. SPU 500 communicates with host 2608 (through external bus interface 2606) over this internal bus 2610.

SPU 500 may be powered by a battery 2612 or other portable power supply that is preferably disposed within housing 2602. Battery 2612 may be, for example, a miniature battery of the type found in watches or credit card sized calculators. Battery 2612 may be supplemented (or replaced) by solar cells, rechargeable batteries, capacitive storage cells, etc.

A random access memory (RAM) 2614 is preferably provided within housing 2602. RAM 2614 may be connected to SPU 500 and not directly connected to bus 2610, so that the contents of RAM 2614 may be accessed only by the SPU and not

by host 2608 (except through and as permitted by the SPU). Looking at Figure 9 for a moment, RAM 2614 may be part of RAM 534 within the SPU 500, although it need not necessarily be contained within the same integrated circuit or other package that houses the rest of the SPU.

Portable appliance 2600 RAM 534 may contain, for example, information which can be used to uniquely identify each instance of the portable appliance. This information may be employed (e.g. as at least a portion of key or password information) in authentication, verification, decryption, and/or encryption processes.

Portable appliance 2600 may, in one embodiment, comprise means to perform substantially all of the functions of a VDE electronic appliance 600. Thus, for example, portable appliance 2600 may include the means for storing and using permissions, methods, keys, programs, and/or other information, and can be capable of operating as a "stand alone" VDE node.

In a further embodiment, portable appliance 2600 may perform preferred embodiment VDE functions once it has been coupled to an additional external electronic appliance 600. Certain information, such as database management permission(s), method(s), key(s), and/or other important

information (such as at least a portion of other VDE programs: administrative, user-interface, analysis, etc.) may be stored (for example as records) at an external VDE electronic appliance 600 that may share information with portable appliance 2600.

One possible "stand alone" configuration for tamper-resistant, portable appliance 2600 arrangements includes a tamper-resistant package (housing 2602) containing one or more processors (500, 2616) and/or other computing devices and/or other control logic, along with random-access-memory 2614. Processors 500, 2616 may execute permissions and methods wholly (or at least in part) within the portable appliance 2600. The portable appliance 2600 may have the ability to encrypt information before the information is communicated outside of the housing 2602 and/or decrypt received information when said received information is received from outside of the housing. This version would also possess the ability to store at least a portion of permission, method, and/or key information securely within said tamper resistant portable housing 2602 on non-volatile memory.

Another version of portable appliance 2600 may obtain permissions and/or methods and/or keys from a local VDE electronic appliance 600 external to the portable appliance 2600 to control, limit, or otherwise manage a user's use of a VDE

protected object. Such a portable appliance 600 may be contained within, received by, installed in, or directly connected to, another electronic appliance 2600.

One example of a "minimal" configuration of portable appliance 2600 would include only SPU 500 and battery 2612 within housing 2602 (the external bus interface 2606 and the RAM 2614 would in this case each be incorporated into the SPU block shown in the Figure). In other, enhanced examples of portable appliance 2600, any or all of the following optional components may also be included within housing 2602:

- one or more CPUs 2616 (with associated support components such as RAM-ROM 2617, I/O controllers (not shown), etc.);
- one or more display devices 2618;
- one or more keypads or other user input buttons/control information 2620;
- one or more removable/replaceable memory device(s) 2622;
- and
- one or more printing device(s) 2624.

In such more enhanced versions, the display 2618, keypad 2620, memory device 2622 and printer 2624 may be connected to bus 2610, or they might be connected to CPU 2616 through an I/O port/controller portion (not shown) of the CPU. Display 2618 may

be used to display information from SPU 500, CPU 2616 and/or host 2608. Keypad 2620 may be used to input information to SPU 500, CPU 2616 and/or host 2608. Printer 2624 may be used to print information from any/all of these sources.

Removable/replaceable memory 2622 may comprise a memory cartridge or memory medium such as a bulk storage device, for providing additional long-term or short-term storage. Memory 2622 may be easily removable from housing 2602 if desired.

In one example embodiment, portable appliance 2600 may have the form factor of a "smart card" (although a "smart card" form factor may provide certain advantages, housing 2602 may have the same or different form factor as "conventional" smart cards). Alternatively, such a portable electronic appliance 2600 may, for example, be packaged in a PCMCIA card configuration (or the like) which is currently becoming quite popular on personal computers and is predicted to become common for desk-top computing devices and Personal Digital Assistants. One advantageous form factor for the portable electronic appliance housing 2602 may be, for example, a Type 1, 2, or 3 PCMCIA card (or other derivations) having credit card or somewhat larger dimensions. Such a form factor is conveniently portable, and may be insertable into a wide array of computers and consumer appliances, as well as receptacles at commercial establishments such as retail establishments and banks, and at public

communications points, such as telephone or other telecommunication "booths."

Housing 2602 may be insertable into and removable from a port, slot or other receptacle provided by host 2608 so as to be physically (or otherwise operatively) connected to a computer or other electronic appliance. The portable appliance connector 2604 may be configured to allow easy removability so that appliance 2600 may be moved to another computer or other electronic appliance at a different location for a physical connection or other operative connection with that other device.

Portable electronic appliance 2600 may provide a valuable and relatively simple means for a user to move permissions and methods between their (compatible) various electronic appliances 600, such as between a notebook computer, a desktop computer and an office computer. It could also be used, for example, to allow a consumer to visit a next door neighbor and allow that neighbor to watch a movie that the consumer had acquired a license to view, or perhaps to listen to an audio record on a large capacity optical disk that the consumer had licensed for unlimited plays.

Portable electronic appliance 2600 may also serve as a "smart card" for financial and other transactions for users to

employ in a variety of other applications such as, for example, commercial applications. The portable electronic appliance 2600 may, for example, carry permission and/or method information used to authorize (and possibly record) commercial processes and services.

An advantage of using the preferred embodiment VDE portable appliance 2600 for financial transactions such as those typically performed by banks and credit card companies is that VDE allows financial clearinghouses (such as VISA, MasterCard, or American Express) to experience significant reductions in operating costs. The clearinghouse reduction in costs result from the fact that the local metering and budget management that occurs at the user site through the use of a VDE electronic appliance 600 such as portable appliance 2600 frees the clearinghouse from being involved in every transaction. In contrast to current requirements, clearinghouses will be able to perform their functions by periodically updating their records (such as once a month). Audit and/or budget "roll-ups" may occur during a connection initiated to communicate such audit and/or budget information and/or through a connection that can occur at periodic or relatively periodic intervals and/or during a credit updating, purchasing, or other portable appliance 2600 transaction.

Clearinghouse VDE digital distribution transactions would require only occasional authorization and/or audit or other administrative "roll-ups" to the central service, rather than far more costly connections during each session. Since there would be no requirement for the maintenance of a credit card purchase "paper trail" (the authorization and then forwarding of the credit card slip), there could be substantial cost reductions for clearinghouses (and, potentially, lower costs to users) due to reduction in communication costs, facilities to handle concurrent processing of information, and paper handling aspects of transaction processing costs. This use of a portable appliance 2600 would allow credit enforcement to exploit distributed processing employing the computing capability in each VDE electronic appliance 600. These credit cost and processing advantages may also apply to the use of non-smart card and non-portable VDE electronic appliance 600s.

Since VDE 100 may be configured as a highly secure commercial environment, and since the authentication processes supported by VDE employ digital signature processes which provide a legal validation that should be equivalent to paper documentation and handwritten signatures, the need for portable appliance 2600 to maintain paper trails, even for more costly transactions, is eliminated. Since auditable billing and control mechanisms are built into VDE 100 and automated, they may

replace traditional electronic interfaces to VISA, Master Card, AMEX, and bank debit accounts for digitally distributed other products and services, and may save substantial operating costs for such clearinghouses.

Portable appliance 2600 may, if desired, maintain for a consumer a portable electronic history. The portable history can be, for example, moved to an electronic "dock" or other receptacle, in or operatively connected to, a computer or other consumer host appliance 2608. Host appliance 2608 could be, for example, an electronic organizer that has control logic at least in part in the form of a microcomputer and that stores information in an organized manner, e.g., according to tax and/or other transaction categories (such as type of use or activity). By use of this arrangement, the consumer no longer has to maintain receipts or otherwise manually track transactions but nevertheless can maintain an electronic, highly secure audit trail of transactions and transaction descriptions. The transaction descriptions may, for example, securely include the user's digital signature, and optionally, the service or goods provider's digital signature.

When a portable appliance 2600 is "docked" to a host 2608 such as a personal computer or other electronic appliance (such as an electronic organizer), the portable appliance 2600 could communicate interim audit information to the host. In one

embodiment, this information could be read, directly or indirectly, into a computer or electronic organizer money and/or tax management program (for example, Quicken or Microsoft Money and/or Turbo Tax and/or Andrew Tobias' Managing Your Money). This automation of receipt management would be an enormous boon to consumers, since the management and maintenance of receipts is difficult and time-consuming, receipts are often lost or forgotten, and the detail from credit card billings is often wholly inadequate for billing and reimbursement purposes since credit card billings normally don't provide sufficient data on the purchased items or significant transaction parameters.

In one embodiment, the portable appliance 2600 could support secure (in this instance encrypted and/or authenticated) two-way communications with a retail terminal which may contain a VDE electronic appliance 600 or communicate with a retailer's or third party provider's VDE electronic appliance 600. During such a secure two-way communication between, for example, each participant's secure VDE subsystem, portable appliance 2600 VDE secure subsystem may provide authentication and appropriate credit or debit card information to the retail terminal VDE secure subsystem. During the same or different communication session, the terminal could similarly, securely communicate back to the portable appliance 2600 VDE

secure subsystem details as to the retail transaction (for example, what was purchased and price, the retail establishment's digital signature, the retail terminal's identifier, tax related information, etc.).

For example, a host 2608 receptacle for receiving and/or attaching to portable appliance 2600 could be incorporated into or operatively connected to, a retail or other commercial establishment terminal. The host terminal 2608 could be operated by either a commercial establishment employee or by the portable appliance 2600 holder. It could be used to, for example, input specific keyboard and/or voice input specific information such as who was taken to dinner, why something was purchased, or the category that the information should be attached to. Information could then be automatically "parsed" and routed into securely maintained (for example, encrypted) appropriate database management records within portable appliance 2600. Said "parsing" and routing would be securely controlled by VDE secure subsystem processes and could, for example, be based on category information entered in by the user and/or based on class of establishment and/or type (category) of expenditure information (or other use). Categorization can be provided by the retail establishment, for example, by securely communicating electronic category information as a portion, for example, of electronic receipt information or alternatively by

printing a hard copy receipt using printer 2624. This process of categorization may take place in the portable appliance 2600 or, alternatively, it could be performed by the retail establishment and periodically "rolled-up" and communicated to the portable appliance 2600 holder.

Retail, clearinghouse, or other commercial organizations may maintain and use by securely communicating to appliance 2600 one or more of generic classifications of transaction types (for example, as specified by government taxation rules) that can be used to automate the parsing of information into records and/or for database information "roll-ups" for; and/or in portable appliance 2600 or one or more associated VDE nodes. In such instances, host 2608 may comprise an auxiliary terminal, for example, or it could comprise or be incorporated directly within a commercial establishments cash registers or other retail transactions devices. The auxiliary terminal could be menu and/or icon driven, and allow very easy user selection of categorization. It could also provide templates, based on transaction type, that could guide the user through specifying useful or required transaction specific information (for example, purpose for a business dinner and/or who attended the dinner). For example, a user might select a business icon, then select from travel, sales, meals, administration, or purchasing icons for example, and then might enter in very specific information

and/or a key word, or other code that might cause the downloading of a transaction's detail into the portable appliance 2600. This information might also be stored by the commercial establishment, and might also be communicated to the appropriate government and/or business organizations for validation of the reported transactions (the high level of security of auditing and communications and authentication and validation of VDE should be sufficiently trusted so as not to require the maintenance of a parallel audit history, but parallel maintenance may be supported, and maintained at least for a limited period of time so as to provide backup information in the event of loss or "failure" of portable appliance 2600 and/or one or more appliance 2600 associated VDE installations employed by appliance 2600 for historical and/or status information record maintenance). For example, of a retail terminal maintained necessary transaction information concerning a transaction involving appliance 2600, it could communicate such information to a clearinghouse for archiving (and/or other action) or it could periodically, for example, at the end of a business day, securely communicate such information, for example, in the form of a VDE content container object, to a clearinghouse or clearinghouse agent. Such transaction history (and any required VDE related status information such as available credit) can be maintained and if necessary, employed to reconstruct the information in a portable appliance 2600 so as to allow a replacement appliance to

be provided to an appliance 2600 user or properly reset internal information in data wherein such replacement and/or resetting provides all necessary transaction and status information.

In a retail establishment, the auxiliary terminal host 2608 might take the form of a portable device presented to the user, for example at the end of a meal. The user might place his portable appliance 2600 into a smart card receptacle such as a PCMCIA slot, and then enter whatever additional information that might appropriately describe the transaction as well as satisfying whatever electronic appliance 600 identification procedure(s) required. The transaction, given the availability of sufficient credit, would be approved, and transaction related information would then be communicated back from the auxiliary terminal directly into the portable appliance 2600. This would be a highly convenient mode of credit usage and record management.

The portable device auxiliary terminal might be "on-line," that is electronically communicating back to a commercial establishment and/or third party information collection point through the use of cellular, satellite, radio frequency, or other communications means. The auxiliary terminal might, after a check by a commercial party in response to receipt of certain identification information at the collection point, communicate back to the auxiliary terminal whether or not to accept the

portable appliance 2600 based on other information, such as a bad credit record or a stolen portable appliance 2600. Such a portable auxiliary terminal would also be very useful at other commercial establishments, for example at gasoline stations, rental car return areas, street and stadium vendors, bars, and other commercial establishments where efficiency would be optimized by allowing clerks and other personnel to consummate transactions at points other than traditional cash register locations.

As mentioned above, portable appliance 2600 may communicate from time to time with other electronic appliances 600 such as, for example, a VDE administrator. Communication during a portable appliance 2600 usage session may result from internally stored parameters dictating that the connection should take place during that current session (or next or other session) of use of the portable appliance. The portable appliance 600 can carry information concerning a real-time date or window of time or duration of time that will, when appropriate, require the communication to take place (e.g., perhaps before the transaction or other process which has been contemplated by the user for that session or during it or immediately following it). Such a communication can be accomplished quickly, and could be a secure, VDE two-way communication during which information is communicated to a central information handler. Certain other

information may be communicated to the portable appliance 2600 and/or the computer or other electronic appliance to which the portable appliance 2600 has been connected. Such communicated other information can enable or prevent a contemplated process from proceeding, and/or make the portable appliance 2600, at least in part, unusable or useable. Information communicated to the portable appliance 2600 could include one or more modifications to permissions and methods, such as a resetting or increasing of one or more budgets, adding or withdrawing certain permissions, etc.

The permissions and/or methods (i.e., budgets) carried by the portable appliance 2600 may have been assigned to it in conjunction with an "encumbering" of another, stationary or other portable VDE electronic appliance 600. In one example, a portable appliance 2600 holder or other VDE electronic appliance 600 and/or VDE electronic appliance 600 user could act as "guarantor" of the financial aspects of a transaction performed by another party. The portable appliance 2600 of the holder would record an "encumbrance," which may be, during a secure communication with a clearinghouse, be recorded and maintained by the clearinghouse and/or some other financial services party until all or a portion of debt responsibilities of the other party were paid or otherwise satisfied. Alternatively or in addition, the encumbrance may also be maintained within the

portable appliance 2600, representing the contingent obligation of the guarantor. The encumbrance may be, by some formula, included in a determination of the credit available to the guarantor. The credit transfer, acceptance, and/or record management, and related processes, may be securely maintained by the security features provided by aspects of the present invention. Portable appliance 600 may be the sole location for said permissions and/or methods for one or more VDE objects 300, or it may carry budgets for said objects that are independent of budgets for said objects that are found on another, non-portable VDE electronic appliance 600. This may allow budgets, for example, to be portable, without requiring "encumbering" and budget reconciliation.

Portable VDE electronic appliance 2600 may carry (as may other VDE electronic appliance 600s described) information describing credit history details, summary of authorizations, and usage history information (e.g., audit of some degree of transaction history or related summary information such as the use of a certain type/class of information) that allows re-use of certain VDE protected information at no cost or at a reduced cost. Such usage or cost of usage may be contingent, at least in part, on previous use of one or more objects or class of objects or amount of use, etc., of VDE protected information.

Portable appliance 2600 may also carry certain information which may be used, at least in part, for identification purposes. This information may be employed in a certain order (e.g. a pattern such as, for example, based on a pseudo-random algorithm) to verify the identity of the carrier of the portable appliance 2600. Such information may include, for example, one's own or a wife's and/or other relatives maiden names, social security number or numbers of one's own and/or others, birth dates, birth hospital(s), and other identifying information. It may also or alternatively provide or include one or more passwords or other information used to identify or otherwise verify/authenticate an individual's identity, such as voice print and retinal scan information. For example, a portable appliance 2600 can be used as a smart card that carries various permissions and/or method information for authorizations and budgets. This information can be stored securely within portable appliance 2600 in a secure database 610 arrangement. When a user attempts to purchase or license an electronic product or otherwise use the "smart card" to authorize a process, portable appliance 2600 may query the user for identification information or may initiate an identification process employing scanned or otherwise entered information (such as user fingerprint, retinal or voice analysis or other techniques that may, for example, employ mapping and/or matching of provided characteristics to information securely stored within the portable appliance 2600).

The portable appliance 2600 may employ different queries at different times (and/or may present a plurality of queries or requests for scanning or otherwise entering identifying information) so as to prevent an individual who has come into possession of appropriate information for one or more of the "tests" of identity from being able to successfully employ the portable appliance 2600.

A portable appliance 600 could also have the ability to transfer electronic currency or credit to another portable appliance 2600 or to another individual's account, for example, using secure VDE communication of relevant content between secure VDE subsystems. Such transfer may be accomplished, for example, by telecommunication to, or presentation at, a bank which can transfer credit and/or currency to the other account. The transfer could also occur by using two cards at the same portable appliance 2600 docking station. For example, a credit transaction workstation could include dual PCMCIA slots and appropriate credit and/or currency transfer application software which allows securely debiting one portable appliance 2600 and "crediting" another portable appliance (i.e., debiting from one appliance can occur upon issuing a corresponding credit and/or currency to the other appliance). One portable appliance 600, for example, could provide an authenticated credit to another user. Employing two "smart card" portable appliance 600 would enable

the user of the providing of "credit" "smart card" to go through a transaction process in which said user provides proper identification (for example, a password) and identifies a "public key" identifying another "smart card" portable appliance 2600. The other portable appliance 2600 could use acceptance processes, and provide proper identification for a digital signature (and the credit and/or currency sender may also digitally sign a transaction certificate so the sending act may not be repudiated and this certificate may accompany the credit and/or currency as VDE container content. The transactions may involve, for example, user interface interaction that stipulates interest and/or other terms of the transfer. It may employ templates for common transaction types where the provider of the credit is queried as to certain parameters describing the agreement between the parties. The receiving portable appliance 2600 may iteratively or as a whole be queried as to the acceptance of the terms. VDE negotiation techniques described elsewhere in this application may be employed in a smart card transfer of electronic credit and/or currency to another VDE smart card or other VDE installation.

Such VDE electronic appliance 600/portable appliance 2600 credit transfer features would significantly reduce the overhead cost of managing certain electronic credit and/or currency activities by significantly automating these processes

through extending the computerization of credit control and credit availability that was begun with credit cards and extended with debit cards. The automation of credit extension and/or currency transfer and the associated distributed processing advantages described, including the absence of any requirement for centralized processing and telecommunications during each transaction, truly make credit and/or currency, for many consumers and other electronic currency and/or credit users, an efficient, trusted, and portable commodity.

The portable appliance 2600 or other VDE electronic appliance 600, can, in one embodiment, also automate many tax collection functions. A VDE electronic appliance 600 may, with great security, record financial transactions, identify the nature of the transaction, and identify the required sales or related government transaction taxes, debit the taxes from the users available credit, and securely communicate this information to one or more government agencies directly at some interval (for example monthly), and/or securely transfer this information to, for example, a financial clearinghouse, which would then transfer one or more secure, encrypted (or unsecure, calculated by clearinghouse, or otherwise computed) information audit packets (e.g., VDE content containers and employing secure VDE communication techniques) to the one or more appropriate, participating government agencies. The overall integrity and

security of VDE 100 could ensure, in a coherent and centralized manner, that electronic reporting of tax related information (derived from one or more electronic commerce activities) would be valid and comprehensive. It could also act as a validating source of information on the transfer of sales tax collection (e.g., if, for example, said funds are transferred directly to the government by a commercial operation and/or transferred in a manner such that reported tax related information cannot be tampered with by other parties in a VDE pathway of tax information handling). A government agency could select transactions randomly, or some subset or all of the reported transactions for a given commercial operation can be selected. This could be used to ensure that the commercial operation is actually paying to the government all appropriate collected funds required for taxes, and can also ensure that end-users are charged appropriate taxes for their transactions (including receipt of interest from bank accounts, investments, gifts, etc.

Portable appliance 2600 financial and tax processes could involve template mechanisms described elsewhere herein. While such an electronic credit and/or currency management capability would be particularly interesting if managed at least in part, through the use of a portable appliance 2600, credit and/or currency transfer and similar features would also be applicable

for non-portable VDE electronic appliance 600's connected to or installed within a computer or other electronic device.

User Notification Exception Interface ("Pop Up") 686

As described above, the User Modification Exception Interface 686 may be a set of user interface programs for handling common VDE functions. These applications may be forms of VDE templates and are designed based upon certain assumptions regarding important options, specifically, appropriate to a certain VDE user model and important messages that must be reported given certain events. A primary function of the "pop-up" user interface 686 is to provide a simple, consistent user interface to, for example, report metering events and exceptions (e.g., any condition for which automatic processing is either impossible or arguably undesirable) to the user, to enable the user to configure certain aspects of the operation of her electronic appliance 600 and, when appropriate, to allow the user to interactively control whether to proceed with certain transaction processes. If an object contains an exception handling method, that method will control how the "pop-up" user interface 686 handles specific classes of exceptions.

The "pop-user" interface 686 normally enables handling of tasks not dedicated to specific objects 300, such as for example:

- Logging onto an electronic appliance 600 and/or entering into a VDE related activity or class of activities,
- Configuring an electronic appliance 600 for a registered user, and/or generally for the installation, with regard to user preferences, and automatic handling of certain types of exceptions,
- Where appropriate, user selecting of meters for use with specific properties, and
- Providing an interface for communications with other electronic appliances 600, including requesting and/or for purchasing or leasing content from distributors, requesting clearinghouse credit and/or budgets from a clearinghouse, sending and/or receiving information to and/or from other electronic appliances, and so on.

Figure 72A shows an example of a common "logon" VDE electronic appliance 600 function that may use user interface 686. "Log-on" can be done by entering a user name, account name, and/or password. As shown in the provided example, a configuration option provided by the "pop-up" user interface 686 dialog can be "Login at Setup", which, if selected, will initiate a VDE Login procedure automatically every time the user's

electronic appliance 600 is turned on or reset. Similarly, the "pop-up" user interface 686 could provide an interface option called "Login at Type" which, if selected, will initiate a procedure automatically every time, for example, a certain type of object or specific content type application is opened such as a file in a certain directory, a computer application or file with a certain identifying extension, or the like.

Figure 72B shows an example of a "pop-up" user interface 686 dialog that is activated when an action by the user has been "trapped," in this case to warn the user about the amount of expense that will be incurred by the user's action, as well as to alert the user about the object 300 which has been requested and what that particular object will cost to use. In this example, the interface dialog provides a button allowing the user to request further detailed information about the object, including full text descriptions, a list of associated files, and perhaps a history of past usage of the object including any residual rights to use the object or associated discounts.

The "Cancel" button 2660 in Figure 72B cancels the user's trapped request. "Cancel" is the default in this example for this dialog and can be activated, for example, by the return and enter keys on the user's keyboard 612, by a "mouse click" on that button, by voice command, or other command mechanisms. The

"Approve button" 2662, which must be explicitly selected by a mouse click or other command procedure, allows the user to approve the expense and proceed. The "More options" control 2664 expands the dialog to another level of detail which provides further options, an example of which is shown in Figure 72C.

Figure 72C shows a secondary dialog that is presented to the user by the "pop-up" user interface 686 when the "More options" button 2664 in Figure 72B is selected by the user. As shown, this dialog includes numerous buttons for obtaining further information and performing various tasks.

In this particular example, the user is permitted to set "limits" such as, for example, the session dollar limit amount (field 2666), a total transaction dollar limit amount (field 2668), a time limit (in minutes) (field 2670), and a "unit limit" (in number of units such as paragraphs, pages, etc.) (field 2672). Once the user has made her selections, she may "click on" the OKAY button (2674) to confirm the limit selections and cause them to take effect.

Thus, pop-up user interface dialogues can be provided to specify user preferences, such as setting limits on budgets and/or other aspects of object content usage during any one session or over a certain duration of time or until a certain point in time.

Dialogs can also be provided for selecting object related usage options such as selecting meters and budgets to be used with one or more objects. Selection of options may be applied to types (that is classes) of objects by associating the instruction with one or more identifying parameters related to the desired one or more types. User specified configuration information can set default values to be used in various situations, and can be used to limit the number or type of occasions on which the user's use of an object is interrupted by a "pop-up" interface 686 dialog. For example, the user might specify that a user request for VDE protected content should be automatically processed without interruption (resulting from an exceptions action) if the requested processing of information will not cost more than \$25.00 and if the total charge for the entire current session (and/or day and/or week, etc.) is not greater than \$200.00 and if the total outstanding and unpaid charge for use hasn't exceeded \$2500.00.

Pop-up user interface dialogs may also be used to notify the user about significant conditions and events. For example, interface 686 may be used to:

- remind the user to send audit information to a clearinghouse,

- inform a user that a budget value is low and needs replenishing,
- remind the user to back up secure database 610, and
- inform the user about expirations of PERCs or other dates/times events.

Other important "pop-up" user interface 686 functions include dialogs which enable flexible browsing through libraries of properties or objects available for licensing or purchase, either from locally stored VDE protected objects and/or from one or more various, remotely located content providers. Such function may be provided either while the user's computer is connected to a remote distributor's or clearinghouse's electronic appliance 600, or by activating an electronic connection to a remote source after a choice (such as a property, a resource location, or a class of objects or resources is selected). A browsing interface can allow this electronic connection to be made automatically upon a user selection of an item, or the connection itself can be explicitly activated by the user. See Figure 72D for an example of such a "browsing" dialog.

Smart Objects

VDE 100 extends its control capabilities and features to "intelligent agents." Generally, an "intelligent agent" can act as

an emissary to allow a process that dispatches it to achieve a result the originating process specifies. Intelligent agents that are capable of acting in the absence of their dispatch process are particularly useful to allow the dispatching process to access, through its agent, the resources of a remote electronic appliance. In such a scenario, the dispatch process may create an agent (e.g., a computer program and/or control information associated with a computer program) specifying a particular desired task(s), and dispatch the agent to the remote system. Upon reaching the remote system, the "agent" may perform its assigned task(s) using the remote system's resources. This allows the dispatch process to, in effect, extend its capabilities to remote systems where it is not present.

Using an "agent" in this manner increases flexibility. The dispatching process can specify, through its agent, a particular desired task(s) that may not exist or be available on the remote system. Using such an agent also provides added trustedness; the dispatch process may only need to "trust" its agent, not the entire remote system. Agents have additional advantages.

Software agents require a high level of control and accountability to be effective, safe and useful. Agents in the form of computer viruses have had devastating effects worldwide. Therefore, a system that allows an agent to access it should be

able to control it or otherwise prevent the agent from damaging important resources. In addition, systems allowing themselves to be accessed by an agent should sufficiently trust the agent and/or provide mechanisms capable of holding the true dispatcher of the agent responsible for the agent's activities. Similarly, the dispatching process should be able to adequately limit and/or control the authority of the agents it dispatches or else it might become responsible for unforeseen activities by the agent (e.g., the agent might run up a huge bill in the course of following imprecise instructions it was given by the process that dispatched it).

These significant problems in using software agents have not been adequately addressed in the past. The open, flexible control structures provided by VDE 100 address these problems by providing the desired control and accountability for software agents (e.g., agent objects). For example, VDE 100 positively controls content access and usage, provides guarantee of payment for content used, and enforces budget limits for accessed content. These control capabilities are well suited to controlling the activities of a dispatched agent by both the process that dispatches the agent and the resource accessed by the dispatched agent.

One aspect of the preferred embodiment provided by the present invention provides a "smart object" containing an agent. Generally, a "smart object" may be a VDE object 300 that contains some type(s) of software programs ("agents") for use with VDE control information at a VDE electronic appliance 600. A basic "smart object" may comprise a VDE object 300 that, for example, contains (physically and/or virtually):

a software agent, and

at least one rule and/or control associated with the

software agent that governs the agent's operation.

Although this basic structure is sufficient to define a "smart object," Figure 73 shows a combination of containers and control information that provides one example of a particularly advantageous smart object structure for securely managing and controlling the operation of software agents.

As shown in Figure 73, a smart object 3000 may be constructed of a container 300, within which is embedded one or more further containers (300z, 300y, etc.). Container 300 may further contain rules and control information for accessing and using these embedded containers 300z, 300y, etc. Container 300z embedded in container 300 is what makes the object 3000 a "smart object." It contains an "agent" that is managed and controlled by VDE 100.

The rules and control information 806f associated with container 300z govern the circumstances under which the agent may be released and executed at a remote VDE site, including any limitations on execution based on the cost of execution for example. This rule and control information may be specified entirely in container 300z, and/or may be delivered as part of container 300, as part of another container (either within container 300 or a separately deliverable container), and/or may be already present at the remote VDE site.

The second container 300y is optional, and contains content that describes the locations at which the agent stored in container 300z may be executed. Container 300y may also contain rules and control information 806e that describe the manner in which the contents of container 300y may be used or altered. This rule and control information 806e and/or further rules 300y(1) also contained within container 300y may describe searching and routing mechanisms that may be used to direct the smart object 3000 to a desired remote information resource. Container 300y may contain and/or reference rules and control information 300y(1) that specify the manner in which searching and routing information use and any changes may be paid for.

Container 300x is an optional content container that is initially "empty" when the smart object 3000 is dispatched to a

remote site. It contains rules and control information 300x(1) for storing the content that is retrieved by the execution of the agent contained in container 300z. Container 300x may also contain limits on the value of content that is stored in the retrieval container so as to limit the amount of content that is retrieved.

Other containers in the container 300 may include administrative objects that contain audit and billing trails that describe the actions of the agent in container 300z and any charges incurred for executing an agent at a remote VDE node. The exact structure of smart object 3000 is dependent upon the type of agent that is being controlled, the resources it will need for execution, and the types of information being retrieved.

The smart object 3000 in the example shown in Figure 73 may be used to control and manage the operation of an agent in VDE 100. The following detailed explanation of an example smart object transaction shown in Figure 74 may provide a helpful, but non-limiting illustration. In this particular example, assume a user is going to create a smart object 3000 that performs a library search using the "Very Fast and Efficient" software agent to search for books written about some subject of interest (e.g., "fire flies"). The search engine is designed to return a list of books to the user. The search engine in this example may spend no more than \$10.00 to find the appropriate books,

may spend no more than \$3.00 in library access or communications charges to get to the library, and may retrieve no more than \$15.00 in information. All information relating to the search or use is to be returned to the user and the user will permit no information pertaining to the user or the agent to be released to a third party.

In this example, a dispatching VDE electronic appliance 3010 constructs a smart object 3000 like the one shown in Figure 73. The rule set in 806a is specified as a control set that contains the following elements:

1. a smart_agent_execution event that specifies the smart agent is stored in embedded container 300z and has rules controlling its execution specified in that container;
2. a smart_agent_use event that specifies the smart agent will operate using information and parameters stored in container 300;
3. a routing_use event that specifies the information routing information is stored in container 300y and has rules controlling this information stored in that container;

4. an information_write event that specifies information written will be stored in container 300y, 300x, or 300w depending on its type (routing, retrieved, or administrative), and that these containers have independent rules that control how information is written into them.

The rule set in control set 806b contains rules that specify the rights desired by this smart object 3000. Specifically, this control set specifies that the software agent desires:

1. A right to use the "agent execution" service on the remote VDE site. Specific billing and charge information for this right is carried in container 300z.
2. A right to use the "software description list" service on the remote VDE site. Specific billing and charge information for this for this right is carried in container 300y.
3. A right to use an "information locator service" on a remote VDE site.

4. A right to have information returned to the user without charge (charges to be incurred on release of information and payment will be by a VISA budget)
5. A right to have all audit information returned such that it is readable only by the sender.

The rule set in control set 806c specifies that container 300w specifies the handling of all events related to its use. The rule set in control set 806d specifies that container 300x specifies the handling of all events related to its use. The rule set in control set 806e specifies that container 300y specifies the handling of all events related to its use. The rule set in control set 806f specifies that container 300z specifies the handling of all events related to its use.

Container 300z is specified as containing the "Very Fast and Efficient" agent content, which is associated with the following rules set:

1. A use event that specifies a meter and VISA budget that limits the execution to \$10.00 charged against the owner's VISA card. Audits of usage are required and will be stored in object 300w under control information specified in that object.

After container 300z and its set are specified, they are constructed and embedded in the smart object container 300.

Container 300y is specified as a content object with two types of content. Content type A is routing information and is read/write in nature. Content type A is associated with a rules set that specifies:

1. A use event that specifies no operation for the release of the content. This has the effect of not charging for the use of the content.
2. A write event that specifies a meter and a VISA budget that limits the value of writing to \$3.00. The billing method used by the write is left unspecified and will be specified by the control method that uses this rule.
3. Audits of usage are required and will be stored in object 300w under control information specified in that object.

Content type B is information that is used by the software agent to specify parameters for the agent. This content is