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.

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:

(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.

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 "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.

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 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

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 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

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 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 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 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."

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 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 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, 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 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 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 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 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 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 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.

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 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 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 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 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 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.

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 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;

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.

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.

The PERC 808 may reference one or more method "cores" 1000'. A method core 1000' may define a basic "method" 1000 (e.g., "control," "billing," "metering," etc.)

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 electronic appliances 600. Basic instructions may be comprised of, for example:

- 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;
- a sequence of electronically represented logical operations for use with an electronic appliance 600;
- 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:

- information that identifies associated basic instructions and said intrinsic data for access, correlation and/or validation purposes;
- required and/or optional parameters for use with basic instructions and said intrinsic data;
- information defining relationships to other methods;
- data elements that may comprise data values, fields of information, and/or the like;
- information specifying and/or defining relationships among data elements, basic instructions and/or intrinsic data;
- information specifying relationships to external data elements;
- information specifying relationships between and among internal and external data elements, methods, and/or the like, if any exist; and

- 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.

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.

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 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, 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 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.

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 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.

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.

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, 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.

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 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 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 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.

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.

More Detailed Discussion of Rights Operating System 602 Architecture

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.

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 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 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.

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 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.

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, 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 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 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.

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

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.

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 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.

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):

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);

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;

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;

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

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 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;

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 communications between electronic appliance 600 and the outside world.

Object Switch 734

Object switch 734 handles, controls and communicates (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 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

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 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.

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 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 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 and/or request services);

External Services Manager 772;

Object Submittal Manager 774; and

Communications Manager 776.

Briefly,

Object Repository Manager 770 provides services relating to access to object repository 728;

External Services Manager 772 provides services relating to requesting and receiving services externally, such as from a network resource or another site;

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 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 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 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 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 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 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 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 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

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 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.

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. The service number that the service is loaded as is passed in as *service_id* parameter. In the preferred embodiment, the service

returns 0 is the initialization process was completed successfully or an error number if some error occurred.

Mount

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 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:

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 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

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 resources, and an OPEN will actually open a socket to a remote installation.

Some services, such as commercial database manager 730 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.

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.

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).

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 (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 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 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 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:

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)

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 *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

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.

RPC IOCTL Call Example: ROI_SVC_IOCTL (long service_id, long subservice_id.
int command. BYTE *buffer)

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 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.
GET_STATS	SVC_STATS	Returns current statistics about a service/subservice.
CLR_STATS	None	Clears the statistics about a service/subservice.

* * * * *

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

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 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 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 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

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 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.

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.

The following table shows one example of SPE device driver 736 calls:

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.
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.
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.

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:

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

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.

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.

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.

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.

Example of an SPE "Reset Interface" Driver Call:

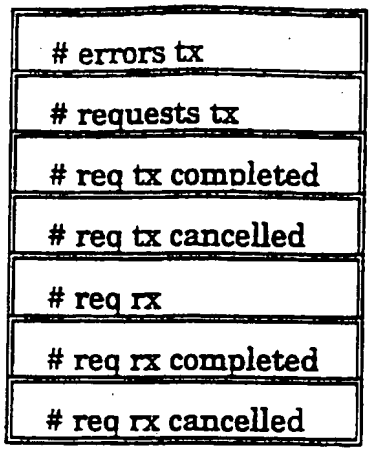
SPE_reset_interface (void)

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 (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). An example of the SPE_STATS structure may have the following definition:

Service id
packets rx
packets tx
bytes rx
bytes tx
errors rx



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.

Example of an SPE "Clear Statistics" Driver Call:

SPE_clear_stats (long service_id)

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))

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)

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:

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.

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 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 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 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 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 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 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 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 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).

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.

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 supporting VDE functions 602 may use these existing hooks to integrate with a preexisting operating system.

User Notification Service Manager 740

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, 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 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.

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.

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 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 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:

- a commercial database approach, and
- 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.

The commercial database approach uses a commercial database to store securely wrappered 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).

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.

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 desirable because it provides high performance in environments where resources are not limited.

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 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 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.

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.

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.

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 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

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., 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.

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 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 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 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 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.

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 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 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.

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).

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.

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 information to be contained within the object (e.g., rules and control information, identifying information, etc.).

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 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. 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.

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 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.

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).

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 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 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 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 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 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 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 687, and the user or the electronic appliance may "register" the new object by including appropriate information within secure database 610.

Communications Subsystem 776

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. 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 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 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 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 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).

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 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 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 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

- Channel Services Manager 562
- SPE RPC Manager 550
- Time Base Manager 554
- Encryption/Decryption Manager 556
- Key and Tag Manager 558

- Summary Services Manager 560
- Authentication Manager/Service Communications Manager 564
- Random Value Generator 565
- Secure Database Manager 566
- Other Services 592.

Each of the major functional blocks of PPE 650 is discussed in detail below.

I. **SPE Kernel/Dispatcher 552**

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-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 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 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

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)

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 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 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.

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 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 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" 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.

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. 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. 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.

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.

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.

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

discarded (e.g., as determined by its DTD use count) if no other swap block references it.

B. Kernel/Dispatcher Memory Management

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 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 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 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.

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 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 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.

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).

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 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 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 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.

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 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.

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 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 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 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. 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 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" 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 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.

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 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.

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 (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.

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 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 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 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 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 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 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.

Figure 15a shows a detailed example of a possible format for a channel header 596 and a channel 594 containing channel 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 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 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 1000' (or fragment thereof), load module 1100 and 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.

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. In 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) 1000', 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 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 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

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.

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 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, 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 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:

- "tick" of RTC 528
- interrupt from bus interface 530
- power fail interrupt
- watchdog timer interrupt
- interrupt from encrypt/decrypt engine 522
- 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.

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.

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 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 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 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 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 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.

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 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 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 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 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.

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., 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.

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 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.

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 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 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 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 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, 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 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:

Call Name	Source	Description
"Write Event"	Write	Writes an event to the channel for response by the channel. The Write Event 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 Bind Item 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 Unbind Item 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.

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 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 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).

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 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.

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.

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:

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.
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	

Call Name	Description
Bind Time	Bind timer services to a channel as an event source.
Unbind Time	Unbind timer services from a channel as an event source.
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.

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:

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.
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.
Initialize DES Instance	Initialize DES instance to be used.
Initialize RC-4 Instance	Initialize RC-4 instance to be used.
Initialize MD5 Instance	Initialize MD5 instance to be used.
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 Convoluted 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 a tag to read returned contents).
Tag Requests	
Get Tag	Get the validation (or other) tag for a specific VDE Item ID.
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. Forces a resynchronization of the hash table.
Get Hash Parameters	Retrieve the current hash parameters/algorithm.
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

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 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 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 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

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 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:

Control Information Retained in an SPE on Access Failures
Object ID
User ID
Type of failure
Time of failure

This information may be analyzed to detect cracking attempts or 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.

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).

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 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:

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.
Increment	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.
Destroy	Destroy the specified summary service if the user has a tag and/or "ticket" that permits them to request this service.

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 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 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.
	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.

SPE Authentication Manager/Service Communications Manager 564

The Authentication Manager/Service Communications 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 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.
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.

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.

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

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.

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 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 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" 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. 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:

Field
Hash Page Header
Distributor ID
Item ID
Site ID
User ID
Transaction Tag
Hash Page Entry
Creator ID
Item ID
Type
Version
Transaction Tag

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 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 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 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 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 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

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 external servicing.

SPE 503 Performance Considerations

Performance of SPE 503 (HPE 655) is a function of:

- complexity of the component assemblies used
- number of simultaneous component assembly operations
- amount of internal SPU memory available
- 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 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 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 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 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.

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

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:

VDE objects,

VDE process elements, and

VDE data structures.

The following table lists examples of some of the VDE items stored in or managed by information stored in secure database 610:

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.

Class		Brief Description
	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 VDE-protected items:

- one or more PERCs 808;
- methods 1000 (including static and dynamic method "cores" 1000, and MDEs 1202);
- Static UDEs 1200a and Dynamic UDEs 1200b; and
- load modules 1100.

Secure database 610 may also include the following additional data structures used and maintained for administrative purposes:

- an "object registry" 450 that references an object storage 728 containing one or more VDE objects;
- name service records 452; and

- configuration records 454 (including site configuration records 456 and user configuration records 458).

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.

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.

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 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 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 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 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 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.

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.

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.

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 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 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 "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 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

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 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.

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 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.

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 (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 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 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).

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 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 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 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 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 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).

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.

1. Stationary Objects

Figure 18 shows an example of a "Stationary Object" structure 850 provided by the preferred embodiment. "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., 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 "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 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 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" 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.

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 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:

(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

(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

(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).

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 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 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 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 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 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 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) 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).

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 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 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. 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.

The "out of channel" distribution may also allow the provider to receive payment for usage and/or elsewise 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 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 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 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 cryptoanalyst both more information in cyphertext to analyze and a greater incentive to perform cryptoanalysis.

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 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 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 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.

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.

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 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 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.

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 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.,

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 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 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 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 objects 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.

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.

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 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 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 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.

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).

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.

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, methods 1000 in the preferred embodiment are represented in secure database 610 by:

- method "cores" 1000';
- Method Data Elements (MDEs) 1202;
- User Data Elements (UDEs) 1200; and
- Data Description Elements (DTDs).

Method "core" 1000' 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 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 1000' that reference them, or the data structures may be delivered as part of the method cores. For example, the method core 1000' in the preferred embodiment may contain one or more MDEs 1202 and/or UDEs 1200 (or portions thereof). Method core 1000' 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 1000' 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.

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.

Figure 22 shows an example of a format for a method core 1000' provided by the preferred embodiment. A method core 1000' in the preferred embodiment contains a method event table 1006 and a method local data area 1008. Method event table 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 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 1000′ can be specific to a single user, or they 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 1000′. This structure allows for lower database overhead than when associating an entire method core 1000′ with a user/group. To enable a user to use a method, the user may be sent a method core 1000′ specifying a UDE 1200. If that method core 1000′ 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 registration time.

The Figure 22 example of a format for a method core 1000′ provided by the preferred embodiment includes a public (unencrypted) header 802, a private (encrypted) header 804, method event table 1006, and a method local data area 1008.

An example of a possible field layout for method core 1000′ public header 802 is shown in the following table:

Field Type		Description
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.
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:

Field Type		Description
Copy of Public Header 802 Method ID and "Other Classification Information"		Method ID from Public Header
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 1000' is used only under appropriate circumstances.
	Validation tag	
	Correlation tag	
Data Structure Reference		Optional Reference to DTD(s) and/or MDE(s)
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 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 1000' may respond to. Methods 1000 in the preferred embodiment may have completely different behavior 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 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 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, 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 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. 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 1000', or they may be stored within the secure database 610 as independent deliverables.

The following table is an example of a possible more detailed field layout for a method event record 1012:

Field Type	Description
Event Field 1014	Identifies corresponding event.
Access tag	Secret tag to grant access to this row of the method event record.

Field Type		Description
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.
# 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.

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.

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 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 authenticated in both of these cases. When a method core 1000' 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 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, 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 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 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 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.

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 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 1000' 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.

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.

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:

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 module.
	LM Exec Stack	Stack size required for the load module.

	Execution space code	Code that describes the execution space for this load module.
Access and reference tags	Access tag	Tags used to determine if the load module is the correct LM requested by the SPE.
	Validation tag	
	Correlation tag	Tag used to determine if the caller of the LM has the right to execute this LM.
	Digital Signature	Used to determine if the LM executable content is intact and was created by a trusted source (one with a correct certificate for creating LMs).
Data record descriptor information	DTD count	Number of DTDs that follow the code block.
	DTD 1 reference	If locally defined, the physical size and offset in bytes of the first DTD defined for this LM. If publicly referenced DTD, this is the DTD ID and the correlation tag to permit access to the record.
	...	
	DTD N reference	If locally defined, the physical size and offset in bytes of the Nth DTD defined for this LM. If publicly referenced DTD, this is the DTD ID and the correlation tag to permit access to the record.
Check Value		Check Value for entire LM.

Each load module 1100 also may use DTD 1108 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 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 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:

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 correct DTD requested by the SPE.
	Validation tag	
	Correlation tag	Tag used to determine if the caller of this DTD has 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.

Some examples of how load modules 1100 may use DTDs 1108 include:

- Increment data element (defined by name in DTD3) value in data area DTD4 by value in DTD1

- Set data element (defined by name in DTD3) value in data area DTD4 to value in DTD3
- Compute atomic element from event in DTD1 from table in DTD3 and return in DTD2
- Compute atomic element from event in DTD1 from equation in DTD3 and return in DTD2
- Create load module from load module creation template referenced in DTD3
- Modify load module in DTD3 using content in DTD4
- Destroy load module named in DTD3

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 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

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 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. 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.

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. 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 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 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 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 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 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 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 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.

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

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. 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:

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 uses.
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 specific 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 age with time.
Last Use Date	time_t	Meter/Budget	Date of last use.

Field type	Format	Typical Use	Description or Use
Start Date	time_t	Budget	Date of first allowable use.
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 audit.
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 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.
Atomic Element(s) & Object ID	Unsigned integer(s) 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 incudes 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 compromising 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.

Event Type	Event	Process Description
	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	Allocate more money to budget.
	receive method update	Update method.
	receive auditor change	Change from one auditor to another.
	receive change to audit interval	Change interval between audits.
	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	Create new budget.
	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.

Event Type	Event	Process Description
"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.
	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 the 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 patters 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 than 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, includes 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 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 2831is 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 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 algorithm 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.

Integrity of Software-Based PPE Security

As discussed above in cofnnection with Figure 10, some applications may use a software-based protected processing environment 650 (such as a "host event processing environment" (HPE) 655) providing a software-based tamper resistant barrier 674. Software-based tamper resistant barrier 674 may be created by software executing on a general-purpose CPU. Various software protection techniques may be used to construct and/or provide software-based tamper resistant barrier 674.

The risks or threat of attacks described above in connection with PPE 650 apply to a software-based PPE. An important threat to be countered with respect to a software-based tamper resistant barrier 674 is an attack based on a distributable computer program that can defeat the tamper

resistant barrier wherever the program is run. Since a software-based tamper resistant barrier 674 typically will not be as secure as a hardware-based tamper resistant barrier 502, it is useful to explore example steps and procedures a "cracker" might use to "'crack' a software"-based tamper resistant barrier.

Figures 67A and 67B show example "cracking" techniques a "cracker" might use to attack software-based tamper resistant barrier 674.

Referring to Figure 67A, the software used to create tamper resistant barrier 674 may be distributed, for example, on a storage medium 3370 such as a floppy diskette or optical disk (or, this software could be distributed electronically over network 108 and stored locally in a computer memory). The software distribution medium 3370 provides software (code and data) for loading into a computing device such as a general purpose personal computer 3372, for example. Personal computer 3372 may include, for example, a random access memory 3374 and a hard disk 3376.

In one example, the software distribution medium 3370 might include installation materials 3470 and operational materials 3472. The installation materials 3470 may be

executed by computer 3372 to install the operational materials 3472 onto the computer's hard disk 3376. The computer 3372 may then execute the operational materials 3472 from its hard disk 3376 to provide software-based protected processing environment 650 and associated software-based tamper resistant barrier 672.

In this example, one attack technique an attacker might use is to analyze software distribution medium 3370 (see Figure 67B, block 3352). Such analysis can take many forms.

Such analysis could be performed by a combination of one or more techniques. Such techniques include, but are not limited to, the following:

- An attacker can manually "dump" and/or disassemble listings of the data from medium 3370. This analysis is represented in Figure 67A by magnifying glass 3352A.
- An attacker can use cryptoanalytic and/or key search techniques to decrypt any encrypted data from medium 3370.
- An attacker can use automated or semi-automated disassembly tools to explore the functions of programs stored on medium 3370 by studying the operation and flow

of the assembly language representation of the programs. This analysis is represented in Figure 67A by block 3352B.

- An attacker can use software reverse-engineering tools to reconstruct high-level language representations of the programs on medium 3370, and study their functions. This analysis is represented in Figure 67A by block 3352C, producing source code 3371.
- An attacker can use software reverse-engineering tools to create an equivalent program to the programs stored on medium 3370. As the equivalent program may be in a more convenient form, possibly in a higher-level language, it may be more amenable to analysis. This analysis is also represented in Figure 67A by block 3352C, producing source code 3371.
- An attacker can use software debugging and/or simulation tools to follow and/or modify the dynamic execution of programs from medium 3370. This technique can be combined with any of the above static analysis techniques to study the program as it operates. This analysis is represented in Figure 67A by block 3352B.
- An attacker can use hardware-based debugging and/or simulation tools (e.g., an in-circuit emulator, or ICE) to follow and/or modify the dynamic execution of programs from medium 3370. This technique may be more effective

than the equivalent using software debugging and/or simulation tools because it has less potential effect on operation of the programs. This analysis is represented in Figure 67A by block 3352B.

Such analysis could provide clues and insights into the installation materials 3470, the operational materials 3472, or both.

Another attack technique could focus on the operational materials 3472 in the form in which they are installed on personal computer 3372. For example, one form of analysis might involve analyzing the on-disk copy of the installed software and/or associated data files installed on computer hard disk 3376 (see Figure 67B, block 3354). This analysis is represented in Figure 67A as a magnifying glass 3354B. Because the installed operational materials 3472 can be executed by computer 3372, the analysis need not be limited to analyzing the static information stored on hard disk 3376, but could involve performing static and/or dynamic analysis of the executing software (see Figure 67B, blocks 3356, 3358). Any of the techniques described above could be used to analyze the operational material software 3472 to yield source code or other more interpretable form 3373A and/or a memory image 3373B.

The static and/or dynamic data within RAM 3374A could be similarly analyzed (see Figure 67A, magnifying glass 3354A).

The resulting source code 3373A and/or memory image 3373B could be carefully analyzed and reviewed (see magnifying glasses 3354D, 3354E) to obtain an understanding of both the static and dynamic structure and operation of operational materials 3272. Dynamic code analysis could involve, for example, tracing, single-stepping, data, or code break points of the executing software image, using analysis techniques such as described above. The executing software could be modified dynamically (for example, by patching) during normal operation to attempt to bypass its protection mechanisms and/or to learn more about how it operates (see Figure 67B, block 3360, and the "changes" inserted into Figure 67A memory image 3373B).

A further attack technique in this example might involve comparing installed operational material 3472 software and data files among several different PPE 650 instances to identify important data structures, such as cryptographic keys (see "compare" block 3362A of Figure 67A; and Figure 67B, block 3362). The resulting list of differences 3362B could be carefully analyzed (see Figure 67A's magnifying glass 3362C) to obtain

important clues, using analysis techniques such as described above.

A further attack technique might involve comparing the memory and/or disk images of installed operational material 3472 software and data files in a single instance of PPE 650, after performing various operations using the PPE. This could serve to identify important data structures, such as cryptographic keys (see "compare" block 3362A of Figure 67A; and Figure 67B, block 3362). The resulting list of differences 3362B could be carefully analyzed (see Figure 67A's magnifying glass 3362C) to obtain important clues, using analysis techniques such as described above.

A further attack technique might involve analyzing the timing and/or order of modification to memory and/or disk images of installed operational material 3472 software and data files in a single instance of PPE 650, during the performance performing various operations using the PPE. This could serve to identify important data structures, such as cryptographic keys (see "compare" block 3362A of Figure 67A; and Figure 67B, block 3362). The resulting list of differences 3362B could be carefully analyzed (see Figure 67A's magnifying glass 3362C) to obtain

important clues, using analysis techniques such as described above.

A further attack technique might involve duplicating one installed operational material 3472 instance by copying the programs and data from one personal computer 3372B to another personal computer 3372C or emulator (see Figure 67B, block 3364, and the "copy" arrow 3364A in Figure 67A). The duplicated PPE instance could be used in a variety of ways, such as, for example, to place an impostor PPE 650 instance on-line and/or to permit further dynamic analysis.

A still additional avenue of attack might involve, for example, saving the state of a PPE 650 (see Figure 67A, block 3366B) - for example, before the expenditure of credit - and restoring the state at a subsequent time (e.g., after a payment operation occurs) (see Figure 67A, arrows 3366A, 3366C, and Figure 67B, block 3366). The stored state information 3366B may also be analyzed (see Figure 67A, magnifying glass 3354F.

No software-only tamper resistant barrier 674 can be wholly effective against all of these threats. A sufficiently powerful dynamic analysis (such as one employing an in-circuit emulator) can lay bare all of the software-based PPE 650's

secrets. Nonetheless, various techniques described below in connection with Figure 69A and following make such an analysis extremely frustrating and time consuming - increasing the "work factor" to a point where it may become commercially unfeasible to attempt to "crack" a software-based tamper resistant barrier 674.

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).

Example Techniques for Forming Software-Based Tamper Resistant Barrier

Various software protection techniques detailed above in connection with Figure 10 may provide software-based tamper resistant barrier 674 within a software-only and/or hybrid software/hardware protected processing environment 650. The following is an elaboration on those above-described techniques.

These software protection techniques may provide, for example, the following:

- An on-line registration process that results in the creation of a shared secret between the registry and the PPE 650 instance - used by the registry to create content and transactions that are meaningful only to that specific PPE instance.
- An installation program (that may be distinct from the PPE operational material software) that creates a customized installation of the PPE software unique to each PPE instance and/or associated electronic appliance 600.
- Camouflage protections that make it difficult to reverse engineer the PPE 650 operational materials during PPE operation.
- Integrity checks performed during PPE 650 operation (e.g., during on-line interactions with trusted servers) to detect compromise and minimize damage associated with any compromise.

In general, the software-based tamper resistant barrier 674 may establish "trust" primarily through uniqueness and complexity. In particular, uniqueness and customization complicate the ability of an attacker to:

- make multiple PPE instances with the same apparent identity;
- make it harder for an attacker to create a software program(s) that will defeat the tamper-resistant barrier 674 of multiple PPE instances;
- make it harder for the attacker to reverse engineer (e.g., based upon encryption so that normal debugging/emulation and other software testing tools can't easily provide access); and
- make it more difficult for an attacker to compare multiple PPE instances to determine differences between them.

In addition, the overall software-based tamper resistant barrier 674 and associated PPE system is sufficiently complex so that it is difficult to tamper with a part of it without destroying other aspects of its functionality (i.e., a "defense in depth"). Camouflaging techniques complicate an attacker's analysis through use of debugging/emulation or other software tools. For example, the PPE 650 may rewrite or overwrite memory locations immediately after using same to make their contents unavailable for scrutiny. Similarly, the PPE 650 operational software may use hardware and/or time dependent sequences to prevent emulation. Additionally, some of the PPE 650 environment code may be self-modifying. These and other techniques make it much harder to crack an individual PPE 650

instance, and more importantly - much harder to write a program that could be used to defeat security on multiple PPE instances. Because the legitimate owner/user of a particular PPE instance may be trying to attack the security of his own system, these techniques assume that individual instances may eventually be cracked and provide additional security and safeguards that prevent (or make it more difficult) for the attacker who has cracked one PPE instance to use that information successfully in cracking other PPE instances. Specifically, these security techniques make it unlikely that an attacker who has successfully cracked one or a small number of PPE instances can write a program capable of compromising the security of any arbitrary other PPE instance, for example.

Example Installation Process

Briefly, the preferred example software-based PPE 650 installation process provides the following security techniques:

- encrypted software distribution,
- installation customized on a unique instance and/or electronic appliance basis,
- encrypted on-disk form,
- installation tied to payment method,
- unique software and data layout, and
- identifiable copies.

Figure 69A shows one example technique for distributing the PPE 650 software. In this example, the PPE 650 software is distributed as two separate parts and/or media: the installation materials 3470, and the operational materials 3472. Installation materials 3470 may provide executable code and associated data structures for installing the operational materials 3472 onto a personal computer hard disk 3376, for example (see Figure 67A). The operational materials 3472 may provide executable code and associated data structures for providing protected processing environment 650 and associated software-based tamper resistant barrier 674.

In this example, installation materials 3470 and operational materials 3472 are each encrypted by a "deliverable preparation" process 3474 to provide encrypted installation materials 3470E and encrypted operational materials 3472E (the encrypted portions are indicated in Figure 69A, by cross-hatching). In this example, a small portion 3470C of the installation materials 3470 may be maintained in clear (unencrypted) form to provide an initial portion of the installation routine that may be executed without decryption. This plain text portion 3470C may, for example, provide an initial dialog, using an encrypted or other secure protocol with a trusted registry 3476 such as VDE administrator 200h for

example. This makes the distributed installation materials 3470 and operational materials 3472 meaningless and unreadable to an attacker without additional information since the entire content (except for the initial dialog with the registry 3476) is unreadable.

In this example, the "deliverable preparation" process 3474 may encrypt the installation materials 3470 and operational materials 3472 using one or more secret keys known to the registry 3476. Multiple versions of these installation materials 3470 and operational materials 3472 may be distributed using different, secret keys so that compromise of one key exposes only a subset of the software distribution to unwanted disclosure. The only non-encrypted part of the software distribution in plaintext is that portion 3470C of installation materials 3470 used to establish initial contact with the registry 3476.

The registry 3476 maintains a copy of the corresponding decryption keys within a key generation and cataloging structure 3478. It provides these keys on demand during the registration process (e.g., using a secure key exchange protocol, for example) to only legitimate users authorized to set up a new protected processing environment 650.

Figures 69B-69C show example steps that may be performed by a installation routine 3470 to install a protected processing environment 650. In this example, upon coupling the installation materials 3470 to an electronic appliance 600 such as a personal computer 3372, the appliance begins executing the unencrypted installation materials portion 3470C. This plain text portion 3470C controls appliance 600 to contact registry 3476 and establish a registry dialog (Figure 69B, block 3470(1)). The appliance 600 and the registry 3476 use a secure key exchange protocol to exchange installation keys so that the registry may deliver the appropriate installation key to the appliance (Figure 69B, block 3470(2)). Using the provided installation key(s), the appliance 600 may decrypt and run additional portions of encrypted installation materials 3470E (Figure 69B, block 3470(3) and following). Based on this additional installation program execution, appliance 600 may decrypt and install encrypted operational materials 3472E (Figure 69B, block 3470(4)).

Rather than simply installing the operational materials 3472, in one example, installation materials 3470 makes the installation different for each PPE 650 instance. For example, the installation materials 3470 may customize the installation by:

- uniquely embedding important data into the installed software,
- uniquely encrypting the installed software,
- uniquely making random changes to the installed software,
- uniquely mating the installed software with a particular electronic appliance 600,
- providing a unique static and/or dynamic layout or other structure.

Randomly Embedded Cryptographic Keys

Installation routine 3470 may, for example, modify the operational materials 3472 to customize embedded locations where critical data such as cryptographic keys are stored. These keys may be embedded into the text of the operational materials 3472 at locations that vary with each installation. In this example, the registry 3476 may choose, on a random or pseudo-random basis, at least some of the operational material 3472 locations in which a particular installation routine 3470 may embed cryptographic keys or other critical data (see Figure 69B, block 3470(5)).

The installation process for the operational software may involve decrypting its distribution (which may be the same for all

end users) and modifying it to encode the specific locations where its critical data (e.g., cryptographic keys) are stored. These keys may be embedded within the text of the program at locations that vary with every installation. The distribution of unique information into the operational software 3472 can be based on a secret key known to the registry 3476. This key may be communicated by the registry 3476 during the registration dialog using a secure key exchange. The key is shared between the registry 3476 and the PPE 650 instance, and can serve both to organize the installed PPE software, and as the basis of subsequent integrity checks.

As shown in Figure 69D, the operational materials 3472 may include embedded locations 3480(a), 3480(b), 3480(c), 3480(d), 3480(e), ... reserved for storing (embedding) critical information such as cryptographic keys. Each of these locations 3480 may initially store a random number string. In one example, the registry 3476 or installation routine 3470 performs a random operation 3482 to randomly select which subset of these locations 3480 is to be used by a particular instance for storing critical data. This selection list 3484 is applied as an input to an operation materials preparation step 3474a (part of the deliverable preparation operation 3474 shown in Figure 69A). The operation materials preparation step 3474a also

accepts, as an input, cryptographic keys from a secure key store 3486. In this example, the operation materials preparation step 3474a embeds the cryptographic keys provided by key store 3486 into the selected locations 3484 of operation materials 3472.

In accordance with one example, the random operation 3482 selects a subset that is much less than all of the possible locations 3480 - and the locations 3480 not used for storing cryptographic keys store random data instead. An attacker attempting to analyze installed operational materials 3472 won't be able to tell the difference between real cryptographic keys and random number strings inserted into a place where cryptographic keys might be stored.

In this example, the random location selection 3484 (which is unique for each installation) may itself be encrypted by block 3488 based on an installation-unique key provided by key generation block 3490 for example. The encryption key may be securely maintained at registry 3476 so that the registry may later notify the installation materials 3470 of this key - allowing the installation materials to decrypt the resulting encrypted key location block 3492 and recover listing 3484 of the subset of locations 3480 used for embedding cryptographic keys.

Embedded Customized Random Changes

Referring once again to Figure 69B, the installed operational materials 3472 may be further customized for each instance by making random changes to reserved, unused portions of the operational materials (Figure 69B, block 3470(6)). An example of this is shown in Figure 69E. In this example, the operational materials 3472 include unused, embedded random data or code portions 3494. Another technique with similar effect is shown in Figure 69F. In this example, false code sections 3496 are included within reserved areas of the operational materials 3472. These false code sections 3496 add complexity, and may also be used as a electronic "fingerprint" to help trace copies. Because the false code sections 3496 are executable program code that are never executed (or if executed perform no actual functions other than confounding analysis by, for example, creating, modifying and/or destroying data that has no impacton the operation of PPE 650 but may appear to have such an impact), they can be used to confound analysis because they may be difficult for an attacker to distinguish from true code sections. In addition other false code may have the effect of disabling the execution of PPE 650 if executed. Correspondence Between Installed Software and Appliance "Signature". Another technique that may be used during the installation routine 3470 is to customize the operational materials 3472 by embedding a

"machine signature" into the operational materials to establish a correspondence between the installed software on a particular electronic appliance 600 (Figure 69C, block 3470(7)). This technique prevents a software-based PPE 650 from being transferred from one electronic appliance 600 to another (except through the use of the appropriate secure, verified backup mechanism).

For electronic appliances 600 where it is feasible to do so, the installation procedure 3470 may determine unique information about the electronic appliance 600 (e.g., a "signature" SIG in the sense of a unique value - not necessarily a "digital signature" in the cryptographic sense). Installation routine 3470 embeds the electronic appliance "signature" SIG in the installed operational materials 3472. Upon initialization, the operational materials 3472 validate the embedded signature value against the actual electronic appliance 600 signature SIG, and may refuse to start if the comparison fails. Depending on the configuration of electronic appliance 600, the machine signature may consist, for example, of some combination of

- a hash of the ROM BIOS 658' (see Figure 69G),
- a hash of a disk defect map 3497a,
- the Ethernet (or other) network adapter 666 address,

- information written into an unused disk sector,
- information stored in a non-volatile CMOS RAM(such as used for hardware configuration data),
- information stored in non-volatile ("flash") memory (such as used for system or peripheral component "BIOS" programs) and/or
- hidden unique information placed into the root directory 3497b of the fixed disk drive 668.

Figure 69G shows an example of some of these appliance-specific signatures.

In this example, machine signature information need not be particularly large. Security is provided by hiding the machine signature rather than on any other cryptographic strength, because there is no more secure mechanism for key storage to protect it. Thus, it is satisfactory for the signature to be just large enough (e.g., two bytes) that it is unlikely to be duplicated by chance.

For some electronic appliances 600 where it can be determined that the technique is safe, an otherwise unused section of the non-volatile CMOS RAM 656a may be used to store a signature 3497d. Signature 3497d is verified against the PPE

650's internal state whenever the PPE is initialized. Signature 3497d may also be updated whenever a significant change is made to the secure database 610. If the CMOS RAM signature 3497d does not match the database value, PPE 650 may take this mismatch as an indication that a previous instance of the secure database 610 and/or PPE 650 software has been restored, and appropriate action can be taken. This mechanism thus ensures that even a bit-for-bit copy of the system's fixed disk 668 or other storage medium cannot be saved and reloaded to restore an earlier PPE 650 state. This particular technique depends upon there being an unused location available within CMOS RAM 656a, and may also require the CMOS RAM checksum algorithm to be known. An incorrect implementation could cause a subsequent reboot of electronic appliance 600 to fail because of a bad CMOS checksum, or worse, could alter some critical configuration parameter within CMOS RAM 656a so that electronic appliance 600 could not be recovered. Thus, care must be taken before modifying the contents of CMOS RAM 656a.

A still alternate technique may involve marking otherwise "good" disk sectors 3497c defective and using the sector(s) to store machine signatures and/or encryption keys. This technique ensures that a logical bit-for-bit copy of the media does not result in a usable PPE 650 instance, and also provides relatively

inaccessible and non-volatile storage for the information. Because a relatively large amount of storage space can be reserved using this technique, there is enough storage for a cryptographically strong value.

Some of the "machine signature" techniques discussed above may be problematic in some electronic appliances 600 because it may be difficult to locate appropriate appliance-unique information. For example, although in a personal computer a ROM BIOS 658' is always available, the ROM BIOS information by itself may be insufficient because it is likely to be identical for a batch of electronic appliances 600 purchased together. Identifying a network adapter 666 and determining its address is potentially difficult due to the wide variety of adapters; additionally, an electronic appliance's network address may change (although this occurrence may be infrequent). Inserting random signature values into unused bytes within the fixed disk root directory 3497b and/or partition records may trigger some virus-checking programs, and the data may be modified by defragmentation or other disk manipulation programs. Where supported, a truly unused disk sector 3497c (e.g., one that is marked "bad" even though it may still viably store information) may be used to store the machine signature. Even so, normal maintenance, upgrades or other failure recovery

procedures may disrupt a particular machine association. Since the VDE administrator 200h participates in restoring a PPE 650 based on an encrypted backup image (as described above for example in connection with Figures 39-40), the VDE administrator may establish new associations at this point to maintain correspondence between a particular PPE 650 installation and a particular electronic appliance 600.

Tie Installation To Payment Method

A still additional example technique for providing additional security is to tie a particular PPE 650 installation at registration time to a particular payment method (see Figure 69C, block 3470(8)). The registration process at installation time may thus serve to tie the PPE 650 installation to some payment method associated with the user, and to store the payment association information both within the PPE 650 instance and at the registry 3476. This technique assures that the actions of a particular PPE 650 instance are accountable to the assigned user with at least the reliability of whatever payment/credit verification technique is employed.

Install Operational Materials In Encrypted Form

Operational materials 3472 may first be customized as described above for the particular instance and/or appliance 600,

then (at least mostly) encrypted for installation into the appliance such as by storage onto disk 668 (see Figure 69C, block 3470(9)). Different installations may use different sets of decryption keys to decrypt the information once installed. Different parts of operational materials 3472 may be encrypted with different cryptographic keys to further complicate the analysis. This encryption makes analysis of the on disk form of the operational materials 3472 more difficult or infeasible.

The beginning of the resulting stored executable file may contain a small decryption program ("decryptor") that decrypts the remainder of the operational materials 3472 as they are loaded into memory. Confounding algorithms (as described below) may be used in this decryptor to make static recovery of the cryptographic keys difficult. Although the decryptor is necessarily in unencrypted form in an all-software installation without hardware support, the use of confounding algorithms to develop the associated cryptographic keys effectively requires a memory image to be captured after the program has been decrypted. Where supported (as described above), an unused and inaccessible disk sector 3497c may be used to store the decryption keys, and the operational materials 3472 may possess only the address for that particular sector. Embedding this address further complicates analysis.

Customized Layout

The installation materials 3470 may store the encrypted operational materials 3472 onto the fixed disk 668 using a customized storage layout (Figure 69C, block 3470(10)). Figure 69F, 69H, 69I and 69J shows example customized software and data layouts. In these examples, each installed instance of operational materials 3472 is different in both executable form and in data layout. These modifications make each PPE 650 instance require separate analysis in order to determine the storage locations of its critical data such as cryptographic keys. This technique is an effective counter to creation of programs that can undo the protections of an arbitrary PPE 650 instance.

Instruction sequences within the operational materials 3472 may be modified by the installation routine to change the execution flow of the executable operational materials 3472 and to alter the locations at which the software expects to locate critical data. The alterations in program flow may include customization of time-consuming confounding algorithms. The locations of the modifiable instruction sequences may be embedded within operational materials 3470, and may therefore be not directly available from an examination of the installation and/or operational materials.

Figure 69H shows one example operational materials 3472 executable code segment provided distinct processes 3498a, 3498b, 3498c, 3498d, 3498e. In this particular example, segment 3498a is executed first and segment 3498e is executed last, but the processes 3498b, 3498c and 3498d may be performed in any order (i.e., they are sequence independent processes). The installation materials 3470 may take advantage of this sequence independence by storing and/or executing them in different and/or depending upon the particular PPE instance 650. Figure 69I, for example, shows a first static layout order, and Figure 69J shows a second, different static layout order. Data elements associated with the executables may similarly be stored in different orders (as shown in Figures 69I, 69J) depending upon the particular installation.

Dynamic Protection Mechanisms

In addition to the more static protection mechanisms described above, dynamic protection mechanisms may be employed to complicate both static and dynamic analysis of the executable (executing) operational materials 3472. Such techniques include, for example:

- implementation complexity,
- immediate overwriting,
- hardware dependent sequences,

- timing dependencies,
- confounding algorithms,
- random modifications,
- dynamic load module decryption,
- on-line integrity checks,
- time integrity checks,
- machine association integrity checks,
- dynamic storage integrity checks, and
- hidden secret storage
- volatile secret storage
- internal consistency checks.

Figures 69K-69L show an example execution of operational materials 3472 that may employ some or all of these various dynamic protection mechanisms.

Upon starting execution (Figure 69K, block 3550), the installed operational materials 3472 may run initialization code as described above that is used to decrypt the stored encrypted operational materials on an "as needed" basis (Figure 69K, block 3552). This initialization code may also check the current value of the real-time clock (Figure 69K, block 3554).

Real Time Check/Validation

Operational materials 3472 may perform this time check, for example, to guard against replay attacks and to ensure that the electronic appliance 600's time is in reasonable agreement with that of the VDE administrator 200h or other trusted node.

Figure 69M shows an example sequence of steps that may be performed by the "check time" block 3554. In this example, PPE 650 uses secure communications (e.g. a cryptographic protocol) to obtain the current real time from a trusted server (Figure 69M, block 3554a). PPE 650 may next ask the user if he or she wishes to reset the electronic appliance real-time clock 528 (which may, for example, be the real-time clock module within a personal computer or the like) so it is synchronized with the trusted server's time clock.

If the user responds affirmatively, PPE 650 may reset the time clock to agree with the real-time provided by the trusted server ("yes" exit to decision block 3554b, Figure 69M, block 3554c). If the user responds that he or she does not want the real-time clock reset ("no" exit to decision block 3554b), then PPE 650 may calculate a delta value of the difference between the server's real-time clock and the electronic appliance's real-time clock 528 (Figure 69M, block 3554d). In either case, PPE 650 may store the current time Tcurrent into a non-volatile storage

location Tstore indicating the current real-time (Figure 69M, block 3554e).

Referring again to Figure 69K, PPE 650 can disable itself if there is too much (or the wrong type) of a difference between the trusted server's time and the electronic appliance's clock - since such differences can indicate replay attacks, the possibility that the PPE 650 has been restored based on a previous state, etc. For example, if desired, PPE 650 can generate a time check fail exception if the electronic appliance's real-time clock 528 disagrees with the trusted server's real-time by more than a certain amount of acceptable drift (Figure 69K, "yes" exit to decision block 3556). In the event of such an exception, PPE 650 may disable itself (Figure 69K, block 3558) and require a dialog between the user and registry 3476 (or other authority) - providing additional protection against replay attacks and also detecting clock failures that could lead to incorrect operation or incorrect charges.

Dynamic Code Decryption and Data OverWriting

Operational materials 3472 may then decrypt the next program segment dynamically (Figure 69K, block 3460. The code may be decrypted dynamically when it is needed, then re-encrypted or overwritten and discarded when not in use.

This mechanism increases the tamper-resistance of the executable code - thus providing additional tamper resistance for PPE operations. As mentioned above, different decryption keys may be required to decode different code portions, and the decryption keys can be installation-specific so that an attacker who successfully comprises the decryption key of one instance cannot use that information to compromise any other instance's decryption key(s).

Once a portion of the operational materials 3472 has been decrypted (Figure 69K, block 3560), that portion may immediately overwrite all initialization code in memory since it is no longer required (Figure 69K, block 3562). The executing operational materials 3472 may similarly overwrite all unwrapped cryptographic keys once they are no longer needed, and may also overwrite expanded key information developed by initializing the cryptographic algorithms once no longer needed. These techniques minimize the amount of time during which usable key information is available for exposure in a memory snapshot -- complicating all but the most dynamic of analysis efforts. Because all keys in permanent storage are either encrypted or otherwise camouflaged, no such treatment is required for I/O buffers.

Dynamic Check of Association Between Appliance and PPE Instance

The executing operational materials 3472 may next compare an embedded electronic appliance signature SIG' against the electronic appliance signature SIG stored in the electronic appliance itself (Figure 69K, decision block 3564). As discussed above, this technique may be used to help prevent operational materials 3472 from operating on any electronic appliance 600 other than the one it was initially installed on. PPE 650 may disable operation if this machine signature check fails ("no" exit to decision block 3564, Figure 69K; disable block 3566).

Self-Modifying and/or Hardware-Dependent Code Sequences

Executing operational materials 3472 may also employ self-modifying code sequences that cannot easily be emulated with a software debugger or single-stepping program (Figure ~~69K,~~ block 3568). These sequences may, for example, be dependent on specific models of electronic appliances 600, and may be patched into the operational materials 3472 as appropriate to installation materials 3470 based on tests performed during the installation process. Such hardware-dependent sequences may be used to ensure that critical algorithms yield different results when executed on the

proper hardware as opposed to when executed on different hardware or under software control such as in a debugger or emulator. To prevent such hardware-dependent sequences from being readily recognizable from a static examination of the code, the sequences may be constructed at run time and then invoked - so that they can be identified only by analysis of the instruction sequences actually executed.

Dynamic Timing Checks

Executing operational materials 3472 may also make dynamic timing checks on various code sequences, and refuse to operate if they do not execute within the expected interval (Figure 69K, block 3570, decision block 3572, "disable" block 3574). . An incorrect execution time suggests that the operational materials 3472 are being externally manipulated and/or analyzed or traced in some manner (e.g., by a software emulator). This technique thus provides additional protection against dynamic analysis and/or modification

The expected execution intervals associated with certain code sequences may be calculated during the installation procedure. Resulting test values may be embedded into the operational materials 3472. These timing tests may be integrated with time integrity tests and dynamic integrity checks

to make it more difficult to bypass them simply by patching out the timing check. Care should be taken to eliminate false alarms due to concurrent system activity (e.g., other tasks and/or windows)..

Figure 69N shows one example of a dynamic time check routine 3570. In this example, a test may be performed to determine whether it is time to perform another time check (decision block 3570a). For example, this test 3570a may be performed periodically and/or at the end of a time-dependent sequence as described above. If performed periodically, a counter V value may be incremented or reset to zero - readying this counter for the next performance of test 3470A (see Figure 69N, block 3570b, 3570c, 3570d).

If it is time to perform a check, PPE 650 compares the stored time value Tstore with the current time value Tcurrent, and determines whether the two values are within an acceptable range (Figure 69N, decision block 3570E). If the two values agree within an acceptable range (this range may be determined, for example, in part by the time-dependent testing described above), then PPE 650 may replace the stored time value Tstore with the current Tcurrent in preparation for the next test (Figure 69N, block 3570F). If, on the other hand, the two values

are not within an acceptable range (the "not within range" exit to decision block 3570E, Figure 69N), PPE 650 may disable operation (block 3570G) and initiate a conversation with a trusted time base or other verification facility to perform further authenticity checking (Figure 69N, block 3570H).

As Figure 69L shows, further time checks may be performed periodically and/or repeatedly based on other events (see block 3582, decision block 3584, disable block 3586).

Confounding Algorithms

The executing operational materials 3472 may also perform various confounding algorithms - computationally intensive algorithms that perform a complex operation in order to generate values required at run time (Figure 69L, block 3576). The purpose of such confounding algorithms is to make infrequently invoked steps (e.g., initialization or other steps not performed very frequently) inscrutable to an attacker who is disassembling or tracing them. Confounding algorithms may also be used for the time-dependent checking described above.

One example of such a "confounding algorithm" is a modified version of the MD5 message digest function (applied repeatedly to the same input value), which tests internally generated results of the round functions and terminates when a specific value is encountered. For example, one may make

random modifications to the confounding algorithm (for example, by adjusting the "magic constants" in MD5) until it terminates quickly enough to be useful with the desired value in some register. This adjustment may be performed beforehand to yield a prior knowledge of modifications that can then be installed differently and to each PPE 650 instance.

As one specific example, a family of 256 customized confounding algorithms could be created, each defined by a single modification of the MD5 "magic constants" (or even the input data to MD5) so that the algorithm terminates with any of 256 possible values in some register. Critical values can then be generated at run time by installing appropriate versions of the algorithm into the operational materials 3472 and assembling the values a byte at a time. Confounding algorithms may be performed in a time-dependent value as described above; their execution times may be logged and checked by PPE 650, and the PPE 650 may disable operation if the confounding algorithms run too rapidly or slowly.

Such confounding algorithms are generally infeasible to simulate by hand because they may require tens or hundreds of millions of instructions to complete. They are expensive to analyze at run time because single-stepping through the code is

time consuming (though not prohibitive, particularly if break points are set at all the possible termination tests rather than for every instruction). Although such confounding algorithms are expensive in computation time, then need not be invoked frequently - preserving efficiency.

Random Modifications to Environment State

The executing operational materials 3472 may randomly modify the PPE 650 environment state during normal operation to reflect both actual PPE 650 operations being performed and to include random modifications of data not significant to the operating PPE 650 (Figure 69L, block 3578). Such techniques help ensure that snapshots of the secure database 610 and operational materials 3472 cannot readily be compared to identify significant values and objects.

Such modifications may be based, for example, on actual random values derived from unpredictable hardware events such as disk I/O completion timing and keyboard timing. Such techniques make it infeasible to experiment with "minor" changes to the PPE 650 state even if the attacker can successfully bypass integrity checks that prevent duplicates from being made.

Load Module Dynamic Decryption & Re-Encryption

The executing operational materials 3472 may decrypt load module 1100 code dynamically as needed, and re-encrypt it or otherwise render it inscrutable when not in use (Figure 69L, block 3580). In accordance with this technique, load module executable code and/or data is decrypted dynamically when it is needed, then re-encrypted or destroyed when not in use. In addition, the location of executing load modules 1100 may be varied randomly to foil attempts to set break points within the load module. Different algorithms and a changing key may be used to further confound dynamic analysis.

Hidden Secret Storage

The source database 610 and/or parts or all of operational materials 3472 may be protected by cryptography employing keys and/or authentication values hidden in normally inaccessible locations in the appliance 600. If the key or authentication value is not available, the decryption cannot be performed, rendering PPE 650 unusable. Examples of such locations include, but are not limited to:

- Disk storage artificially marked as damaged (for the purpose of storing secrets);
- Disk storage normally reserved as alternates for sectors that may be marked as damaged;

- Disk storage normally reserved for non-general purpose use, such as sectors reserved by the manufacturer for firmware storage, for storage of statistics, or for test purposes, etc.;
- Non-volatile, writable, storage in the appliance or its components, such as that used for configuration data, device and controlled firmware, standard BIOS software, etc;
- Unused storage in files maintained by an operating system, such as the bytes between the logical end of a file and the end of its last physical sector, etc.;
- Unused storage in file system control structures, such as the bytes available to store as-yet-undefined attributes, unused storage in file allocation maps and other structures, unused storage in redundant (duplicate) file maps and directories, unused or unneeded bytes in boot records, etc.;

Storing secrets (e.g., cryptographic keys and authentication values) in these locations serves two purposes: it makes them difficult to locate by analysis of the PPE 650, and it makes them difficult to copy between one instance of the PPE and another (or to replace the PPE's contents with an earlier version of the same).

Volatile Secret Storage

The secure database 610 and/or parts or all of operational materials 3472 may be protected by cryptography employing keys and/or authentication values ("cryptovariables") that are maintained only in volatile storage during normal operation. For example, during an initialization sequence, cryptovariables can be read from permanent storage (e.g., disk), overwritten, and held only in volatile memory during system operation. During the shutdown sequence, the cryptovariables can be rewritten to permanent storage.

This provides resistance to tampering because the initialization sequence for an appliance, particularly a general-purpose computer, is typically more difficult to tamper with than is the computer during normal operation. This technique prevents the computer from itself being used to analyze the contents of permanent storage media; only by removing the media and analyzing it independently can the cryptovariables be located and extracted. This technique has the drawback of requiring the appliance's operation always to be terminated normally so that the termination sequence is guaranteed to update the permanent storage. This drawback can be ameliorated by maintaining frequent backups of the secure database 610 and/or the protected crypto-variables that can be

restored with administration by VDE administrator 200h if a disorderly termination occurs.

Dynamic Integrity Checks

In this example, operational materials 3472 may also perform a variety of dynamic integrity checks that tie an executing PPE 650 to a particular electronic appliance 600 and to guard against various forms of replay or substitution attacks. One example of a replay attack, for example, is an attack in which a user restores the PPE 650 state from an earlier backup - wiping out all recent billing records. PPE 650 includes a backup mechanism (as discussed above in connection with Figures 39 and 40) that supports restoration of previous states after system failure. Executing operational materials 3472 in this example provides certain dynamic protection mechanisms (integrity checks) that prevent such backup and restoration processes from being misused to allow such replay attacks. Such checks may identify incomplete or erroneous attempts to subvert tamper resistant barrier 672. Great care must be taken to ensure that these checks do not trigger as a result of execution or implementation error, as there is potential for significant disruption.

For example, during PPE 650 operation, the internal state of the PPE is constantly being updated. During each interaction with a trusted server, PPE 650 (and the trusted server) may test the internal state of PPE 650 to determine whether it could be derived from the internal state last seen by the trusted server for this particular PPE 650 instance. If it could not, the result may be taken as indicating a replay attack of some sort, and an appropriate action can be taken (see Figure 69L, block 3592, 3594, 3596).

For example, such a check could be implemented using a counter stored in PPE 650 and updated every time an operation is performed. If the trusted server finds the counter to be smaller than at the previous server interaction, this finding is strong evidence that a previous state of the PPE 650 environment has been restored. In practice, the check might be implemented with an obscure technique to prevent easy manipulation of the counter value. For example, the counter could be repeated hashing (e.g., with MD5) of a value that is stored redundantly in several different locations within the operational materials 3472 and secure database 610 - so that the trusted server could verify that the current value can be derived (e.g., by repeated MD5 applications) from a previous value. Such checks may limit the severity of loss resulting from off-line

manipulation of PPE 650. Because the trusted server verifies the consistency of PPE 650 at each interaction, the only loss that may occur as a result of wholesale reloading of an earlier PPE 650 state is that of content that has already been delivered by has not yet been charged for.

One example of a dynamic integrity check that executing operational materials 3472 may perform (Figure 69L, block 3588) might, for example, be the periodic verification of the integrity of the operational materials code in memory by a checksum invoked by a timer. If the timer does not tick regularly, the PPE 650 may detect it and cease to operate (see Figure 69N). This verification may counter attacks that might, for example, attempt to trick PPE 650 access methods into releasing content that has been decrypted but not electronically fingerprinted. Executing operational materials 3472 may also include numerous internal consistency checks to prevent substitution (replay) of stale database 610 records, introduction of invalid load modules 1100, external modification of the secure database 610, and so on. Such checks may be made sufficiently complex and interwoven as to make modifications likely to be detected.

When an inconsistency is detected ("yes" exit to decision block 3590, Figure 69L), PPE 650 can take appropriate action

such as locking itself up from further use until reconstructed under the trusted server's control (Figure 69L, disable block 3591). For example, PPE 650 could encrypt its secure database 610 with a new, random key, then encrypt that with the server's public key. Only the server could then arrange to reconstruct the user's instance of PPE 650.

Defense In Depth

Finally, although not a "camouflage" technique per se, the complexity of operational materials 3472 may make it difficult to understand them from the outside in. As discussed above, PPE 650 may make extensive use of RPC and coordinated work in different threads of execution. Because much of the RPC traffic may be encrypted, it will be difficult to unravel even if operational materials 3472 are heavily instrumented by the attacker. Although the cryptographic keys are, in principal, readily available in memory (e.g., because after all, the PPE 650 must be able to get them), there may be many keys and it will be difficult to identify the right one rapidly. In addition, a primary benefit to be sought by subverting protection of software-based PPE 650 installations is the ability to acquire content without paying for it - in other words, the ability to "create money". The integrity checks discussed above mean that any error in manipulating the budget and usage information data is likely to

be detected quickly. Even if the checks occur off-line without notification to any trusted server, it will make the user's PPE 650 instance effectively useless - requiring its destruction and recreation.

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 trials 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.

Such printer protections as described above may be particularly useful, for example, in the case of content providers that want to restrict or otherwise control (e.g., charge for) printing of their content. These controls can be easily enforced in the case of printers as described above having SPUs 500 (PPEs 650), but may be difficult to enforce in the case of general-purpose printers that do not have an SPU (PPE). It may be relatively easy in such environments to use printer redirectors, print output to a file, or otherwise manipulate the system's printing functions. It therefore may be advantageous to

provide a strategy that protects printed outputs in such general purpose printing environments.

So-called "intelligent" printers are capable of executing "scripts" or other programs comprising executable instructions or commands. Such "script" languages can be used to provide a degree of tamper-resistance and security without the necessity of an SPU 500 (PPE 650).

For example, it is possible to create a decryption program 3900, in PostScript or another printer control language, that can be downloaded 3801 to an associated printer 3901, creating a program 3904 stored inside the memory of printer 3901. Because PostScript, as well as other similar printer control languages, is a general-purpose programming language, such a program could decrypt (3802) an encrypted data stream 3902 sent to such a printer 3901. This approach would allow a local PPE 650 to prepare (3800) files for printing inside the PPE by creating an encrypted

file 3902 to be printed and delivering it for processing by the decryption program inside the printer.

The decryption program 3900 could be downloaded (3801) as a printer initialization activity. Using features of the PostScript or other language, and/or other mechanisms in the printer, the decryption program 3904 could be locked into the memory 3901m of printer 3901 so it could not be viewed and/or modified except with appropriate authorization. The downloaded decryption program 3904 could engage in an interactive secure protocol dialogue (3802) with PPE 650 to demonstrate that it has not been tampered with, as a precondition to creating and delivering the encrypted printable content 3902. The decryption program 3900 could also be downloaded (3801) to printer 3901 prior to printing one or more encrypted content streams 3902. The decryption program 3904 could destroy itself

after printing one or more encrypted content streams 3902 to protect
itself against viewing or tampering.

As shown in Figure 70B, the decryption program 3900 could alternatively or additionally provide a fingerprinting function--for example by selecting characters for printing from several related character fonts (3910a-3910z) in a pattern 3911 that is generated from a fingerprint key 3912 using standard cryptographic and/or steganographic techniques. Because each of the characters 3913aa, 3913ba, etc. representing the letter "A" in fonts 3910a-3910z is printed with a slightly different image, it is possible to identify the font from which each character was drawn by careful examination of the printed output, and thus to reconstruct the pattern
3911 and from that the fingerprint key 3912. There are similarly different patterns for other characters in fonts 3910a-3910z, shown as 3913ab-3913zb, etc., permitting a more efficient and/or tamper-resistant encoding of fingerprint information.

For printers that use non-executable control languages, such as PCL5,
scrambled fonts can be used without requiring that a decryption program be downloaded or otherwise installed in the printer. As

shown in figure 70, it is possible download permuted font images 3921 to such printers. Scrambled fonts 3921 are created by rearranging the character images in a normal font 3920. Such fonts allow PPE 650 to scramble the data before it is delivered for printing, so that it could not be easily interpreted except by being printed on a printer with appropriate scrambled fonts. As a simple substitution cipher, this could be inverted using automated techniques, but it provides good security against accidental disclosure. Plural scrambled fonts 3921, scrambled according to different patterns, could be resident simultaneously in the printer, and used for different sections of the printed content or different pages. Scrambled fonts 3921 could be downloaded and/or replaced dynamically in the printer differently for each page or other division in the output.

The technique of using multiple character images for fingerprinting shown in figure 70B is also applicable to printers incorporating non-executable control languages, by downloading multiple fonts 3910a-z incorporating different images for characters.

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 be adequately addressed in the past. The open, flexible control structures provided by VDE 100 addresses 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 specified as the string "fire fly" or "fire flies". Content type B is associated with the following rule set:

1. A use event that specifies that the use may only be by the software agent or a routing agent. The software agent has read only permission, the routing agent has read/write access to the information. There are no charges associated with using the information, but two meters; one by read and one by write are kept to track use of the information by various steps in the process.

2. Audits of usage are required and will be stored in object 300w under control information specified in that object.

After container 300y and its control sets are specified, they are constructed and embedded in the smart object container 300.

Container 300x is specified as a content object that is empty of content. It contains a control set that contains the following rules:

1. A write_without_billing event that specifies a meter and a general budget that limits the value of writing to $15.00.

2. Audits of usage are required and will be stored in object 300w under control information specified in that object.

3. An empty use control set that may be filled in by the owner of the information using predefined methods (method options).

After container 300x and its control sets are specified, they are constructed and embedded in the smart object container 300.

Container 300w is specified as an empty administrative object with a control set that contains the following rules:

1. A use event that specifies that the information contained in the administrative object may only be released to the creator of smart object container 300.
2. No other rules may be attached to the administrative content in container 300w.

After container 300w and its control sets are specified, they are constructed and embedded in the smart object container 300.

At this point, the smart object has been constructed and is ready to be dispatched to a remote VDE site. The smart object is sent to a remote VDE site (e.g., using electronic mail or another transport mechanism) that contains an information locator service 3012 via path 3014. The smart object is registered at the remote site 3012 for the "item locator service." The control set in container related to "item locator service" is selected and the rules contained within it activated at the remote site 3012. The

remote site 3012 then reads the contents of container 300y under the control of rule set 806f and 300y(1), and permits writes of a list of location information into container 300y pursuant to these rules. The item locator service writes a list of three items into the smart object, and then "deregisters" the smart object (now containing the location information) and sends it to a site 3016 specified in the list written to the smart object via path 3018. In this example, the user may have specified electronic mail for transport and a list of remote sites that may have the desired information is stored as a forwarding list.

The smart object 3000, upon arriving at the second remote site 3016, is registered with that second site. The site 3016 provides agent execution and software description list services compatible with VDE as a service to smart objects. It publishes these services and specifies that it requires $10.00 to start the agent and $20/piece for all information returned. The registration process compares the published service information against the rules stored within the object and determines that an acceptable overlap does not exist. Audit information for all these activities is written to the administrative object 300w. The registration process then fails (the object is not registered), and the smart object is forwarded by site 3016 to the next VDE site 3020 in the list via path 3022.

The smart object 3000, upon arriving at the third remote site 3020, is registered with that site. The site 3020 provides agent execution and software description list services compatible with VDE as a service to smart objects. It publishes these services and specifies that it requires $1.00 to start the agent and $0.50/piece for all information returned. The registration process compares the published service information against the rules stored within the object and determines that an acceptable overlap exists. The registration process creates a URT that specifies the agreed upon control information. This URT is used in conjunction with the other control information to execute the software agent under VDE control.

The agent software starts and reads its parameters out of container 300y. It then starts searching the database and obtains 253 "hits" in the database. The list of hits is written to container 300x along with a completed control set that specifies the granularity of each item and that each item costs $0.50. Upon completion of the search, the budget for use of the service is incremented by $1.00 to reflect the use charge for the service. Audit information for all these activities is written to the administrative object 300w.

The remote site 3020 returns the now "full" smart object 3000 back to the original sender (the user) at their VDE node 3010 via path 3024. Upon arrival, the smart object 3000 is registered and the database records are available. The control information specified in container 300x is now a mix of the original control information and the control information specified by the service regarding remote release of their information. The user then extracts 20 records from the smart object 3000 and has $10.00 charged to her VISA budget at the time of extraction.

In the above smart agent VDE examples, a certain organization of smart object 3000 and its constituent containers is described. Other organizations of VDE and smart object related control information and parameter data may be created and may be used for the same purposes as those ascribed to object 3000 in the above example.

Negotiation and Electronic Contracts

An electronic contract is an electronic form of an agreement including rights, restrictions, and obligations of the parties to the agreement. In many cases, electronic agreements may surround the use of digitally provided content; for example, a license to view a digitally distributed movie. It is not required, however, that an electronic agreement be conditioned on the

presence or use of electronic content by one or more parties to the agreement. In its simplest form, an electronic agreement contains a right and a control that governs how that right is used.

Electronic agreements, like traditional agreements, may be negotiated between their parties (terms and conditions submitted by one or more parties may simply be accepted (cohesion contract) by one or more other parties and/or such other parties may have the right to select certain of such terms and conditions (while others may be required)). Negotiation is defined in the dictionary as "the act of bringing together by mutual agreement." The preferred embodiment provides electronic negotiation processes by which one or more rights and associated controls can be established through electronic automated negotiation of terms. Negotiations normally require a precise specification of rights and controls associated with those rights. PERC and URT structures provide a mechanism that may be used to provide precise electronic representations of rights and the controls associated with those rights. VDE thus provides a "vocabulary" and mechanism by which users and creators may specify their desires. Automated processes may interpret these desires and negotiate to reach a common middle ground based on these desires. The results of said negotiation

may be concisely described in a structure that may be used to control and enforce the results of the electronic agreement. VDE further enables this process by providing a secure execution space in which the negotiation process(es) are assured of integrity and confidentiality in their operation. The negotiation process(es) may also be executed in such a manner that inhibits external tampering with the negotiation.

A final desirable feature of agreements in general (and electronic representations of agreements in particular) is that they be accurately recorded in a non-repudiatable form. In traditional terms, this involves creating a paper document (a contract) that describes the rights, restrictions, and obligations of all parties involved. This document is read and then signed by all parties as being an accurate representation of the agreement. Electronic agreements, by their nature, may not be initially rendered in paper. VDE enables such agreements to be accurately electronically described and then electronically signed to prevent repudiation. In addition, the preferred embodiment provides a mechanism by which human-readable descriptions of terms of the electronic contract can be provided.

VDE provides a concise mechanism for specifying control sets that are VDE site interpretable. Machine interpretable

mechanisms are often not human readable. VDE often operates the negotiation process on behalf of at least one human user. It is thus desirable that the negotiation be expressible in "human readable form." VDE data structures for objects, methods, and load modules all have provisions to specify one or more DTDs within their structures. These DTDs may be stored as part of the item or they may be stored independently. The DTD describes one or more data elements (MDE, UDE, or other related data elements) that may contain a natural language description of the function of that item. These natural language descriptions provide a language independent, human readable description for each item. Collections of items (for example, a BUDGET method) can be associated with natural language text that describes its function and forms a term of an electronically specified and enforceable contract. Collections of terms (a control set) define a contract associated with a specific right. VDE thus permits the electronic specification, negotiation, and enforcement of electronic contracts that humans can understand and adhere to.

VDE 100 enables the negotiation and enforcement of electronic contracts in several ways:

- it enables a concise specification of rights and control information that permit a common vocabulary and procedure for negotiation,

- it provides a secure processing environment within which to negotiate,

- it provides a distributed environment within which rights and control specifications may be securely distributed,

- it provides a secure processing environment in which negotiated contracts may be electronically rendered and signed by the processes that negotiate them, and

- it provides a mechanism that securely enforces a negotiated electronic contract.

Types of Negotiations

A simple form of a negotiation is a demand by one party to form an "adhesion" contract. There are few, if any, options that may be chosen by the other party in the negotiation. The recipient of the demand has a simple option; she may accept or

reject the terms and conditions (control information) in the demand. If she accepts the conditions, she is granted rights subject to the specified control information. If she rejects the conditions, she is not granted the rights. PERC and URT structures may support negotiation by demand; a PERC or control set from a PERC may be presented as a demand, and the recipient may accept or reject the demand (selecting any permitted method options if they are presented).

A common example of this type of negotiation today is the purchase of software under the terms of a "shrink-wrap license." Many widely publicized electronic distribution schemes use this type of negotiation. CompuServe is an example of an on-line service that operates in the same manner. The choice is simple: either pay the specified charge or don't use the service or software. VDE supports this type of negotiation with its capability to provide PERCs and URTs that describe rights and control information, and by permitting a content owner to provide a REGISTER method that allows a user to select from a set of predefined method options. In this scenario, the REGISTER method may contain a component that is a simplified negotiation process.

A more complex form of a negotiation is analogous to "haggling." In this scenario, most of the terms and conditions are fixed, but one or more terms (e.g., price or payment terms) are not. For these terms, there are options, limits, and elements that may be negotiated over. A VDE electronic negotiation between two parties may be used to resolve the desired, permitted, and optional terms. The result of the electronic negotiation may be a finalized set of rules and control information that specify a completed electronic contract. A simple example is the scenario for purchasing software described above adding the ability of the purchaser to select a method of payment (VISA, Mastercard, or American Express). A more complex example is a scenario for purchasing information in which the price paid depends on the amount of information about the user that is returned along with a usage audit trail. In this second example, the right to use the content may be associated with two control sets. One control set may describe a fixed ("higher") price for using the content. Another control set may describe a fixed ("lower") price for using the content with additional control information and field specifications requiring collection and return the user's personal information. In both of these cases, the optional and permitted fields and control sets in a PERC may describe the options that may be selected as part of the negotiation. To perform the negotiation, one party may propose a control set containing

specific fields, control information, and limits as specified by a PERC; the other party may pick and accept from the control sets proposed, reject them, or propose alternate control sets that might be used. The negotiation process may use the permitted, required, and optional designations in the PERC to determine an acceptable range of parameters for the final rule set. Once an agreement is reached, the negotiation process may create a new PERC and/or URT that describes the result of the negotiation. The resulting PERCs and/or URTs may be "signed" (e.g., using digital signatures) by all of the negotiation processes involved in the negotiation to prevent repudiation of the agreement at a later date.

Additional examples of negotiated elements are: electronic cash, purchase orders, purchase certificates (gift certificates, coupons), bidding and specifications, budget "rollbacks" and reconciliation, currency exchange rates, stock purchasing, and billing rates.

A set of PERCs that might be used to support the second example described above is presented in Figures 75A (PERC sent by the content owner), 75B (PERC created by user to represent their selections and rights), and 75C (PERC for controlling the negotiation process). These PERCs might be used in conjunction

with any of the negotiation process(es) and protocols described later in this section.

Figure 75A shows an example of a PERC 3100 that might be created by a content provider to describe their rights options. In this example, the PERC contains information regarding a single USE right. Two alternate control sets 3102a, 3102b are presented for this right in the example. Control set 3102a permits the use of the content without passing back information about the user, and another control set 3102b permits the use of the content and collects "response card" type information from the user. Both control sets 3102a, 3102b may use a common set of methods for most of the control information. This common control information is represented by a CSR 3104 and CS0 3106.

Control set 3102a in this PERC 3100 describes a mechanism by which the user may obtain the content without providing any information about its user to the content provider. This control set 3102a specifies a well-known vending control method and set of required methods and method options. Specifically, in this example, control set 3102a defines a BUDGET method 3108 (e.g., one of VISA, Mastercard, or American Express) and it defines a BILLING method 3110 that specifies a charge (e.g., a one-time charge of $100.00).

Control set 3102b in this PERC 3100 describes another mechanism by which the user may obtain the content. In this example, the control set 3102b specifies a different vending control method and a set of required methods and method options. This second control set 3102b specifies a BUDGET method 3112 (e.g., one of VISA, Mastercard, or American Express), a BILLING method 3116 that specifies a charge (e.g., a lesser one-time charge such as $25.00) and an AUDIT method 3114 that specifies a set of desired and required fields. The required and desired field specification 3116 may take the form of a DTD specification, in which, for example, the field names are listed.

The content creator may "prefer" one of the two control sets (e.g., control set 2) over the other one. If so, the "preferred" control set may be "offered" first in the negotiation process, and withdrawn in favor of the "non-preferred" control set if the other party to the negotiation "rejects" the "preferred" control set.

In this example, these two control sets 3102a, 3102b may share a common BUDGET method specification. The BUDGET method specification may be included in the CSR 3104 or CS0 3106 control sets if desired. Selecting control set 3102a (use with no information passback) causes a unique component assembly

to be assembled as specified by the PERC 3100. Specifically, in this example it selects the "Vending" CONTROL method 3118, the BILLING method 3110 for a $100 fixed charge, and the rest of the control information specified by CSR 3104 and CS0 3106. It also requires the user to specify her choice of acceptable BUDGET method (e.g., from the list including VISA, Mastercard, and American Express). Selecting control set 3102b assembles a different component assembly using the "Vending with 'response card'" CONTROL method 3120, the BILLING method 3116 (e.g., for a $25 fixed charge), an AUDIT method 3114 that requires the fields listed in the Required Fields DTD 3116. The process may also select as many of the fields listed in the Desired Fields DTD 3116 as are made available to it. The rest of the control information is specified by CSR 3104 and CS0 3106. The selection of control set 3102b also forces the user to specify their choice of acceptable BUDGET methods (e.g., from the list including VISA, Mastercard, and American Express).

Figure 75B shows an example of a control set 3125 that might be used by a user to specify her desires and requirements in a negotiation process. This control set has a USE rights section 3127 that contains an aggregated CSR budget specification 3129 and two optional control sets 3131a, 3131b for use of the content. Control set 3131a requires the use of a

specific CONTROL method 3133 and AUDIT method 3135. The specified AUDIT method 3135 is parameterized with a list of fields 3137 that may be released in the audit trail. Control set 3131a also specifies a BILLING method 3139 that can cost no more than a certain amount (e.g., $30.00). Control set 3131b in this example describes a specific CONTROL method 3141 and may reference a BILLING method 3143 that can cost no more than a certain amount (e.g., $150.00) if this option is selected.

Figure 75E shows a more high-level view of an electronic contract 3200 formed as a "result" of a negotiation process as described above. Electronic contract 3200 may include multiple clauses 3202 and multiple digital signatures 3204. Each clause 3202 may comprise a PERC/URT such as item 3160 described above and shown in Figure 75D. Each "clause" 3202 of electronic contract 3200 thus corresponds to a component assembly 690 that may be assembled and executed by a VDE electronic appliance 600. Just as in normal contracts, there may be as many contract clauses 3202 within electronic contract 3200 as is necessary to embody the "agreement" between the "parties." Each of clauses 3202 may have been electronically negotiated and may thus embody a part of the "agreement" (e.g., a "compromise") between the parties. Electronic contract 3200 is "self-executing" in the sense that it may be literally executed by

a machine, i.e., a VDE electronic appliance 600 that assembles component assemblies 690 as specified by various electronic clauses 3202. Electronic contract 3200 may be automatically "enforced" using the same VDE mechanisms discussed above that are used in conjunction with any component assembly 690. For example, assuming that a clause 3202(2) corresponds to a payment or BILLING condition or term, its corresponding component assembly 690 when assembled by a user's VDE electronic appliance 600 may automatically determine whether conditions are right for payment and, when they are, automatically access an appropriate payment mechanism (e.g., a virtual "credit card" object for the user) to arrange that payment to be made. As another example, assuming that electronic contract clause N 3202(N) corresponds to a user's obligation to provide auditing information to a particular VDE participant, electronic contract 3200 will cause VDE electronic appliance 600 to assemble a corresponding component assembly 690 that may, for example, access the appropriate audit trails within secure database 610 and provide them in an administrative object to the correct participant. Figure 75F shows that clause 3202(N) may, for example, specify a component assembly 690 that arranges for multiple steps in a transaction 3206 to occur. Some of these steps (e.g., step 3208(4), 3208(5)) may be conditional on a test (e.g., 3208(3)) such as, for example, whether content usage has

exceeded a certain amount, whether a certain time period has expired, whether a certain calendar date has been reached, etc.

Digital signatures 3204 shown in the Figure 75E electronic contract 3200 may comprise, for example, conventional digital signatures using public key techniques as described above. Some electronic contracts 3200 may not bear any digital signatures 3204. However, it may be desirable to require the electronic appliance 600 of the user who is a party to the electronic contract 3200 to digitally "sign" the electronic contract so that the user cannot later repudiate the contract, for evidentiary purposes, etc. Multiple parties to the same contract may each digitally "sign" the same electronic contract 3200 similarly to the way multiple parties to a contract memorialized in a written instrument use an ink pen to sign the instrument.

Although each of the clauses 3202 of electronic contract 3200 may ultimately correspond to a collection of data and code that may be executed by a PPE 650, there may in some instances be a need for rendering a human readable version of the electronic contract. This need can be accommodated by, as mentioned above, providing text within one or more DTDs associated with the component assembly or assemblies 690 used to "self-execute" the contract. Such text might, for example,

describe from a functional point of view what the corresponding electronic contract clause 3202 means or involves, and/or might describe in legally enforceable terms what the legal obligation under the contract is or represents. "Templates" (described elsewhere herein) might be used to supply such text from a text library. An expert system and/or artificial intelligence capability might be used to impose syntax rules that bind different textual elements together into a coherent, humanly readable contract document. Such text could, if necessary, be reviewed and modified by a "human" attorney in order customize it for the particular agreement between the parties and/or to add further legal obligations augmenting the "self-executing" electronic obligations embodied within and enforced by the associated component assemblies 690 executing on a VDE electronic appliance 600. Such text could be displayed automatically or on demand upon execution of the electronic contract, or it could be used to generate a printed humanly-readable version of the contract at any time. Such a document version of the electronic contract 3200 would not need to be signed in ink by the parties to the agreement (unless desired) in view of the fact that the digital signatures 3204 would provide a sufficiently secure and trusted evidentiary basis for proving the parties' mutual assent to all the terms and conditions within the contract.

In the preferred embodiment, the negotiation process executes within a PPE 650 under the direction of a further PERC that specifies the process. Figure 75C shows an example of a PERC 3150 that specifies a negotiation process. The PERC 3150 contains a single right 3152 for negotiation, with two permitted control sets 3154a, 3154b described for that right. The first control set 3154a may be used for a "trusted negotiation"; it references the desired negotiation CONTROL method ("Negotiate") 3156 and references (in fields 3157a, 3157b) two UDEs that this CONTROL method will use. These UDEs may be, for example, the PERCs 3100, 3125 shown in Figures 75A and 75B. The second control set 3154b may be used by "multiple negotiation" processes to manage the negotiation, and may provide two negotiation methods: "Negotiate1," and "Negotiate2". Both negotiation processes may be described as required methods ("Negotiate1" and "Negotiate2") 3156, 3158 that take respective PERCs 3100, 3125 as their inputs. The CONTROL method 3158 for this control set in this example may specify the name of a service that the two negotiation processes will use to communicate with each other, and may also manage the creation of the URT resulting from the negotiation.

When executed, the negotiation process(es) specified by the PERC 3150 shown in Figure 75C may be provided with the

PERCs 3100, 3125 as input that will be used as the basis for negotiation. In this example, the choice of negotiation process type (trusted or multiple) may be made by the executing VDE node. The PERC 3150 shown in Figure 75C might be, for example, created by a REGISTER method in response to a register request from a user. The process specified by this PERC 3150 may then be used by a REGISTER method to initiate negotiation of the terms of an electronic contract.

During this example negotiation process, the PERCs 3100, 3125 shown in Figures 75A and 75B act as input data structures that are compared by a component assembly created based on PERC 3150 shown in Figure 35C. The component assembly specified by the control sets may be assembled and compared, starting with required "terms," and progressing to preferred/desired "terms" and then moving on to permitted "terms," as the negotiation continues. Method option selections are made using the desired method and method options specified in the PERCs 3100, 3125. In this example, a control set for the PERC 3100 shown in Figure 75A may be compared against the PERC 3125 shown in Figure 75B. If there is a "match," the negotiation is successfully concluded and "results" are generated.

In this embodiment, the results of such negotiation will generally be written as a URT and "signed" by the negotiation process(es) to indicate that an agreement has been reached. These electronic signatures provide the means to show that a (virtual) "meeting of minds" was reached (one of the traditional legal preconditions for a contract to exist). An example of the URT 3160 that would have been created by the above example is shown in Figure 75D.

This URT 3160 (which may itself be a PERC 808) includes a control set 3162 that reflects the "terms" that were "agreed upon" in the negotiation. In this example, the "agreed upon" terms must "match" terms required by input PERCs 3100, 3125 in the sense that they must be "as favorable as" the terms required by those PERCs. The negotiation result shown includes, for example, a "negotiated" control set 3162 that in some sense corresponds to the control set 3102a of the Figure 75A PERC 3100 and to the control set 3131a of the Figure 75B control set 3125. Resulting "negotiated" control set 3162 thus includes a required BUDGET method 3164 that corresponds to the control set 3125 desired BUDGET method 3142 but which is "within" the range of control sets allowed by control set 3100 required BUDGET method 3112. Similarly, resulting negotiated control set 3162 includes a required AUDIT method 3166 that

complies with the requirements of both PERC 3100 required AUDIT method 3114 and PERC 3125 required AUDIT method 3135. Similarly, resulting negotiated control set 3162 includes a required BILLING method 3170 that "matches" or complies with each of PERC 3100 required BILLING method 3116 and PERC 3125 required BILLING method 3170.

Another class of negotiation is one under which no rules are fixed and only the desired goals are specified. The negotiation processes for this type of negotiation may be very complex. It may utilize artificial intelligence, fuzzy logic, and/or related algorithms to reach their goals. VDE supports these types of processes by providing a mechanism for concisely specifying rights, control information, fields and goals (in the form of desired rights, control information, and fields). Goals for these types of processes might be specified as one more control sets that contain specific elements that are tagged as optional, permitted, or desired.

Types of Negotiations

Negotiations in the preferred embodiment may be structured in any of the following ways:

1. shared knowledge
2. trusted negotiator

3. "zero-based" knowledge

"Shared knowledge" negotiations are based on all parties knowing all of the rules and constraints associated with the negotiation. Demand negotiations are a simple case of shared knowledge negotiations; the demander presents a list of demands that must be accepted or rejected together. The list of demands comprises a complete set of knowledge required to accept or reject each item on the list. VDE enables this class of negotiation to occur electronically by providing a mechanism by which demands may be encoded, securely passed, and securely processed between and with secure VDE subsystems using VDE secure processing, and communication capabilities. Other types of shared knowledge negotiations employed by VDE involve the exchange of information between two or more negotiating parties; the negotiation process(es) can independently determine desired final outcome(s) based on their independent priorities. The processes can then negotiate over any differences. Shared knowledge negotiations may require a single negotiation process (as in a demand type negotiation) or may involve two or more cooperative processes. Figures 76A and 76B illustrate scenarios in which one and two negotiation processes are used in a shared knowledge negotiation.

Figure 76A shows a single negotiation process 3172 that takes any number of PERCs 808 (e.g., supplied by different parties) as inputs to the negotiation. The negotiation process 3172 executes at a VDE node under supervision of "Negotiation Process Rules and Control information" that may be supplied by a further PERC (e.g., PERC 3150 shown in Figure 75C). The process 3172 generates one or more PERCs/URTs 3160 as results of the negotiation.

Figure 76B shows multiple negotiation processes 3172A-3172N each of which takes as input a PERC 808 from a party and a further PERC 3150 that controls the negotiation process, and each of which generates a negotiated "result" PERC/URT 3160 as output. Processes 3172A-3172N may execute at the same or different VDE nodes and may communicate using a "negotiation protocol."

Single and multiple negotiation processes may be used for specific VDE sites. The negotiation processes are named, and can be accessed using well known method names. PERCs and URTs may be transported in administrative or smart objects to remote VDE sites for processing at that site, as may the control PERCs and REGISTER method that controls the negotiation.

Multiple negotiation processes require the ability to communicate between these processes 3172; including secure communication between secure processes that are present at physically separate VDE sites (secure subsystems). VDE generalizes the inter-process communication into a securely provided service that can be used if the configuration requires it. The inter-process communication uses a negotiation protocol to exchange information about rule sets between processes 3172. An example of a negotiation protocol includes the following negotiation "primitives":

WANT	Want a set of terms and conditions
ACCEPT	Accept a set of terms and conditions
REJECT	Reject a set of terms and conditions
OFFER	Offer a set of terms and conditions in exchange for other terms and conditions
HAVE	Assert a set of terms and conditions are possible or desirable
QUIT	Assert the end of the negotiation without reaching an agreement

AGREEMENT Conclude the negotiation and pass the rule set for signature

The WANT primitive takes rights and control set (or parts of control sets) information, and asserts to the other process(es) 3172 that the specified terms are desired or required. Demand negotiations are a simple case of a WANT primitive being used to assert the demand. This example of a protocol may introduce a refined form of the WANT primitive, REQUIRE. In this example, REQUIRE allows a party to set terms that she decides are necessary for a contract to be formed, WANT may allow the party to set terms that are desirable but not essential. This permits a distinction between "must have" and "would like to have."

In this example, WANT primitives must always be answered by an ACCEPT, REJECT, or OFFER primitive. The ACCEPT primitive permits a negotiation process 3172 to accept a set of terms and conditions. The REJECT primitive permits a process 3172 to reject an offered set of terms and conditions. Rejecting a set of required terms and conditions may terminate the negotiation. OFFER permits a counter-offer to be made.

The HAVE, QUIT, and AGREEMENT primitives permit the negotiation protocols to pass information about rule sets. Shared knowledge negotiations may, for example, start with all negotiation processes 3172A-3172N asserting HAVE (my PERC) to the other processes. HAVE is also used when an impasse is reached and one process 3172 needs to let the other process 3172 know about permitted options. QUIT signals an unsuccessful end of the negotiation without reaching an agreement, while AGREEMENT signals a successful end of an agreement and passes the resulting "negotiated" PERC/URT 3160 to the other process(es) 3172 for signature.

In "trusted negotiator" negotiations, all parties provide their demands and preferences to a "trusted" negotiator and agree to be bound by her decision. This is similar to binding arbitration in today's society. VDE enables this mode of negotiation by providing an environment in which a "trusted" negotiation service may be created. VDE provides not only the mechanism by which demands, desires, and limits may be concisely specified (e.g., in PERCs), but in which the PERCs may be securely transferred to a "trusted" negotiation service along with a rule set that specifies how the negotiation will be conducted, and by providing a secure execution environment so that the negotiation process may not be tampered with. Trusted

negotiator services can be used at VDE sites where the integrity of the site is well known. Remote trusted negotiation services can be used by VDE sites that do not possess sufficient computing resources to execute one or more negotiation process(es); they can establish a communication link to a VDE site that provides this service and permits the service to handle the negotiation on their behalf.

"Zero-based" knowledge negotiations share some characteristics of the zero-based knowledge protocols used for authentication. It is well understood in the art how to construct a protocol that can determine if a remote site is the holder of a specific item without exchanging or exposing the item. This type of protocol can be constructed between two negotiation processes operating on at least one VDE site using a control set as their knowledge base. The negotiation processes may exchange information about their control sets, and may make demands and counter proposals regarding using their individual rule sets. For example, negotiation process A may communicate with negotiation process B to negotiate rights to read a book. Negotiation process A specifies that it will pay not more than $10.00 for rights to read the book, and prefers to pay between $5.00 and $6.00 for this right. Process A's rule set also specifies that for the $5.00 option, it will permit the release of the reader's

name and address. Process B's rule set specifies that it wants $50.00 for rights to read the book, and will provide the book for $5.50 if the user agrees to release information about himself. The negotiation might go something like this:

Process A	<--- > Process B
Want (right to read, unrestricted)	---->
	<---- Have(right to read, unrestricted, $50)
Offer (right to read, tender user info)	---->
	< ---- Have(right to read, tender user info, $5.50)
Accept(right to read, tender user info, $5.50)	----- >

In the above example, process A first specifies that it desires the right to read the book without restrictions or other information release. This starting position is specified as a rights option in the PERC that process A is using as a rule. Process B checks its rules and determines that an unrestricted right to read is indeed permitted for a price of $50. It replies to process A that these terms are available. Process A receives this reply and checks it against the control set in the PERC it uses as

a rule base. The $50 is outside the $10 limit specified for this control set, so Process A cannot accept the offer. It makes a counter offer (as described in another optional rights option) of an unrestricted right to read coupled with the release of the reader's name and address. The name and address fields are described in a DTD referenced by Process A's PERC. Process B checks its rules PERC and determines that an unrestricted right to read combined with the release of personal information is a permitted option. It compares the fields that would be released as described in the DTD provided by Process A against the desired fields in a DTD in its own PERC, and determines an acceptable match has occurred. It then sends an offer for unrestricted rights with the release of specific information for the cost of $5.50 to Process A. Process A compares the right, restrictions, and fields against its rule set and determines that $5.50 is within the range of $5-$6 described as acceptable in its rule set. It accepts the offer as made. The offer is sealed by both parties "signing" a new PERC that describes the results of the final negotiation (unrestricted rights, with release of user information, for $5.50). The new PERC may be used by the owner of Process A to read the content (the book) subject to the described terms and conditions.

Further Chain of Handling Model

As described in connection with Figure 2, there are four (4) "participant" instances of VDE 100 in one example of a VDE chain of handling and control used, for example, for content distribution. The first of these participant instances, the content creator 102, is manipulated by the publisher, author, rights owner or distributor of a literary property to prepare the information for distribution to the consumer. The second participant instance, VDE rights distributor 106, may distribute rights and may also administer and analyze customers' use of VDE authored information. The third participant instance, content user 112, is operated by users (included end-users and distributors) when they use information. The fourth participant instance, financial clearinghouse 116 enables the VDE related clearinghouse activities. A further participant, a VDE administrator, may provide support to keep VDE 100 operating properly. With appropriate authorizations and Rights Operating System components installed, any VDE electronic appliance 600 can play any or all of these participant roles.

Literary property is one example of raw material for VDE 100. To transfer this raw material into finished goods, the publisher, author, or rights owner uses tools to transform digital information (such as electronic books, databases, computer

software and movies) into protected digital packages called "objects." Only those consumers (or others along the chain of possession such as a redistributor) who receive permission from a distributor 106 can open these packages. VDE packaged content can be constrained by "rules and control information" provided by content creator 102 and/or content distributor 106—or by other VDE participants in the content's distribution pathway, i.e., normally by participants "closer" to the creation of the VDE secured package than the participant being constrained.

Once the content is packaged in an "object," the digital distribution process may begin. Since the information packages themselves are protected, they may be freely distributed on CD-ROM disks, through computer networks, or broadcast through cable or by airwaves. Informal "out of channel" exchange of protected packages among end-users does not pose a risk to the content property rights. This is because only authorized individuals may use such packages. In fact, such "out of channel" distribution may be encouraged by some content providers as a marginal cost method of market penetration. Consumers with usage authorizations (e.g., a VISA clearinghouse budget allowing a certain dollar amount of usage) may, for example, be free to license classes of out of channel

VDE protected packages provided to them, for example, by a neighbor.

To open a VDE package and make use of its content, an end-user must have permission. Distributors 106 can grant these permissions, and can very flexibly (if permitted by senior control information) limit or otherwise specify the ways in which package contents may be used. Distributors 106 and financial clearinghouses 116 also typically have financial responsibilities (they may be the same organization in some circumstances if desired). They ensure that any payments required from end-users fulfill their own and any other participant's requirements. This is achieved by auditing usage.

Distributors 106 using VDE 100 may include software publishers, database publishers, cable, television, and radio broadcasters, and other distributors of information in electronic form. VDE 100 supports all forms of electronic distribution, including distribution by broadcast or telecommunications, or by the physical transfer of electronic storage media. It also supports the delivery of content in homogeneous form, seamlessly integrating information from multiple distribution types with separate delivery of permissions, control mechanisms and content.

Distributors 106 and financial clearinghouses 116 may themselves be audited based on secure records of their administrative activities and a chain of reliable, "trusted" processes ensures the integrity of the overall digital distribution process. This allows content owners, for example, to verify that they are receiving appropriate compensation based on actual content usage or other agreed-upon bases.

Since the end-user 112 is the ultimate consumer of content in this example, VDE 100 is designed to provide protected content in a seamless and transparent way—so long as the end-user stays within the limits of the permissions she has received. The activities of end-user 112 can be metered so that an audit can be conducted by distributors 106. The auditing process may be filtered and/or generalized to satisfy user privacy concerns. For example, metered, recorded VDE content and/or appliance usage information may be filtered prior to reporting it to distributor 106 to prevent more information than necessary from being revealed about content user 112 and/or her usage.

VDE 100 gives content providers the ability to recreate important aspects of their traditional distribution strategies in electronic form and to innovatively structure new distribution mechanisms appropriate to their individual needs and

circumstances. VDE 100 supports relevant participants in the chain of distribution, and also enables their desired pricing strategies, access and redistribution permissions, usage rules, and related administrative and analysis procedures. The reusable functional primitives of VDE 100 can be flexibly combined by content providers to reflect their respective distribution objectives. As a result, content providers can feed their information into established distribution channels and also create their own personalized distribution channels.

A summary of the roles of the various participants of virtual distribution environment 100 is set forth in the table below:

Role	Description
Traditional Participants	
Content creator	Packager and initial distributor of digital information
Content owner	Owner of the digital information.
Distributors	Provide rights distribution services for budgets and/or content.
Auditor	Provides services for processing and reducing usage based audit trails.
Clearinghouse	Provides intermediate store and forward services for content and audit information. Also, typically provides a platform for other services, including third party financial providers and auditors.

Role	Description
Network provider	Provides communication services between sites and other participants.
Financial providers	Provider of third party sources of electronic funds to end-users and distributors. Examples of this class of users are VISA, American Express, or a government.
End Users	Consumers of information.
Other Participants	
Redistributor	Redistributes rights to use content based on chain of handling restrictions from content providers and/or other distributors.
VDE Administrator	Provider of trusted services for support of VDE nodes.
Independent Audit Processor	Provider of services for processing and summarizing audit trail data. Provides anonymity to end-users while maintaining the comprehensive audit capabilities required by the content providers.
Agents	Provides distributed presence for end-users and other VDE participants.

Of these various VDE participants, the "redistributor," "VDE Administrator," "independent audit processor" and "agents" are, in certain respects "new" participants that may have no counterpart in many "traditional" business models. The other VDE participants (i.e., content provider, content owner, distributors, auditor, clearinghouse, network provider and

financial providers) have "traditional" business model counterparts in the sense that traditional distribution models often included non-electronic participants performing some of the same business roles they serve in the virtual distribution environment 100.

VDE distributors 106 may also include "end-users" who provide electronic information to other end-users. For example, Figure 77 shows a further example of a virtual distribution environment 100 chain of handling and control provided by the present invention. As compared to Figure 2, Figure 77 includes a new "client administrator" participant 700. In addition, Figure 77 shows several different content users 112(1), 112(2), . . . , 112(n) that may all be subject to the "jurisdiction" of the client administrator 700. Client administrator 700 may be, for example, a further rights distributor within a corporation or other organization that distributes rights to employees or other organization participant units (such as divisions, departments, networks, and or groups, etc.) subject to organization-specific "rules and control information." The client administrator 700 may fashion rules and control information for distribution, subject to "rules and control" specified by creator 102 and/or distributor 106.

As mentioned above, VDE administrator 116b is a trusted VDE node that supports VDE 100 and keeps it operating properly. In this example, VDE administrator 116b may provide, among others, any of all of the following:

- VDE appliance initialization services
- VDE appliance reinitialization/update services
- Key management services
- "Hot lists" of "rogue" VDE sites
- Certification authority services
- Public key registration
- Client participant unit content budgets and other authorizations

All participants of VDE 100 have the innate ability to participate in any role. For example, users may gather together existing protected packages, add (create new content) packages of their own, and create new products. They may choose to serve as their own distributor, or delegate this responsibility to others. These capabilities are particularly important in the object oriented paradigm which is entering the marketplace today. The production of compound objects, object linking and embedding, and other multi-source processes will create a need for these capabilities of VDE 100. The distribution process provided by VDE 100 is symmetrical; any end-user may redistribute

information received to other end-users, provided they possess permission from and follow the rules established by the distribution chain VDE control information governing redistribution. End-users also may, within the same rules and permissions restriction, encapsulate content owned by others within newly published works and distribute these works independently. Royalty payments for the new works may be accessed by the publisher, distributors, or end-users, and may be tracked and electronically collected at any stage of the chain.

Independent financial providers can play an important role in VDE 100. The VDE financial provider role is similar to the role played by organizations such as VISA in traditional distribution scenarios. In any distribution model, authorizing payments for use of products or services and auditing usage for consistency and irregularities, is critical. In VDE 100, these are the roles filled by independent financial providers. The independent financial providers may also provide audit services to content providers. Thus, budgets or limits on use, and audits, or records of use, may be processed by (and may also be put in place by) clearinghouses 116, and the clearinghouses may then collect usage payments from users 112. Any VDE user 112 may assign the right to process information or perform services on their behalf to the extend allowed by senior control information.

The arrangement by which one VDE participant acts on behalf of another is called a "proxy." Audit, distribution, and other important rights may be "proxied" if permitted by the content provider. One special type of "proxy" is the VDE administrator 116b. A VDE administrator is an organization (which may be acting also as a financial clearinghouse 116) that has permission to manage (for example, "intervene" to reset) some portion or all of VDE secure subsystem control information for VDE electronic appliances. This administration right may extend only to admitting new appliances to a VDE infrastructure and to recovering "crashed" or otherwise inoperable appliances, and providing periodic VDE updates.

More On Object Creation, Distribution Methods, Budgets, and Audits

VDE node electronic appliances 600 in the preferred embodiment can have the ability to perform object creation, distribution, audit collection and usage control functions provided by the present invention. Incorporating this range of capabilities within each of many electronic appliances 600 provided by the preferred embodiment is important to a general goal of creating a single (or prominent) standard for electronic transactions metering, control, and billing, that, in its sum of installations, constitutes a secure, trusted, virtual

transaction/distribution management environment. If, generally speaking, certain key functions were generally or frequently missing, at least in general purpose VDE node electronic appliances 600, then a variety of different products and different standards would come forth to satisfy the wide range of applications for electronic transaction/distribution management; a single consistent set of tools and a single "rational," trusted security and commercial distribution environment will not have been put in place to answer the pressing needs of the evolving "electronic highway." Certain forms of certain electronic appliances 600 containing VDE nodes which incorporate embedded dedicated VDE microcontrollers such as certain forms of video cassette players, cable television converters and the like may not necessarily have or need full VDE capabilities. However, the preferred embodiment provides a number of distributed, disparately located electronic appliances 600 each of which desirably include authoring, distribution, extraction, audit, and audit reduction capabilities, along with object authoring capabilities.

The VDE object authoring capabilities provided by the preferred embodiment provides an author, for example, with a variety of menus for incorporating methods in a VDE object 300, including:

- menus for metering and/or billing methods which define how usage of the content portion of a VDE object is to be controlled,
- menus related to extraction methods for limiting and/or enabling users of a VDE object from extracting information from that object, and may include placing such information in a newly created and/or pre-existing VDE content container,,
- menus for specifying audit methods—that is, whether or not certain audit information is to be generated and communicated in some secure fashion back to an object provider, object creator, administrator, and/or clearinghouse, and
- menus for distribution methods for controlling how an object is distributed, including for example, controlling distribution rights of different participant's "down" a VDE chain of content container handling.

The authoring capabilities may also include procedures for distributing administrative budgets, object distribution control keys, and audit control keys to distributors and other VDE participants who are authorized to perform distribution and/or

auditing functions on behalf of the author, distributors, and/or themselves. The authoring capabilities may also include procedures for selecting and distributing distribution methods, audit methods and audit reduction methods, including for example, securely writing and/or otherwise controlling budgets for object redistribution by distributors to subsequent VDE chain of content handling participants.

The content of an object 300 created by an author may be generated with the assistance of a VDE aware application program or a non-VDE aware application program. The content of the object created by an author in conjunction with such programs may include text, formatted text, pictures, moving pictures, sounds, computer software, multimedia, electronic games, electronic training materials, various types of files, and so on, without limitation. The authoring process may encapsulate content generated by the author in an object, encrypt the content with one or more keys, and append one or more methods to define parameters of allowed use and/or required auditing of use and/or payment for use of the object by users (and/or by authorized users only). The authoring process may also include some or all aspects of distributing the object.

In general, in the preferred embodiment, an author can:

A. Specify what content is to be included in an object.

B. Specify content oriented methods including:

Information--typically abstract, promotional, identifying, scheduling, and/or other information related to the content and/or author

Content--e.g. list of files and/or other information resources containing content, time variables, etc.

C. Specify control information (typically a collection of methods related to one another by one or more permissions records, including any method defining variables) and any initial authorized user list including, for example:

Control information over Access & Extraction

Control information over Distribution

Control information over Audit Processing

A VDE node electronic appliance 600 may, for example, distribute an object on behalf of an object provider if a VDE node

receives from an object provider administrative budget information for distributing the object and associated distribution key information.

A VDE node electronic appliance 600 may receive and process audit records on behalf of an object provider if that VDE node receives any necessary administrative budget, audit method, and audit key information (used, for example, to decrypt audit trails), from the object provider. An auditing-capable VDE electronic appliance 600 may control execution of audit reduction methods. "Audit reduction" in the preferred embodiment is the process of extracting information from audit records and/or processes that an object provider (e.g., any object provider along a chain of handling of the object) has specified to be reported to an object's distributors, object creators, client administrators, and/or any other user of audit information. This may include, for example, advertisers who may be required to pay for a user's usage of object content. In one embodiment, for example, a clearinghouse can have the ability to "append" budget, audit method, and/or audit key information to an object or class or other grouping of objects located at a user site or located at an object provider site to ensure that desired audit processes will take place in a "trusted" fashion. A participant in a chain of handling of a VDE content container and/or content container

control information object may act as a "proxy" for another party in a chain of handling of usage auditing information related to usage of object content (for example a clearinghouse, an advertiser, or a party interested in market survey and/or specific customer usage information). This may be done by specifying, for that other party, budget, audit method, and/or key information that may be necessary to ensure audit information is gathered and/or provided to, in a proper manner, said additional party. For example, employing specification information provided by said other party.

Object Creation and Initial Control Structures

The VDE preferred embodiment object creation and control structure design processes support fundamental configurability of control information. This enables VDE 100 to support a full range of possible content types, distribution pathways, usage control information, auditing requirements, and users and user groups. VDE object creation in the preferred embodiment employs VDE templates whose atomic elements represent at least in part modular control processes. Employing VDE creation software (in the preferred embodiment a GUI programming process) and VDE templates, users may create VDE objects 300 by, for example, partitioning the objects, placing "meta data" (e.g., author's name, creation date, etc.) into them,

and assigning rights associated with them and/or object content to, for example, a publisher and/or content creator. When an object creator runs through this process, she normally will go through a content specification procedure which will request required data. The content specification process, when satisfied, may proceed by, for example, inserting data into a template and encapsulating the content. In addition, in the preferred embodiment, an object may also automatically register its presence with the local VDE node electronic appliance 600 secure subsystem, and at least one permissions record 808 may be produced as a result of the interaction of template instructions and atomic methods, as well as one or more pieces of control structure which can include one or more methods, budgets, and/or etc. A registration process may require a budget to be created for the object. If an object creation process specifies an initial distribution, an administrative object may also be created for distribution. The administrative object may contain one or more permission records 808, other control structures, methods, and/or load modules.

Permissions records 808 may specify various control relationships between objects and users. For example, VDE 100 supports both single access (e.g., one-to-one relationship between a user and a right user) and group access (any number of people

may be authorized as a group). A single permissions record 808 can define both single and group access. VDE 100 may provide "sharing," a process that allows multiple users to share a single control budget as a budget. Additional control structure concepts include distribution, redistribution, and audit, the latter supporting meter and budget information reduction and/or transfer. All of these processes are normally securely controlled by one or more VDE secure subsystems.

Templates and Classes

VDE templates, classes, and flexible control structures support frameworks for organizations and individuals that create, modify, market, distribute, redistribute, consume, and otherwise use movies, audio recordings and live performances, magazines, telephony based retail sales, catalogs, computer software, information databases, multimedia, commercial communications, advertisements, market surveys, infomercials, games, CAD/CAM services for numerically controlled machines, and the like. As the context surrounding these classes changes or evolves, the templates provided by the preferred embodiment of the present invention can be modified to meet these changes for broad use, or more focused activities.

VDE 100 authoring may provide three inputs into a create process: Templates, user input and object content. Templates act as a set of control instructions and/or data for object control software which are capable of creating (and/or modifying) VDE objects in a process that interacts with user instructions and provided content to create a VDE object. Templates are usually specifically associated with object creation and/or control structures. Classes represent user groups which can include "natural" groups within an organization, such as department members, specific security clearance levels, etc., or ad hoc lists of individual's and/or VDE nodes.

For example, templates may be represented as text files defining specific structures and/or component assemblies. Templates, with their structures and/or component assemblies may serve as VDE object authoring or object control applications. A creation template may consist of a number of sub-templates, which, at the lowest level, represent an "atomic level" of description of object specification. Templates may present one or more models that describe various aspects of a content object and how the object should be created including employing secure atomic methods that are used to create, alter, and/or destroy permissions records 808 and/or associated budgets, etc.

Templates, classes (including user groups employing an object under group access), and flexible control structures including object "independent" permissions records (permissions that can be associated with a plurality of objects) and structures that support budgeting and auditing as separate VDE processes, help focus the flexible and configurable capabilities inherent within authoring provided by the present invention in the context of specific industries and/or businesses and/or applications. VDE rationalizes and encompasses distribution scenarios currently employed in a wide array of powerful industries (in part through the use of application or industry specific templates). Therefore, it is important to provide a framework of operation and/or structure to allow existing industries and/or applications and/or businesses to manipulate familiar concepts related to content types, distribution approaches, pricing mechanisms, user interactions with content and/or related administrative activities, budgets, and the like.

The VDE templates, classes, and control structures are inherently flexible and configurable to reflect the breadth of information distribution and secure storage requirements, to allow for efficient adaptation into new industries as they evolve, and to reflect the evolution and/or change of an existing industry and/or business, as well as to support one or more groups of

users who may be associated with certain permissions and/or budgets and object types. The flexibility of VDE templates, classes, and basic control structures is enhanced through the use of VDE aggregate and control methods which have a compound, conditional process impact on object control. Taken together, and employed at times with VDE administrative objects and VDE security arrangements and processes, the present invention truly achieves a content control and auditing architecture that can be configured to most any commercial distribution embodiment. Thus, the present invention fully supports the requirements and biases of content providers without forcing them to fit a predefined application model. It allows them to define the rights, control information, and flow of their content (and the return of audit information) through distribution channels.

Modifying Object Content (Adding, Hiding, Modifying, Removing, and/or Extending)

Adding new content to objects is an important aspect of authoring provided by the present invention. Providers may wish to allow one or more users to add, hide, modify, remove and/or extend content that they provide. In this way, other users may add value to, alter for a new purpose, maintain, and/or

otherwise change, existing content. The ability to add content to an empty and/or newly created object is important as well.

When a provider provides content and accompanying control information, she may elect to add control information that enables and/or limits the addition, modification, hiding and/or deletion of said content. This control information may concern:

- the nature and/or location of content that may be added, hidden, modified, and/or deleted;
- portions of content that may be modified, hidden, deleted and/or added to;
- required secure control information over subsequent VDE container content usage in a chain of control and/or locally to added, hidden, and/or modified content;
- requirements that provider-specified notices and/or portions of content accompany added, hidden, deleted and/or modified content and/or the fact that said adding, hiding, modification and/or deletion occurred;
- secure management of limitations and/or requirements concerning content that may be removed, hidden and/or deleted from content,

including the amount and/or degree of addition, hiding, modification and/or deletion of content;

- providing notice to a provider or providers that modification, hiding, addition and/or deletion has occurred and/or the nature of said occurrence; and
- other control information concerned with modification, addition, hiding, and/or deleting provider content.

A provider may use this control information to establish an opportunity for other users to add value to and/or maintain existing content in a controlled way. For example, a provider of software development tools may allow other users to add commentary and/or similar and/or complementary tools to their provided objects. A provider of movies may allow commentary and/or promotional materials to be added to their materials. A provider of CAD/CAM specifications to machine tool owners may allow other users to modify objects containing instructions associated with a specification to improve and/or translate said instructions for use with their equipment. A database owner may allow other users to add and/or remove records from a

provided database object to allow flexibility and/or maintenance of the database.

Another benefit of introducing control information is the opportunity for a provider to allow other users to alter content for a new purpose. A provider may allow other users to provide content in a new setting.

To attach this control information to content, a provider may be provided with, or if allowed, design and implement, a method or methods for an object that govern addition, hiding, modification and/or deletion of content. Design and implementation of such one or more methods may be performed using VDE software tools in combination with a PPE 650. The provider may then attach the method(s) to an object and/or provide them separately. A permissions record 808 may include requirements associated with this control information in combination with other control information, or a separate permissions record 808 may be used.

An important aspect of adding or modifying content is the choice of encryption/decryption keys and/or other relevant aspects of securing new or altered content. The provider may specify in their method(s) associated with these processes a

technique or techniques to be used for creating and/or selecting the encryption/decryption keys and/or other relevant aspect of securing new and/or altered content. For example, the provider may include a collection of keys, a technique for generating new keys, a reference to a load module that will generate keys, a protocol for securing content, and/or other similar information.

Another important implication is the management of new keys, if any are created and/or used. A provider may require that such keys and reference to which keys were used must be transmitted to the provider, or she may allow the keys and/or securing strategy to remain outside a provider's knowledge and/or control. A provider may also choose an intermediate course in which some keys must be transmitted and others may remain outside her knowledge and/or control.

An additional aspect related to the management of keys is the management of permissions associated with an object resulting from the addition, hiding, modification and/or deletion of content. A provider may or may not allow a VDE chain of control information user to take some or all of the VDE rules and control information associated with granting permissions to access and/or manipulate VDE managed content and/or rules and control information associated with said resulting object.

For example, a provider may allow a first user to control access to new content in an object, thereby requiring any other user of that portion of content to receive permission from the first user. This may or may not, at the provider's discretion, obviate the need for a user to obtain permission from the provider to access the object at all.

Keys associated with addition, modification, hiding and/or deletion may be stored in an independent permissions record or records 808. Said permissions record(s) 808 may be delivered to a provider or providers and potentially merged with an existing permissions record or records, or may remain solely under the control of the new content provider. The creation and content of an initial permissions record 808 and any control information over the permissions record(s) are controlled by the method(s) associated with activities by a provider. Subsequent modification and/or use of said permission record(s) may involve a provider's method(s), user action, or both. A user's ability to modify and/or use permissions record(s) 808 is dependent on, at least in part, the senior control information associated with the permissions record(s) of a provider.

Distribution Control information

To enable a broad and flexible commercial transaction environment, providers should have the ability to establish firm control information over a distribution process without unduly limiting the possibilities of subsequent parties in a chain of control. The distribution control information provided by the present invention allow flexible positive control. No provider is required to include any particular control, or use any particular strategy, except as required by senior control information. Rather, the present invention allows a provider to select from generic control components (which may be provided as a subset of components appropriate to a provider's specific market, for example, as included in and/or directly compatible with, a VDE application) to establish a structure appropriate for a given chain of handling/control. A provider may also establish control information on their control information that enable and limit modifications to their control information by other users.

The administrative systems provided by the present invention generate administrative "events." These "events" correspond to activities initiated by either the system or a user that correspond to potentially protected processes within VDE. These processes include activities such as copying a permissions record, copying a budget, reading an audit trail record, copying a

method, updating a budget, updating a permissions record, updating a method, backing up management files, restoring management files, and the like. Reading, writing, modifying, updating, processing, and/or deleting information from any portion of any VDE record may be administrative events. An administrative event may represent a process that performs one or more of the aforementioned activities on one or more portions of one or more records.

When a VDE electronic appliance 600 encounters an administrative event, that event is typically processed in conjunction with a VDE PPE 650. As in the case of events generally related to access and/or use of content, in most cases administrative events are specified by content providers (including, for example, content creators, distributors, and/or client administrators) as an aspect of a control specified for an object, group and/or class of objects.

For example, if a user initiates a request to distribute permission to use a certain object from a desktop computer to a notebook computer, one of the administrative events generated may be to create a copy of a permissions record that corresponds to the object. When this administrative event is detected by ROS 602, an EVENT method for this type of event may be present. If

an EVENT method is present, there may also be a meter, a billing, and a budget associated with the EVENT method. Metering, billing, and budgeting can allow a provider to enable and limit the copying of a permissions record 808.

For example, during the course of processing a control program, a meter, a billing, and a budget and/or audit records may be generated and/or updated. Said audit records may record information concerning circumstances surrounding an administrative event and processing of said event. For example, an audit record may contain a reference to a user and/or system activity that initiated an event, the success or failure of processing said event, the date and/or time, and/or other relevant information.

Referring to the above example of a user with both a desktop and notebook computer, the provider of a permissions record may require an audit record each time a meter for copying said permissions record is processed. The audit record provides a flexible and configurable control and/or recording environment option for a provider.

In some circumstances, it may be desirable for a provider to limit which aspects of a control component may be modified,

updated, and/or deleted. "Atomic element definitions" may be used to limit the applicability of events (and therefore the remainder of a control process, if one exists) to certain "atomic elements" of a control component. For example, if a permissions record 808 is decomposed into "atomic elements" on the fields described in Figure 26, an event processing chain may be limited, for example, to a certain number of modifications of expiration date/time information by specifying only this field in an atomic element definition. In another example, a permissions record 808 may be decomposed into atomic elements based on control sets. In this example, an event chain may be limited to events that act upon certain control sets.

In some circumstances, it may be desirable for a provider to control how administrative processes are performed. The provider may choose to include in distribution records stored in secure database 610 information for use in conjunction with a component assembly 690 that controls and specifies, for example, how processing for a given event in relation to a given method and/or record should be performed. For example, if a provider wishes to allow a user to make copies of a permissions record 808, she may want to alter the permissions record internally. For example, in the earlier example of a user with a desktop and a notebook computer, a provider may allow a user to make copies

of information necessary to enable the notebook computer based on information present in the desktop computer, but not allow any further copies of said information to be made by the notebook VDE node. In this example, the distribution control structure described earlier would continue to exist on the desktop computer, but the copies of the enabling information passed to the notebook computer would lack the required distribution control structure to perform distribution from the notebook computer. Similarly, a distribution control structure may be provided by a content provider to a content provider who is a distributor in which a control structure would enable a certain number of copies to be made of a VDE content container object along with associated copies of permissions records, but the permissions records would be altered (as per specification of the content provider, for example) so as not to allow end-users who received distributor created copies from making further copies for distribution to other VDE nodes.

Although the preceding example focuses on one particular event (copying) under one possible case, similar processes may be used for reading, writing, modifying, updating, processing, and/or deleting information from records and/or methods under any control relationship contemplated by the present invention. Other examples include: copying a budget, copying a meter,

updating a budget, updating a meter, condensing an audit trail, and the like.

Creating Custom Methods

In the preferred embodiment of the present invention, methods may be created "at will," or aliased to another method. These two modes contribute to the superior configurability, flexibility, and positive control of the VDE distribution process. Generally, creating a method involves specifying the required attributes or parameters for the data portion of the method, and then "typing" the method. The typing process typically involves choosing one or more load modules to process any data portions of a method. In addition to the method itself, the process of method creation may also result in a method option subrecord for inclusion in, or modification of, a permissions record, and a notation in the distribution records. In addition to any "standard" load module(s) required for exercise of the method, additional load modules, and data for use with those load modules, may be specified if allowed. These event processing structures control the distribution of the method.

For example, consider the case of a security budget. One form of a typical budget might limit the user to 10Mb of decrypted data per month. The user wishes to move their rights

to use the relevant VDE content container object to their notebook. The budget creator might have limited the notebook to the same amount, half the original amount, a prorated amount based on the number of moves budgeted for an object, etc. A distribute method (or internal event processing structure) associated with the budget allows the creator of the budget to make a determination as to the methodology and parameters involved. Of course, different distribution methods may be required for the case of redistribution, or formal distribution of the method. The aggregate of these choices is stored in a permissions record for the method.

An example of the process steps used for the move of a budget record might look something like this:

1) Check the move budget (e.g., to determine the number of moves allowed)

2) Copy static fields to new record (e.g., as an encumbrance)

3) Decrement the Decr counter in the old record (the original budget)

4) Increment the Encumbrance counter in the old record

5) Write a distribution record

6) Write a Distribution Event Id to the new record

7) Increment the move meter

8) Decrement the move budget

9) Increment the Decr counter in the new record

Creating a Budget

In the preferred embodiment, to create a budget, a user manipulates a Graphical User Interface budget distribution application (e.g., a VDE template application). The user fills out any required fields for type(s) of budget, expiration cycle(s), auditor(s), etc. A budget may be specified in dollars, deutsche marks, yen, and/or in any other monetary or content measurement schema and/or organization. The preferred embodiment output of the application, normally has three basic elements. A notation in the distribution portion of secure database 610 for each budget record created, the actual budget records, and a method option record for inclusion in a permissions record. Under some circumstances, a budget process may not result in the creation of a method option since an existing method option may be being used. Normally, all of this output is protected by storage in secure database 610 and/or in one or more administrative objects.

There are two basic modes of operation for a budget distribution application in the preferred embodiment. In the

first case, the operator has an unlimited ability to specify budgets. The budgets resulting from this type of activity may be freely used to control any aspect of a distribution process for which an operator has rights, including for use with "security" budgets such as quantities limiting some aspect of usage. For example, if the operator is a "regular person," he may use these budgets to control his own utilization of objects based on a personal accounting model or schedule. If the operator is an authorized user at VISA, the resulting budgets may have broad implications for an entire distribution system. A core idea is that this mode is controlled strictly by an operator.

The second mode of operation is used to create "alias" budgets. These budgets are coupled to a preexisting budget in an operator's system. When an operator fills a budget, an encumbrance is created on the aliased budget. When these types of budgets are created, the output includes two method option subrecords coupled together: the method option subrecord for the aliased budget, and a method option subrecord for the newly created budget. In most cases, the alias budget can be used in place of the original budget if the budget creator is authorized to modify the method options within the appropriate required method record of a permissions record.

For example, assume that a user (client administrator) at a company has the company's VISA budget on her electronic appliance 600. She wants to distribute budget to a network of company users with a variety of preexisting budgets and requirements. She also wants to limit use of the company's VISA budget to certain objects. To do this, she aliases a company budget to the VISA budget. She then modifies (if so authorized) the permissions record for all objects that the company will allow their users to manipulate so that they recognize the company budget in addition to, or instead of, the VISA budget. She then distributes the new permissions records and budgets to her users. The audit data from these users is then reduced against the encumbrance on the company's VISA budget to produce a periodic billing.

In another example, a consumer wants to control his family's electronic appliance use of his VISA card, and prevent his children from playing too many video games, while allowing unlimited use of encyclopedias. In this case, he could create two budgets. The first budget can be aliased to his VISA card, and might only be used with encyclopedia objects (referenced to individual encyclopedia objects and/or to one or more classes of encyclopedia objects) that reference the aliased budget in their explicitly modified permissions record. The second budget could

be, for example, a time budget that he redistributes to the family for use with video game objects (video game class). In this instance, the second budget is a "self-replenishing" security/control budget, that allows, for example, two hours of use per day. The first budget operates in the same manner as the earlier example. The second budget is added as a new required method to permissions records for video games. Since the time budget is required to access the video games, an effective control path is introduced for requiring the second budget -- only permissions records modified to accept the family budget can be used by the children for video games and they are limited to two hours per day.

Sharing and Distributing Rights and Budgets

Move

The VDE "move" concept provided by the preferred embodiment covers the case of "friendly sharing" of rights and budgets. A typical case of "move" is a user who owns several machines and wishes to use the same objects on more than one of them. For example, a user owns a desktop and a notebook computer. They have a subscription to an electronic newspaper that they wish to read on either machine, i.e., the user wishes to move rights from one machine to the other.

An important concept within "move" is the idea of independent operation. Any electronic appliance 600 to which rights have been moved may contact distributors or clearinghouses independently. For example, the user mentioned above may want to take their notebook on the road for an extended period of time, and contact clearinghouses and distributors without a local connection to their desktop.

To support independent operation, the user should be able to define an account with a distributor or clearinghouse that is independent of the electronic appliance 600 she is using to connect. The transactions must be independently traceable and reconcilable among and between machines for both the end user and the clearinghouse or distributor. The basic operations of moving rights, budgets, and bitmap or compound meters between machines is also supported.

Redistribution

Redistribution forms a UDE middle ground between the "friendly sharing" of "move," and formal distribution. Redistribution can be thought of as "anonymous distribution" in the sense that no special interaction is required between a creator, clearinghouse, or distributor and a redistributor. Of

course, a creator or distributor does have the ability to limit or prevent redistribution.

Unlike the "move" concept, redistribution does not imply independent operation. The redistributor serves as one point of contact for users receiving redistributed rights and/or budgets, etc. These users have no knowledge of, or access to, the clearinghouse (or and/or distributor) accounts of the redistributor. The redistributor serves as an auditor for the rights and/or budgets, etc. that they redistribute, unless specifically overridden by restrictions from distributors and/or clearinghouses. Since redistributees (recipients of redistributed rights and/or budgets, etc.) would place a relatively unquantifiable workload on clearinghouses. and furthermore, since a redistributor would be placing himself at an auditable risk (responsible for all redistributed rights and/or budgets, etc.), the audit of rights, budgets, etc. of redistributees by redistributors is assumed as the default case in the preferred embodiment.

Distribution

Distribution involves three types of entity. Creators usually are the source of distribution. They typically set the control structure "context" and can control the rights which are

passed into a distribution network. Distributors are users who form a link between object (content) end users and object (content) creators. They can provide a two-way conduit for rights and audit data. Clearinghouses may provide independent financial services, such as credit and/or billing services, and can serve as distributors and/or creators. Through a permissions and budgeting process, these parties collectively can establish fine control over the type and extent of rights usage and/or auditing activities.

Encumbrance

An "encumbrance" is a special type of VDE budget. When that a budget distribution of any type occurs, an "encumbrance" may be generated. An encumbrance is indistinguishable from an original budget for right exercise (e.g., content usage payment) purposes, but is uniquely identified within distribution records as to the amount of the encumbrance, and all necessary information to complete a shipping record to track the whereabouts of an encumbrance. For right exercise purposes, an encumbrance is identical to an original budget; but for tracking purposes, it is uniquely identifiable.

In the preferred embodiment of the present invention, a Distribution Event ID will be used by user VDE nodes and by

clearinghouse services to track and reconcile encumbrances, even in the case of asynchronous audits. That is, the "new" encumbrance budget is unique from a tracking point of view, but indistinguishable from a usage point of view.

Unresolved encumbrances are a good intermediate control for a VDE distribution process. A suitable "grace period" can be introduced during which encumbrances must be resolved. If this period elapses, an actual billing or payment may occur. However, even after the interval has expired and the billing and/or payment made, an encumbrance may still be outstanding and support later reconciliation. In this case, an auditor may allow a user to gain a credit, or a user may connect to a VDE node containing an encumbered budget, and resolve an amount as an internal credit. In some cases, missing audit trails may concern a distributor sufficiently to revoke redistribution privileges if encumbrances are not resolved within a "grace period," or if there are repeated grace period violations or if unresolved encumbrances are excessively large.

Encumbrances can be used across a wide variety of distribution modes. Encumbrances, when used in concert with aliasing of budgets, opens important additional distribution possibilities. In the case of aliasing a budget, the user places

himself in the control path for an object -- an aliased budget may only be used in conjunction with permissions records that have been modified to recognize it. An encumbrance has no such restrictions.

For example, a user may want to restrict his children's use of his electronic, VDE node VISA budget. In this case, the user can generate an encumbrance on his VISA budget for the children's family alias budget, and another for his wife that is a transparent encumbrance of the original VISA budget. BigCo may use a similar mechanism to distribute VISA budget to department heads, and aliased BigCo budget to users directly.

Account Numbers and User IDs

In the preferred embodiment, to control access to clearinghouses, users are assigned account numbers at clearinghouses. Account numbers provide a unique "instance" value for a secure database record from the point of view of an outsider. From the point of view of an electronic appliance 600 site, the user, group, or group/user ids provide the unique instance of a record. For example, from the point of view of VISA, your Gold Card belongs to account number #123456789. From the point of view of the electronic appliance site (for example, a server at a corporation), the Gold card might belong

to user id 1023. In organizations which have plural users and/or user groups using a VDE node, such users and/or user groups will likely be assigned unique user IDs. differing budgets and/or other user rights may be assigned to different users and/or user groups and/or other VDE control information may be applied on a differing manner to electronic content and/or appliance usage by users assigned with different such IDs. Of course, both a clearinghouse and a local site will likely have both pieces of information, but "used data" versus the "comment data" may differ based on perspective.

In the preferred embodiment case of "move," an account number stored with rights stays the same. In the preferred embodiment of other forms of distribution, a new account number is required for a distributee. This may be generated automatically by the system, or correspond to a methodology developed by a distributor or redistributor. Distributors maintain account numbers (and associated access secrets) in their local name services for each distributee. Conversely, distributees' name services may store account numbers based on user id for each distributor. This record usually is moved with other records in the case of move, or is generated during other forms of distribution.

Organizations (including families) may automatically assign unique user IDs when creating control information (e.g., a budget) for a new user or user group.

Requirements Record

In order to establish the requirements, and potentially options, for exercising a right associated with a VDE content container object before one or more required permissions records are received for that object, a requirements record may exist in the private header of such an object. This record will help the user establish what they have, and what they need from a distributor prior to forming a connection. If the requirements or possibilities for exercising a particular right have changed since such an object was published, a modified requirements record may be included in a container with an object (if available and allowed), or a new requirements record may be requested from a distributor before registration is initiated. Distributors may maintain "catalogs" online, and/or delivered to users, of collections of requirements records and/or descriptive information corresponding to objects for which they may have ability to obtain and/or grant rights to other users.

Passing an Audit

In the preferred embodiment of VDE there may be at least two types of auditing. In the case of budget distribution, billing records that reflect consumption of a budget generally need to be collected and processed. In the case of permissions distribution, usage data associated with an object are also frequently required.

In order to effect control over an object, a creator may establish the basic control information associated with an object. This is done in the formulation of permissions, the distribution of various security, administrative and/or financial budgets, and the level of redistribution that is allowed, etc. Distributors (and redistributors) may further control this process within the rights, budgets, etc. (senior control information) they have received.

For example, an object creator may specify that additional required methods may be added freely to their permissions records, establish no budget for this activity, and allow unlimited redistribution of this right. As an alternative example, a creator may allow moving of usage rights by a distributor to half a dozen subdistributors, each of whom can distribute 10,000 copies, but with no redistribution rights being allowed to be allocated to subdistributors' (redistributors') customers. As another example, a creator may authorize the moving of usage rights to only 10

VDE nodes, and to only one level of distribution (no redistribution). Content providers and other contributors of control information have the ability through the use of permissions records and/or component assemblies to control rights other users are authorized to delegate in the permissions records they send to those users, so long as such right to control one, some, or all such rights of other users is either permitted or restricted (depending on the control information distribution model). It is possible and often desirable, using VDE, to construct a mixed model in which a distributor is restricted from controlling certain rights of subsequent users and is allowed to control other rights. VDE control of rights distribution in some VDE models will in part or whole, at least for certain one or more "levels" of a distribution chain, be controlled by electronic content control information providers who are either not also providers of the related content or provide only a portion of the content controlled by said content control information. for example, in certain models, a clearinghouse might also serve as a rights distribution agent who provides one or more rights to certain value chain participants, which one or more rights may be "attached" to one or more rights to use the clearinghouse's credit (if said clearinghouse is, at least in part, a financial clearinghouse (such a control information provider may alternatively, or in addition, restrict other users' rights.

A content creator or other content control information provider may budget a user (such as a distributor) to create an unlimited number of permissions records for a content object, but revoke this right and/or other important usage rights through an expiration/termination process if the user does not report his usage (provide an audit report) at some expected one or more points in time and/or after a certain interval of time (and/or if the user fails to pay for his usage or violates other aspects of the agreement between the user and the content provider). This termination (or suspension or other specified consequence) can be enforced, for example, by the expiration of time-aged encryption keys which were employed to encrypt one or more aspects of control information. This same termination (or other specified consequence such as budget reduction, price increase, message displays on screen to users, messages to administrators, etc.) can also be the consequence of the failure by a user or the users VDE installation to complete a monitored process, such as paying for usage in electronic currency, failure to perform backups of important stored information (e.g., content and/or appliance usage information, control information, etc.), failure to use a repeated failure to use the proper passwords or other identifiers, etc.).

Generally, the collection of audit information that is collected for reporting to a certain auditor can be enforced by expiration and/or other termination processes. For example, the user's VDE node may be instructed (a) from an external source to no longer perform certain tasks, (b) carries within its control structure information informing it to no longer perform certain tasks, or (c) is elsewise no longer able to perform certain tasks. The certain tasks might comprise one or more enabling operations due to a user's (or installation's) failure to either report said audit information to said auditor and/or receive back a secure confirmation of receipt and/or acceptance of said audit information. If an auditor fails to receive audit information from a user (or some other event fails to occur or occur properly), one or more time-aged keys which are used, for example, as a security component of an embodiment of the present invention, may have their aging suddenly accelerated (completed) so that one or more processes related to said time-aged keys can no longer be performed.

Authorization Access Tags and Modification Access Tags

In order to enable a user VDE installation to pass audit information to a VDE auditing party such as a Clearinghouse, VDE allows a VDE auditing party to securely, electronically communicate with the user VDE installation and to query said

installation for certain or all information stored within said installation's secure sub-system, depending on said auditing party's rights (said party shall normally be unable to access securely stored information that said party is not expressly authorized to access, that is one content provider will normally not be authorized to access content usage information related to content provided by a different content provider). The auditing party asserts a secure secret (e.g., a secure tag) that represents the set of rights of the auditor to access certain information maintained by said subsystem. If said subsystem validates said tag, the auditing party may then receive auditing information that it is allowed to request and receive.

Great flexibility exists in the enforcement of audit trail requirements. For example, a creator (or other content provider or control information provider or auditor in an object's or audit report's chain of handling) may allow changes by an auditor for event trails, but not allow anyone but themselves to read those trails, and limit the redistribution of this right to, for example, six levels. Alternatively, a creator or other controlling party may give a distributor the right to process, for example, 100,000 audit records (and/or, for example, the right to process 12 audit records from a given user) before reporting their usage. If a creator or other controlling party desires, he may allow (and/or require)

separate (and containing different, a subset of, overlapping, or the same information) audit "packets" containing audit information, certain of said audit information to be processed by a distributor and certain other of said audit information to be passed back to the creator and/or other auditors (each receiving the same, overlapping, a subset of, or different audit information). Similarly, as long as allowed by, for example, an object creator, a distributor (or other content and/or control information provider) may require audit information to be passed back to it, for example, after every 50,000 audit records are processed (or any other unit of quantity and/or after a certain time interval and/or at a certain predetermined date) by a redistributor. In the preferred embodiment, audit rules, like other control structures, may be stipulated at any stage of a distribution chain of handling as long as the right to stipulate said rules has not been restricted by a more "senior" object and/or control information distributing (such as an auditing) participant.

Audit information that is destined for different auditors may be encrypted by different one or more encryption keys which have been securely provided by each auditor's VDE node and communicated for inclusion in a user's permissions record(s) as a required step, for example, during object registration. This can

provide additional security to further ensure (beyond the use of passwords and/or other identification information and other VDE security features) that an auditor may only access audit information to which he is authorized. In one embodiment, encrypted (and/or unencrypted) "packets" of audit information (for example, in the form of administrative objects) may be bound for different auditors including a clearinghouse and/or content providers and/or other audit information users (including, for example, market analysts and/or list purveyors). The information may pass successively through a single chain of handling, for example, user to clearinghouse to redistributor to distributor to publisher/object creator, as specified by VDE audit control structures and parameters. Alternatively, encrypted (or, normally less preferably, unencrypted) audit packets may be required to be dispersed directly from a user to a plurality of auditors, some one or more who may have the responsibility to "pass along" audit packets to other auditors. In another embodiment, audit information may be passed, for example, to a clearinghouse, which may then redistribute all and/or appropriate subsets of said information (and/or some processed result) to one or more other parties, said redistribution employing VDE secure objects created by said clearinghouse.

An important function of an auditor (receiver of audit information) is to pass administrative events back to a user VDE node in acknowledgement that audit information has been received and/or "recognized." In the preferred embodiment, the receipt and/or acceptance of audit information may be followed by two processes. The first event will cause the audit data at a VDE node which prepared an audit report to be deleted, or compressed into, or added to, one or more summary values. The second event, or set of events, will "inform" the relevant security (for example, termination and/or other consequence) control information (for example, budgets) at said VDE node of the audit receipt, modify expiration dates, provide key updates, and/or etc. In most cases, these events will be sent immediately to a site after an audit trail is received. In some cases, this transmission may be delayed to, for example, first allow processing of the audit trail and/or payment by a user to an auditor or other party.

In the preferred embodiment, the administrative events for content objects and independently distributed methods/component assemblies are similar, but not necessarily identical. For example, key updates for a budget may control encryption of a billing trail, rather than decryption of object content. The billing trail for a budget is in all respects a method event trail. In one embodiment, this trail must include sufficient

references into distribution records for encumbrances to allow reconciliation by a clearinghouse. This may occur, for example, if a grace period elapses and the creator of a budget allows unresolved encumbrances to ultimately yield automatic credits if an expired encumbrance is "returned" to the creator.

Delivery of audit reports through a path of handling may be in part insured by an inverse (return of information) audit method. Many VDE methods have at least two pieces: a portion that manages the process of producing audit information at a user's VDE node; and a portion that subsequently acts on audit data. In an example of the handling of audit information bound for a plurality of auditors, a single container object is received at a clearinghouse (or other auditor). This container may contain (a) certain encrypted audit information that is for the use of the clearinghouse itself, and (b) certain other encrypted audit information bound for other one or more auditor parties. The two sets of information may have the same, overlapping and in part different, or entirely different, information content. Alternatively, the clearinghouse VDE node may be able to work with some or all of the provided audit information. The audit information may be, in part, or whole, in some summary and/or analyzed form further processed at the clearinghouse and/or may be combined with other information to form a, at least in part,

derived set of information and inserted into one or more at least in part secure VDE objects to be communicated to said one or more (further) auditor parties. When an audit information container is securely processed at said clearinghouse VDE node by said inverse (return) audit method, the clearinghouse VDE node can create one or more VDE administrative objects for securely carrying audit information to other auditors while separately processing the secure audit information that is specified for use by said clearinghouse. Secure audit processes and credit information distribution between VDE participants normally takes place within the secure VDE "black box," that is processes are securely processed within secure VDE PPE650 and audit information is securely communicated between the VDE secure subsystems of vDE participants employing VDE secure communication techniques (e.g., public key encryption, and authentication).

This type of inverse audit method may specify the handling of returned audit information, including, for example, the local processing of audit information and/or the secure passing along of audit information to one or more auditor parties. If audit information is not passed to one or more other auditor parties as may be required and according to criteria that may have been set by said one or more other auditor parties and/or

content providers and/or control information providers during a permissions record specification and/or modification process, the failure to, for example, receive notification of successful transfer of required audit information by an auditor party, e.g., a content provider, can result in the disablement of at least some capability of the passing through party's VDE node (for example, disablement of the ability to further perform certain one or more VDE managed business functions that are related to object(s) associated with said audit or party). In this preferred embodiment example, when an object is received by an auditor, it is automatically registered and permissions record(s) contents are entered into the secure management database of the auditor's VDE node.

One or more permissions records that manage the creation and use of an audit report object (and may manage other aspects of object use as well) may be received by a user's system during an audit information reporting exchange (or other electronic interaction between a user and an auditor or auditor agent). Each received permissions record may govern the creation of the next audit report object. After the reporting of audit information, a new permissions record may be required at a user's VDE node to refresh the capability of managing audit report creation and audit information transfer for the next audit

reporting cycle. In our above example, enabling an auditor to supply one or more permissions records to a user for the purpose of audit reporting may require that an auditor (such as a clearinghouse) has received certain, specified permissions records itself from "upstream" auditors (such as, for example, content and/or other content control information providers). Information provided by these upstream permissions records may be integrated into the one or more permissions records at an auditor VDE (e.g., clearinghouse) installation that manage the permissions record creation cycle for producing administrative objects containing permissions records that are bound for users during the audit information reporting exchange. If an upstream auditor fails to receive, and/or is unable to process, required audit information, this upstream auditor may fail to provide to the clearinghouse (in this example) the required permissions record information which enables a distributor to support the next permission record creation/auditing cycle for a given one or more objects (or class of objects). As a result, the clearinghouse's VDE node may be unable to produce the next cycle's permissions records for users, and/or perform some other important process. This VDE audit reporting control process may be entirely electronic process management involving event driven VDE activities at both the intended audit information receiver and

sender and employing both their secure PPE650 and secure VDE communication techniques.

In the preferred embodiment, each time a user registers a new object with her own VDE node, and/or alternatively, with a remote clearinghouse and/or distributor VDE node, one or more permissions records are provided to, at least in part, govern the use of said object. The permissions records may be provided dynamically during a secure UDE registration process (employing the VDE installation secure subsystem), and/or may be provided following an initial registration and received at some subsequent time, e.g. through one or more separate secure VDE communications, including, for example, the receipt of a physical arrangement containing or otherwise carrying said information. At least one process related to the providing of the one or more permissions records to a user can trigger a metering event which results in audit information being created reflecting the user's VDE node's, clearinghouse's, and/or distributor's permissions records provision process. This metering process may not only record that one or more permissions records have been created. It may also record the VDE node name, user name, associated object identification information, time, date, and/or other identification information. Some or all of this information can become part of audit information securely reported by a

clearinghouse or distributor, for example, to an auditing content creator and/or other content provider. This information can be reconciled by secure VDE applications software at a receiving auditor's site against a user's audit information passed through by said clearinghouse or distributor to said auditor. For each metered one or more permissions records (or set of records) that were created for a certain user (and/or VDE node) to manage use of certain one or more VDE object(s) and/or to manage the creation of VDE object audit reports, it may be desirable that an auditor receive corresponding audit information incorporated into an, at least in part, encrypted audit report. This combination of metering of the creation of permissions records; secure, encrypted audit information reporting processes; secure VDE subsystem reconciliation of metering information reflecting the creation of registration and/or audit reporting permissions with received audit report detail; and one or more secure VDE installation expiration and/or other termination and/or other consequence processes; taken together significantly enhances the integrity of the VDE secure audit reporting process as a trusted, efficient, commercial environment.

Secure Document Management Example

VDE 100 may be used to implement a secure document management environment. The following are some examples of how this can be accomplished.

In one example, suppose a law firm wants to use VDE 100 to manage documents. In this example, a law firm that is part of a litigation team might use VDE in the following ways:

1. to securely control access to, and/or other usage of, confidential client records,
2. to securely control access, distribution, and/or other rights to documents and memoranda created at the law firm,
3. to securely control access and other use of research materials associated with the case,
4. to securely control access and other use, including distribution of records, documents, and notes associated with the case,
5. to securely control how other firms in the litigation team may use, including change, briefs that have been distributed for comment and review,
6. to help manage client billing.

The law firm may also use VDE to electronically file briefs with the court (presuming the court is also VDE capable) including providing secure audit verification of the ID (e.g., digital signature) of filers and other information pertinent to said filing procedure.

In this example, the law firm receives in VDE content containers documents from their client's VDE installation secure subsystem(s). Alternatively, or in addition, the law firm may receive either physical documents which may be scanned into electronic form, and/or they receive electronic documents which have not yet been placed in VDE containers. The electronic form of a document is stored as a VDE container (object) associated with the specific client and/or case. The VDE container mechanism supports a hierarchical ordering scheme for organizing files and other information within a container; this mechanism may be used to organize the electronic copies of the documents within a container, A VDE container is associated with specific access control information and rights that are described in one or more permissions control information sets (PERCs) associated with that container. In this example, only those members of the law firm who possess a VDE instance, an appropriate PERC, and the VDE object that contains the desired document, may use the document. Alternatively or in addition, a

law firm member may use a VDE instance which has been installed on the law firm's network server. In this case, the member must be identified by an appropriate PERC and have access to the document containing VDE object (in order to use the server VDE installation). Basic access control to electronic documents is enabled using the secure subsystem of one or more user VDE installations.

VDE may be used to provide basic usage control in several ways. First, it permits the "embedding" of multiple containers within a single object. Embedded objects permit the "nesting" of control structures within a container. VDE also extends usage control information to an arbitrary granular level (as opposed to a file based level provided by traditional operating systems) and provides flexible control information over any action associated with the information which can be described as a VDE controlled process. For example, simple control information may be associated with viewing the one or more portions of documents and additional control information may be associated with editing, printing and copying the same and/or different one or more portions of these same documents.

In this example, a "client" container contains all documents that have been provided by the client (documents

received in other containers can be securely extracted and embedded into the VDE client container using VDE extraction and embedding capabilities). Each document in this example is stored as an object within the parent, client VDE container. The "client" container also has several other objects embedded within it; one for each attorney to store their client notes, one (or more) for research results and related information, and at least one for copies of letters, work papers, and briefs that have been created by the law firm. The client container may also contain other information about the client, including electronic records of billing, time, accounting, and payments. Embedding VDE objects within a parent VDE content container provides a convenient way to securely categorize and/or store different information that shares similar control information. All client provided documents may, for example, be subject to the same control structures related to use and non-disclosure. Attorney notes may be subject to control information, for example, their use may be limited to the attorney who created the notes and those attorneys to whom such creating attorney expressly grants access rights. Embedded containers also provide a convenient mechanism to control collections of dissimilar information. For example, the research object(s) may be stored in the form of (or were derived from) VDE "smart objects" that contain the results of research performed by that object. Research results related to

one aspect of the case retrieved from a VDE enabled LEXIS site might be encapsulated as one smart object; the results of another session related to another (or the same) aspect of the case may be encapsulated as a different object. Smart objects are used in this example to help show that completely disparate and separately delivered control information may be incorporated into a client container as desired and/or required to enforce the rights of providers (such as content owners).

Control structures may be employed to manage any variety of desired granularities and/or logical document content groupings; a document, page, paragraph, topically related materials, etc. In this example, the following assumptions are made: client provided documents are controlled at the page level, attorney notes are controlled at the document level on an attorney by attorney basis, court filings and briefs are controlled at a document level, research information is controlled at whatever level the content provider specifies at the time the research was performed, and certain highly confidential information located in various of the above content may be identified as subject to display and adding comments only, and only by the lead partner attorneys, with only the creator and/or embedder of a given piece of content having the right to be otherwise used (printed, extracted, distributed, etc).

In general, container content in this example is controlled with respect to distribution of rights. This control information are associated at a document level for all internally generated documents, at a page level for client level documents, and at the level specified by the content provider for research documents.

VDE control information can be structured in either complex or simple structures, depending on the participant's desires. In some cases, a VDE creator will apply a series of control structure definitions that they prefer to use (and that are supported by the VDE application managing the specification of rules and control information, either directly, or through the use of certified application compatible VDE component assemblies.

In this example, the law firm sets up a standard VDE client content container for a new client at the time they accept the case. A law firm VDE administrator would establish a VDE group for the new client and add the VDE IDs of the attorneys at the firm that are authorized to work on the case, as well as provide, if appropriate, one or more user template applications. These templates provide, for example, one or more user interfaces and associated control structures for selection by users of additional and/or alternative control functions (if allowed by senior control information), entry of control parameter data,

and/or performing user specific administrative tasks. The administrator uses a creation tool along with a predefined creation template to create the container. This creation template specifies the document usage (including distribution control information) for documents as described above. Each electronic document from the client (including letters, memoranda, E-mail, spreadsheet, etc.) are then added to the container as separate embedded objects. Each new object is created using a creation template that satisfies that the default control structures specified with the container as required for each new object of a given type.

As each attorney works on the case, they may enter notes into an object stored within the client's VDE container. These notes may be taken using a VDE aware word processor already in use at the law firm. In this example, a VDE redirector handles the secure mapping of the word processor file requests into the VDE container and its objects through the use of VDE control processes operating with one or more VDE PPEs. Attorney note objects are created using the default creation template for the document type with assistance from the attorney if the type cannot be automatically determined from the content. This permits VDE to automatically detect and protect

the notes at the predetermined level, e.g. document, page, paragraph.

Research can be automatically managed using VDE. Smart objects can be, used to securely search out, pay for if necessary, and retrieve information from VDE enabled information resources on the information highway.

Examples of such resources might include LEXIS, Westlaw, and other related legal databases. Once the information is retrieved, it may be securely embedded in the VDE content client container. If the smart object still contains unreleased information, the entire smart object may be embedded in the client's VDE container. This places the unreleased information under double VDE control requirements: those associated with releasing the information from smart object (such as payment and/or auditing requirements) and those associated with access to, or other usage of, client information of the specified type.

Briefs and other filings may be controlled in a manner similar to that for attorney notes. The filings may be edited using the standard word processors in the law firm; with usage control structures controlling who may review, change, and/or

add to the document (or, in a more sophisticated example, a certain portion of said document). VDE may also support electronic filing of briefs by providing a trusted source for time/date stamping and validation of filed documents.

When the client and attorney want to exchange confidential information over electronic mail or other means, VDE can play an important role in ensuring that information exchanged under privilege, properly controlled, and not inappropriately released and/or otherwise used. The materials (content) stored in a VDE content container object will normally be encrypted. Thus wrapped, a VDE object may be distributed to the recipient without fear of unauthorized access and/or other use. The one or more authorized users who have received an object are the only parties who may open that object and view and/or manipulate and/or otherwise modify its contents and VDE secure auditing ensures a record of all such user content activities. VDE also permits the revocation of rights to use client/attorney privileged information if such action becomes necessary, for example, after an administrator review of user usage audit information.

Large Organization Example

In a somewhat more general example, suppose an organization (e.g., a corporation or government department) with thousands of employees and numerous offices disposed throughout a large geographic area wishes to exercise control over distribution of information which belongs to said organization (or association). This information may take the form of formal documents, electronic mail messages, text files, multimedia files, etc,, which collectively are referred to as "documents."

Such documents may be handled by people (referred to as "users") and/or by computers operating on behalf of users. The documents may exist both in electronic form for storage and transmission and in paper form for manual handling.

These documents may originate wholly within the organization, or may be created, in whole or in part, from information received from outside the organization. Authorized persons within the organization may choose to release documents, in whole or in part, to entities outside the organization. Some such entities may also employ VDE 100 for document control, whereas others may not.

Document Control Policies

The organization as a whole may have a well-defined policy for access control to, and/or other usage control of documents. This policy may be based on a "lattice model" of information flow, in which documents are characterized as having one or more hierarchical "classification" security attributes 9903 and zero or more non-hierarchical "compartment" security attributes, all of which together comprise a sensitivity security attribute.

The classification attributes may designate the overall level of sensitivity of the document as an element of an ordered set. For example, the set "unclassified," "confidential," "secret," "top secret" might be appropriate in a government setting, and the set "public," "internal," "confidential," "registered confidential" might be appropriate in a corporate setting.

The compartment attributes may designate the document's association with one or more specific activities within the organization, such as departmental subdivisions (e.g., "research," "development," "marketing") or specific projects within the organization.

Each person using an electronic appliance 600 would be assigned, by an authorized user, a set of permitted sensitivity attributes to designate those documents, or one or more portions of certain document types, which could be processed in certain one or more ways, by the person's electronic appliance. A document's sensitivity attribute would have to belong to the user's set of permitted sensitivity values to be accessible.

In addition, the organization may desire to permit users to exercise control over specific documents for which the user has some defined responsibility. As an example, a user (the "originating user") may wish to place an "originator controlled" ("ORCON") restriction on a certain document, such that the document may be transmitted and used only by those specific other users whom he designates (and only in certain, expressly authorized ways). Such a restriction may be flexible if the "distribution list" could be modified after the creation of the document, specifically in the event of someone requesting permission from the originating user to transmit the document outside the original list of authorized recipients. The originating user may wish to permit distribution only to specific users, defined groups of users, defined geographic areas, users authorized to act in specific organizational roles, or a combination of any or all such attributes.

In this example, the organization may also desire to permit users to define a weaker distribution restriction such that access to a document is limited as above, but certain or all information within the document may be extracted and redistributed without further restriction by the recipients.

The organization and/or originating users may wish to know to what uses or geographic locations a document has been distributed. The organization may wish to know where documents with certain protection attributes have been distributed, for example, based on geographic information stored in site configuration records and/or name services records.

A user may wish to request a "return receipt" for a distributed document, or may wish to receive some indication of how a document has been handled by its recipients (e.g., whether it has been viewed, printed, edited and/or stored), for example, by specifying one or more audit requirements (or methods known to have audit requirements) in a PERC associated with such document(s).

User Environment

In an organization (or associateion) such as that described above, users may utilize a variety of electronic appliances 600 for

processing and managing documents. This may include personal computers, both networked and otherwise, powerful single-user workstations, and servers or mainframe computers. To provide support for the control information described in this example, each electronic appliance that participates in use and management of VDE-protected documents may be enhanced with a VDE secure subsystem supporting an SPE 503 and/or HPE 655.

In some organizations, where the threats to secure operation are relatively low, an HPE 655 may suffice. In other organizations (e.g., government defense), it may be necessary to employ an SPE 503 in all situations where VDE-protected documents are processed. The choice of enhancement environment and technology may be different in different of the organization. Even if different types of PPE 650 are used within an organization to serve different requirements, they may be compatible and may operate on the same types (or subsets of types) of documents.

Users may employ application programs that are customized to operate in cooperation with the VDE for handling of VDE-protected documents. Examples of this may include VDE-aware document viewers, VDE aware electronic mail

systems, and similar applications. Those programs may communicate with the PPE 650 component of a user's electronic appliance 600 to make VDE-protected documents available for use while limiting the extent to which their contents may be copied, stored, viewed, modified, and/or transmitted and/or otherwise further distributed outside the specific electronic appliance.

Users may wish to employ commercial, off-the-shelf ("COTS") operating systems and application programs to process the VDE-protected documents. One approach to permit the use of COTS application programs and operating systems would be to allow such use only for documents without restrictions on redistribution. The standard VDE operating system redirector would allow users to access VDE-protected documents in a manner equivalent to that for files. In such an approach, however, a chain of control for metering and/or auditing use may be "broken" to some extent at the point that the protected object was made available to the COTS application. The fingerprinting (watermarking) techniques of VDE may be used to facilitate further tracking of any released information.

A variety of techniques may be used to protect printing of protected documents, such as, for example: server-based decryption engines, special fonts for "fingerprinting," etc.

Another approach to supporting COTS software would use the VDE software running on the user's electronic appliance to create one or more "virtual machine" environments in which COTS operating system and application programs may run, but from which no information may be permanently stored or otherwise transmitted except under control of VDE. Such an environment would permit VDE to manage all VDE-protected information, yet may permit unlimited use of COTS applications to process that information within the confines of a restricted environment. The entire contents of such an environment could be treated by VDE 100 as an extension to any VDE-protected documents read into the environment. Transmission of information out of the environment could be governed by the same rules as the original document(s).

"Coarse-Grain" Control Capabilities

As mentioned above, an organization may employ VDE-enforced control capabilities to manage the security, distribution, integrity, and control of entire documents. Some examples of these capabilities may include:

1) A communication channel connecting two or more electronic appliances 600 may be assigned a set of permitted sensitivity attributes. Only documents whose sensitivity attributes belong to this set would be permitted to be transmitted over the channel. This could be used to support the Device Labels requirement of the Trusted Computer System Evaluation Criteria (TCSEC).

2) A writable storage device (e.g., fixed disk, diskette, tape drive. optical disk) connected to or incorporated in an electronic appliance 600 may be assigned a set of permitted sensitivity attributes. Only documents whose sensitivity attributes belong to this set would be permitted to be stored on the device. This could be used to support the TCSEC Device Labels requirement.

3) A document may have a list of users associated with it representing the users who are permitted to "handle" the document. This list of users may represent, for example, the only users who may view the document, even if other users receive the document container, they could not manipulate the

contents. This could be used to support the standard ORCON handling caveat.

4) A document may have an attribute designating its originator and requiring an explicit permission to be granted by an originator before the document's content could be viewed. This request for permission may be made at the time the document is accessed by a user, or, for example, at the time one user distributes the document to another user. If permission is not granted, the document could not be manipulated or otherwise used.

5) A document may have an attribute requiring that each use of the document be reported to the document's originator. This may be used by an originator to gauge the distribution of the document. Optionally, the report may be required to have been made successfully before any use of the document is permitted, to ensure that the use is known to the controlling party at the time of use. Alternatively, for example, the report could be made in a deferred ("batch") fashion.

6) A document may have an attribute requiring that each use of the document be reported to a central document tracking clearinghouse. This could be used by the organization to track specific documents, to identify documents used by any particular user and/or group of users to track documents with specific attributes (e.g., sensitivity), etc. Optionally, for example, the report may be required to have been made successfully before any use of the document is permitted.

7) A VDE protected document may have an attribute requiring that each use of the document generate a "return receipt," to an originator. A person using the document may be required to answer specific questions in order to generate a return receipt, for example by indicating why the document is of interest, or by indicating some knowledge of the document's contents (after reading it). This may be used as assurance that the document had been handled by a person, not by any automated software mechanism.

8) A VDE protected document's content may be made available to a VDE-unaware application program in such a way that it is uniquely identifiable (traceable) to a user who caused its release. Thus, if the released form of the document is further distributed, its origin could be determined. This may be done by employing VDE "fingerprinting" for content release. Similarly, a printed VDE protected document may be marked in a similar, VDE fingerprinted unique way such that the person who originally printed the document could be determined, even if copies have since been made.

9) Usage of VDE protected documents could be permitted under control of budgets that limit (based on size, time of access, etc.) access or other usage of document content. This may help prevent wholesale disclosure by limiting the number of VDE documents accessible to an individual during a fixed time period. For example, one such control might permit a user, for some particular class of documents, to view at most 100 pages/day, but only print 10 pages/day and permit printing only on weekdays between nine and five. As a further

example, a user might be restricted to only a certain quantity of logically related, relatively "contiguous" and/or some other pattern (such as limiting the use of a database's records based upon the quantity of records that share a certain identifier in field) of VDE protected document usage to identify, for example, the occurrence of one or more types of excessive database usage (under normal or any reasonable circumstances). As a result, VDE content providers can restrict usage of VDE content to acceptable usage characteristics and thwart and/or identify (for example, by generating an exception report for a VDE administrator or organization supervisor) user attempts to inappropriately use, for example, such an information database resource.

These control capabilities show some examples of how VDE can be used to provide a flexible, interactive environment for tracking and managing sensitive documents. Such an environment could directly trace the flow of a document from person to person, by physical locations, by organizations, etc. It would also permit specific questions to be answered such as "what persons outside the R&D department have received any

R&D-controlled document." Because the control information is carried with each copy of a VDE protected document, and can ensure that central registries are updated and/or that originators are notified of document use, tracking can be prompt and accurate.

This contrasts with traditional means of tracking paper documents: typically, a paper-oriented system of manually collected and handled receipts is used. Documents may be individually copy-numbered and signed for, but once distributed are not actively controlled. In a traditional paper-oriented system, it is virtually impossible to determine the real locations of documents; what control can be asserted is possible only if all parties strictly follow the handling rules (which are at best inconvenient).

The situation is no better for processing documents within the context of ordinary computer and network systems. Although said systems can enforce access control information based on user identity, and can provide auditing mechanisms for tracking accesses to files, these are low-level mechanisms that do not permit tracking or controlling the flow of content. In such systems, because document content can be freely copied and manipulated, it is not possible to determine where document

content has gone, or where it came from. In addition, because the control mechanisms in ordinary computer operating systems operate at a low level of abstraction, the entities they control are not necessarily the same as those that are manipulated by users. This particularly causes audit trails to be cluttered with voluminous information describing uninteresting activities.

Fine-Grain" Control Capabilities

In addition to controlling and managing entire documents, users may employ customized VDE-aware application software to control and manage individual modifications to documents. Examples of these capabilities include the following:

1) A VDE content user may be permitted to append further information to a VDE document to indicate a proposed alternative wording. This proposed alteration would be visible to all other users (in addition to the original text) of the document but would (for example) be able to be incorporated into the actual text only by the document's owner.

2) A group of VDE users could be permitted to modify one or more parts of a document in such a way that each individual alteration would be unambiguously

traceable to the specific user who performed it. The rights to modify certain portions of a document, and the extension of differing sets of rights to different users, allows an organization or secure environment to provide differing permissions enabling different rights to users of the same content.

3) A group of users could create a VDE document incrementally, by building it from individual contributions. These contributions would be bound together within a single controlled document, but each would be individually identified, for example, through their incorporation in VDE content containers as embedded container objects.

4) VDE control and management capabilities could be used to track activities related to individual document areas, for instance recording how many times each section of a document was viewed.

Example - VDE Protected Content Repository

As the "Digital Highway" emerges, there is increased discussion concerning the distribution of content across networks and, in particular, public networks such as the Internet. Content

may be made available across public networks in several ways including:

- "mailing" content to a user in response to a request or advance purchase (sending a token representing the commitment of electronic funds or credit to purchase an item);

- supporting content downloadable from an organization's own content repository, such a repository comprising, for example, a store of products (such as software programs) and/or a store of information resources, normally organized into one or more databases; and

- supporting a public repository into which other parties can deposit their products for redistribution to customers (normally by making electronic copies for distribution to a customer in response to a request).

One possible arrangement of VDE nodes involves use of one or more "repositories." A repository, for example, may serve as a location from which VDE participants may retrieve VDE content containers. In this case, VDE users may make use of a

network to gain access to a "server" system that allows one or more VDE users to access an object repository containing VDE content containers.

Some VDE participants may create or provide content and/or VDE content container objects, and then store content and/or content objects at a repository so that other participants may access such content from a known and/or efficiently organized (for retrieval) location. For example, a VDE repository (portion of a VDE repository, multiple VDE repositories, and/or providers of content to such repositories) may advertise the availability of certain types of VDE protected content by sending out email to a list of network users. If the network users have secure VDE subsystems in their electronic appliances, they may then choose to access such a repository directly, or through one or more smart agents and, using an application program for example, browse (and/or electronically search) through the offerings of VDE managed content available at the repository, download desirable VDE content containers, and make use of such containers. If the repository is successful in attracting users who have an interest in such content, VDE content providers may determine that such a repository is a desirable location(s) to make their content available for easy access by users. If a repository, such as CompuServe, stores content in

non-encrypted (plaintext) form, it may encrypt "outgoing" content on an "as needed" basis through placing such content in VDE content containers with desired control information, and may employ VDE secure communications techniques for content communication to VDE participants.

VDE repositories may also offer other VDE services. For example, a repository may choose to offer financial services in the form of credit from the repository that may be used to pay fees associated with use of VDE objects obtained from the repository. Alternatively or in addition, a VDE repository may perform audit information clearinghouse services on behalf of VDE creators or other participants (e.g. distributors, redistributors, client administrators, etc.) for usage information reported by VDE users. Such services may include analyzing such usage information, creating reports, collecting payments, etc.

A "full service" VDE repository may be very attractive to both providers and users of VDE managed content. Providers of VDE managed content may desire to place their content in a location that is well known to users, offers credit, and/or performs audit services for them. In this case, providers may be able to focus on creating content, rather than managing the

administrative processes associated with making content available in a "retail" fashion, collecting audit information from many VDE users, sending and receiving bills and payments, etc. VDE users may find the convenience of a single location (or an integrated arrangement of repositories) appealing as they are attempting to locate content of interest. In addition, a full service VDE repository may serve as a single location for the reporting of usage information generated as a consequence of their use of VDE managed content received from a VDE repository and/or, for example, receiving updated software (e.g. VDE-aware applications, load modules, component assemblies, non VDE-aware applications, etc.) VDE repository services may be employed in conjunction with VDE content delivery by broadcast and/or on physical media, such as CD-ROM, to constitute an integrated array of content resources that may be browsed, searched, and/or filtered, as appropriate, to fulfill the content needs of VDE users.

A public repository system may be established and maintained as a non-profit or for-profit service. An organization offering the service may charge a service fee, for example, on a per transaction basis and/or as a percentage of the payments by, and/or cost of, the content to users. A repository service may supply VDE authoring tools to content creators, publishers,

distributors, and/or value adding providers such that they may apply rules and controls that define some or all of the guidelines managing use of their content and so that they may place such content into VDE content container objects.

A repository may be maintained at one location or may be distributed across a variety of electronic appliances, such as a variety of servers (e.g. video servers, etc.) which may be at different locations but nonetheless constitute a single resource. A VDE repository arrangement may employ VDE secure communications and VDE node secure subsystems ("protected processing environments"). The content comprising a given collection or unit of information desired by a user may be spread across a variety of physical locations. For example, content representing a company's closing stock price and the activity (bids, lows, highs, etc.) for the stock might be located at a World Wide Web server in New York, and content representing an analysis of the company (such as a discussions of the company's history, personnel, products, markets, and/or competitors) might be located on a server in Dallas. The content might be stored using VDE mechanisms to secure and audit use. The content might be maintained in clear form if sufficient other forms of security are available at such one or more of sites (e.g. physical security, password, protected operating system, data encryption,

or other techniques adequate for a certain content type). In the latter instances, content may be at least in part encrypted and placed in VDE containers as it streams out of a repository so as to enable secure communication and subsequent VDE usage control and usage consequence management.

A user might request information related to such a company including stock and other information. This request might, for example, be routed first through a directory or a more sophisticated database arrangement located in Boston. This arrangement might contain pointers to, and retrieve content from, both the New York and Dallas repositories. This information content may, for example, be routed directly to the user in two containers (e.g. such as a VDE content container object from Dallas and a VDE content container object from New York). These two containers may form two VDE objects within a single VDE container (which may contain two content objects containing the respective pieces of content from Dallas and New York) when processed by the user's electronic appliance. Alternatively, such objects might be integrated together to form a single VDE container in Boston so that the information can be delivered to the user within a single container to simplify registration and control at the user's site. The information content from both locations may be stored as separate

information objects or they may be joined into a single, integrated information object (certain fields and/or categories in an information form or template may be filled in by one resource and other fields and/or categories may be filled by information provided by a different resource). A distributed database may manage such a distributed repository resource environment and use VDE to secure the storing, communicating, auditing, and/or use of information through VDE's electronic enforcement of VDE controls. VDE may then be used to provide both consistent content containers and content control services.

An example of one possible repository arrangement 3300 is shown in Figure 78. In this example, a repository 3302 is connected to a network 3304 that allows authors 3306A, 3306B, 3306C, and 3306D; a publisher 3308; and one or more end users 3310 to communicate with the repository 3302 and with each other. A second network 3312 allows the publisher 3308, authors 3306E and 3306F, an editor 3314, and a librarian 3316 to communicate with each other and with a local repository 3318. The publisher 3308 is also directly connected to author 3306E. In this example, the authors 3306 and publisher 3308 connect to the repository 3302 in order to place their content into an environment in which end users 3310 will be able to gain access to a broad selection of content from a common location.

In this example, the repository has two major functional areas: a content system 3302A and a clearinghouse system 3302B. The content system 3302A is comprised of a user/author registration system 3320, a content catalog 3322, a search mechanism 3324, content storage 3326, content references 3328, and a shipping system 3330 comprised of a controls packager 3322, a container packager 3334, and a transaction system 3336. The clearinghouse system 3302B is comprised of a user/author registration system 3338; template libraries 3340; a control structure library 3342; a disbursement system 3344; an authorization system 3346 comprised of a financial system 3348 and a content system 3350; a billing system 3352 comprised of a paper system 3354, a credit card system 3356, and an electronic funds transfer (EFT) system 3358; and an audit system 3360 comprised of a receipt system 3362, a response system 3364, a transaction system 3366, and an analysis system 3368.

In this example, author 3306A creates content in electronic form that she intends to make broadly available to many end users 3310, and to protect her rights through use of VDE. Author 3306A transmits a message to the repository 3302 indicating her desire to register with the repository to distribute her content. In response to this message, the user/author registration system 3320 of the content system 3302A, and the

user/author registration system 3338 of the clearinghouse system 3302B transmit requests for registration information to author 3306A using the network 3304. These requests may be made in an on-line interactive mode; or they may be transmitted in a batch to author 3306A who then completes the requested information and transmits it as a batch to the repository 3302; or some aspects may be handled on-line (such as basic identifying information) and other information may be exchanged in a batch mode.

Registration information related to the content system 3302A may, for example, include:

- a request that Author 3306A provide information concerning the types and/or categories of content proposed for storage and access using the repository,

- the form of abstract and/or other identifying information required by the repository—in addition to providing author 3306A with an opportunity to indicate whether or not author 3306A generally includes other information with content submissions (such as promotional materials, detailed information regarding the format of submitted content, any equipment requirements that should or must be met

for potential users of submitted content to successfully exploit its value, etc.),

- requests for information from author 3306A concerning where the content is to be located (stored at the repository, stored at author 3306A's location, stored elsewhere, or some combination of locations),

- what general search characteristics should be associated with content submissions (e.g. whether abstracts should be automatically indexed for searches by users of the repository, the manner in which content titles, abstracts, promotional materials, relevant dates, names of performers and/or authors, or other information related to content submissions may or should be used in lists of types of content and/or in response to searches, etc.), and/or

- how content that is stored at and/or passed through the repository should be shipped (including any container criteria, encryption requirements, transaction requirements related to content transmissions, other control criteria, etc.)

The information requested from author 3306A by the user/author registration system of the clearinghouse may, for example, consist of:

- VDE templates that author 3306A may or must make use of in order to correctly format control information such that, for example, the audit system 3360 of the clearinghouse system 3302B is properly authorized to receive and/or process usage information related to content submitted by author 3306A,

- VDE control information available from the clearinghouse 3302B that may or must be used by author 3306A (and/or included by reference) in some or all of the VDE component assemblies created and/or used by author 3306A associated with submitted content,

- the manner in which disbursement of any funds associated with usage of content provided by, passed through, or collected by the repository clearinghouse system 3302B should be made,

- the form and/or criteria of authorizations to use submitted content and/or financial transactions associated with content,

- the acceptable forms of billing for use of content and/or information associated with content (such as analysis reports that may be used by others),

- how VDE generated audit information should be received,

- how responses to requests from users should be managed,

- how transactions associated with the receipt of audit information should be formatted and authorized,

- how and what forms of analysis should be performed on usage information, and/or

- under what circumstances (if any) usage information and/or analysis results derived from VDE controlled content usage information should be managed (including tc whom they may or must be delivered,

the form of delivery, any control information that may be associated with use of such information, etc.)

The repository 3302 receives the completed registration information from author 3306A and uses this information to build an account profile for author 3306A. In addition, software associated with the authoring process may be transmitted to author 3306A. This software may, for example, allow author 3306A to place content into a VDE content container with appropriate controls in such a way that many of the decisions associated with creating such containers are made automatically to reflect the use of the repository 3302 as a content system and/or a clearinghouse system (for example, the location of content, the party to contact for updates to content and/or controls associated with content, the party or parties to whom audit information may and/or must be transmitted and the pathways for such communication, the character of audit information that is collected during usage, the forms of payment that are acceptable for use of content, the frequency of audit transmissions required, the frequency of billing, the form of abstract and/or other identifying information associated with content, the nature of at least a portion of content usage control information, etc.)

Author 3306A makes use of a VDE authoring application to specify the controls and the content that she desires to place within a VDE content container, and produces such a container in accordance with any requirements of the repository 3302. Such a VDE authoring application may be, for example, an application provided by the repository 3302 which can help ensure adherence to repository content control requirements such as the inclusion of one or more types of component assemblies or other VDE control structures and/or required parameter data, an application received from another party, and/or an application created by author 3306A in whole or in part. Author 3306A then uses the network 3304 to transmit the container and any deviations from author 3306A's account profile that may relate to such content to the repository 3302. The repository 3302 receives the submitted content, and then -- in accordance with any account profile requirements, deviations and/or desired options in this example—makes a determination as to whether the content was produced within the boundaries of any content and/or control information requirements of the repository and therefore should be placed within content storage or referenced by a location pointer or the like. In addition to placing the submitted content into content storage or referencing such content's location, the repository 3302 may also make note of characteristics associated with such submitted content in the

search mechanism 3324, content references 3328, the shipping system 3330, and/or the relevant systems of the clearinghouse system 3302B related to templates and control structures, authorizations, billing and/or payments, disbursements, and/or audits of usage information.

During an authoring process, author 3306A may make use of VDE templates. Such templates may be used as an aspect of a VDE authoring application. For example, such templates may be used in the construction of a container as described above. Alternatively or in addition, such templates may also be used when submitted content is received by the repository 3302. References to such templates may be incorporated by author 3306A as an aspect of constructing a container for submitted content (in this sense the container delivered to the repository may be in some respects "incomplete" until the repository "completes" the container through use of indicated templates). Such references may be required for use by the repository 3302 (for example, to place VDE control information in place to fulfill an aspect of the repository's business or security models such as one or more map tables corresponding to elements of content necessary for interacting with other VDE control structures to accommodate certain metering, billing, budgeting, and/or other usage and/or distribution related controls of the repository).

For example, if content submitted by author 3306A consists of a periodical publication, a template delivered to the author by the repository 3302 when the author registers at the repository may be used as an aspect of an authoring application manipulated by the author in creating a VDE content container for such a periodical. Alternatively or in addition, a template designed for use with periodical publications may be resident at the repository 3302, and such a template may be used by the repository to define, in whole or in part, control structures associated with such a container. For example, a VDE template designed to assist in formulating control structures for periodical publications might indicate (among other things) that:

- usage controls should include a meter method that records each article within a publication that a user opens,

- a certain flat rate fee should apply to opening the periodical regardless of the number of articles opened, and/or

- a record should be maintained of every advertisement that is viewed by a user.

If content is maintained in a known and/or identifiable format, such a template may be used during initial construction of a container without author 3306A's intervention to identify any map tables that may be required to support such recording and billing actions. If such a VDE template is unavailable to author 3306A, she may choose to indicate that the container submitted should be reconstructed (e.g. augmented) by the repository to include the VDE control information specified in a certain template or class of templates. If the format of the content is known and/or identifiable by the repository, the repository may be able to reconstruct (or "complete") such a container automatically.

One factor in a potentially ongoing financial relationship between the repository and author 3306A may relate to usage of submitted content by end users 3310. For example, author 3306A may negotiate an arrangement with the repository wherein the repository is authorized to keep 20% of the total revenues generated from end users 3310 in exchange for maintaining the repository services (e.g. making content available to end users 3310, providing electronic credit, performing billing activities, collecting fees, etc.) A financial relationship may be recorded in control structures in flexible and configurable ways. For example, the financial relationship

described above could be created in a VDE container and/or installation control structure devised by author 3306A to reflect author 3306A's financial requirements and the need for a 20% split in revenue with the repository wherein all billing activities related to usage of submitted content could be processed by the repository, and control structures representing reciprocal methods associated with various component assemblies required for use of author 3306A's submitted content could be used to calculate the 20% of revenues. Alternatively, the repository may independently and securely add and/or modify control structures originating from author 3306A in order to reflect an increase in price. Under some circumstances, author 3306A may not be directly involved (or have any knowledge of) the actual price that the repository charges for usage activities, and may concern herself only with the amount of revenue and character of usage analysis information that she requires for her own purposes, which she specifies in VDE control information which governs the use, and consequences of use, of VDE controlled content.

Another aspect of the relationship between authors and the repository may involve the character of transaction recording requirements associated with delivery of VDE controlled content and receipt of VDE controlled content usage audit information. For example, author 3306A may require that the repository

make a record of each user that receives a copy of content from the repository. Author 3306A may further require collection of information regarding the circumstances of delivery of content to such users (e.g. time, date, etc.) In addition, the repository may elect to perform such transactions for use internally (e.g. to determine patterns of usage to optimize systems, detect fraud, etc.)

In addition to recording information regarding delivery of such VDE controlled content, author 3306A may have required or requested the repository to perform certain VDE container related processes. For example, author 3306A may want differing abstract and/or other descriptive information delivered to different classes of users. In addition, author 3306A may wish to deliver promotional materials in the same container as submitted content depending on, for example, the character of usage exhibited by a particular user (e.g. whether the user has ever received content from author 3306A, whether the user is a regular subscriber to author 3306A's materials, and/or other patterns that may be relevant to author 3306A and/or the end user that are used to help determine the mix of promotional materials delivered to a certain VDE content end user.) In another example, author 3306A may require that VDE

fingerprinting be performed on such content prior to transmission of content to an end user.

In addition to the form and/or character of content shipped to an end user, authors may also require certain encryption related processes to be performed by the repository as an aspect of delivering content. For example, author 3306A may have required that the repository encrypt each copy of shipped content using a different encryption key or keys in order to help maintain greater protection for content (e.g. in case an encryption key was "cracked" or inadvertently disclosed, the "damage" could be limited to the portion(s) of that specific copy of a certain content deliverable). In another example, encryption functions may include the need to use entirely different encryption algorithms and/or techniques in order to fulfill circumstantial requirements (e.g. to comply with export restrictions). In a further example, encryption related processes may include changing the encryption techniques and/or algorithms based on the level of trustedness and/or tamper resistance of the VDE site to which content is delivered.

In addition to transaction information gathered when content is shipped from a VDE repository to an end user, the repository may be required to keep transaction information

related to the receipt of usage information, requests, and/or responses to and/or from end users 3310. For example, author 3306A may require the repository to keep a log of some or all connections made by end users 3310 related to transmissions and or reception of information related to the use of author 3306A's content (e.g. end user reporting of audit information, end user requests for additional permissions information, etc.)

Some VDE managed content provided to end users 3310 through the repository may be stored in content storage. Other information may be stored elsewhere, and be referenced through the content references. In the case where content references are used, the repository may manage the user interactions in such a manner that all repository content, whether stored in content storage or elsewhere (such as at another site), is presented for selection by end users 3310 in a uniform way, such as, for example, a consistent or the same user interface. If an end user requests delivery of content that is not stored in content storage, the VDE repository may locate the actual storage site for the content using information stored in content references (e.g. the network address where the content may be located, a URL, a filesystem reference, etc.) After the content is located, the content may be transmitted across the network to the repository or it may be delivered directly from where it is stored to the

requesting end user. In some circumstances (e.g. when container modification is required, when encryption must be changed, if financial transactions are required prior to release, etc.), further processing may be required by the repository in order to prepare such VDE managed content and/or VDE content container for transmission to an end user.

In order to provide a manageable user interface to the content available to VDE repository end users 3310 and to provide administrative information used in the determination of control information packaged in VDE content containers shipped to end users 3310, the repository in this example includes a content catalog 3322. This catalog is used to record information related to the VDE content in content storage, and/or content available through the repository reflected in content references. The content catalog 3322 may consist of titles of content, abstracts, and other identifying information. In addition, the catalog may also indicate the forms of electronic agreement and/or agreement VDE template applications (offering optional, selectable control structures and/or one or more opportunities to provide related parameter data) that are available to end users 3310 through the repository for given pieces of content in deciding, for example, options and/or requirements for: what type(s) of information is recorded during such content's use, the

charge for certain content usage activities, differences in charges based on whether or not certain usage information is recorded and/or made available to the repository and/or content provider, the redistribution rights associated with such content, the reporting frequency for audit transmissions, the forms of credit and/or currency that may be used to pay certain fees associated with use of such content, discounts related to certain volumes of usage, discounts available due to the presence of rights associated with other content from the same and/or different content providers, sales, etc. Furthermore, a VDE repository content catalog 3322 may indicate some or all of the component assemblies that are required in order to make use of content such that the end user's system and the repository can exchange messages to help ensure that any necessary VDE component assemblies or other VDE control information is identified, and if necessary and authorized, are delivered along with such content to the end user (rather than, for example, being requested later after their absence has been detected during a registration and/or use attempt).

In order to make use of the VDE repository in this example, an end user must register with the repository. In a manner similar to that indicated above in the case of an author, a VDE end user transmits a message from her VDE installation

to the repository across the network indicating that she wishes to make use of the services provided by the repository (e.g. access content stored at and/or referenced by the repository, use credit provided by the repository, etc.) In response to this message, the user/author registration systems of the content system 3302A and the clearinghouse system 3302B of the repository transmit requests for information from the end user (e.g. in an on-line and/or batch interaction). The information requested by the user/author registration system of the content system 3302A may include type(s) of content that the user wishes to access, the characteristics of the user's electronic appliance 600, etc. The information requested by the user/author registration system of the clearinghouse system 3302B may include whether the user wishes to establish a credit account with the clearinghouse system 3302B, what other forms of credit the user may wish to use for billing purposes, what other clearinghouses may be used by the end user in the course of interacting with content obtained from the repository, any general rules that the user has established regarding their preferences for release and handling of usage analysis information, etc. Once the end user has completed the registration information and transmitted it to the repository, the repository may construct an account profile for the user. In this example, such requests and responses are

handled by secure VDE communications between secure VDE subsystems of both sending and receiving parties.

In order to make use of the repository, the end user may operate application software. In this example, the end user may either make use of a standard application program (e.g. a World Wide Web browser such as Mosaic), or they may make use of application software provided by the repository after completion of the registration process. If the end user chooses to make use of the application software provided by the repository, they may be able to avoid certain complexities of interaction that may occur if a standard package is used. Although standardized packages are often relatively easy to use, a customized package that incorporates VDE aware functionality may provide an easier to use interface for a user. In addition, certain characteristics of the repository may be built in to the interface to simplify use of the services (e.g. similar to the application programs provided by America Online).

The end user may connect to the repository using the network. In this example, after the user connects to the repository, an authentication process will occur. This process can either be directed by the user (e.g. through use of a login and password protocol) or may be established by the end user's

electronic appliance secure subsystems interacting with a repository electronic appliance in a VDE authentication. In either event, the repository and the user must initially ensure that they are connected to the correct other party. In this example, if secured information will flow between the parties, a VDE secured authentication must occur, and a secure session must be established. On the other hand, if the information to be exchanged has already been secured and/or is available without authentication (e.g. certain catalog information, containers that have already been encrypted and do not require special handling, etc.), the "weaker" form of login/password may be used.

Once an end user has connected to the VDE repository and authentication has occurred, the user may begin manipulating and directing their user interface software to browse through a repository content catalog 3322 (e.g. lists of publications, software, games, movies, etc.), use the search mechanism to help locate content of interest, schedule content for delivery, make inquiries of account status, availability of usage analysis information, billing information, registration and account profile information, etc. If a user is connecting to obtain content, the usage requirements for that content may be delivered to them. If the user is connecting to deliver usage information to the repository, information related to that transmission may be

delivered to them. Some of these processes are described in more detail below.

In this example, when an end user requests content from the VDE repository (e.g. by selecting from a menu of available options), the content system 3302A locates the content either in the content references and/or in content storage. The content system 3302A may then refer to information stored in the content catalog 3322, the end user's account profile, and/or the author's account profile to determine the precise nature of container format and/or control information that may be required to create a VDE content container to fulfill the end user's request. The shipping system then accesses the clearinghouse system 3302Bto gather any necessary additional control structures to include with the container, to determine any characteristics of the author's and/or end user's account profiles that may influence either the transaction(s) associated with delivering the content to the end user or with whether the transaction may be processed. If the transaction is authorized, and all elements necessary for the container are available, the controls packager forms a package of control information appropriate for this request by this end user, and the container packager takes this package of control information and the content and forms an appropriate container (including any

permissions that may be codeliverable with the container, incorporating any encryption requirements, etc.) If required by the repository or the author's account profile, transactions related to delivery of content are recorded by the transaction system of the shipping system. When the container and any transactions related to delivery have been completed, the container is transmitted across the network to the end user.

An end user may make use of credit and/or currency securely stored within the end user's VDE installation secure subsystem to pay for charges related to use of VDE content received from the repository, and/or the user may maintain a secure credit and/or currency account remotely at the repository, including a "virtual" repository where payment is made for the receipt of such content by an end user. This later approach may provide greater assurance for payment to the repository and/or content providers particularly if the end user has only an HPE based secure subsystem. If an end user electronic credit and/or currency account is maintained at the repository in this example, charges are made to said account based on end user receipt of content from the repository. Further charges to such a remote end user account may be made based on end user usage of such received content and based upon content usage information communicated to the repository clearinghouse system 3302B.

In this example, if an end user does not have a relationship established with a financial provider (who has authorized the content providers whose content may be obtained through use of the repository to make use of their currency and/or credit to pay for any usage fees associated with such provider's content) and/or if an end user desires a new source of such credit, the end user may request credit from the repository clearinghouse system 3302B. If an end user is approved for credit, the repository may extend credit in the form of credit amounts (e.g. recorded in one or more UDEs) associated with a budget method managed by the repository. Periodically, usage information associated with such a budget method is transmitted by the end user to the audit system of the repository. After such a transmission (but potentially before the connection is terminated), an amount owing is recorded for processing by the billing system, and in accordance with the repository's business practices, the amount of credit available for use by the end user may be replenished in the same or subsequent transmission. In this example, the clearinghouse of the repository supports a billing system with a paper system for resolving amounts owed through the mail, a credit card system for resolving amounts owed through charges to one or more credit cards, and an electronic funds transfer system for resolving such amounts through direct debits to a bank account. The repository may

automatically make payments determined by the disbursement system for monies owed to authors through use of similar means. Additional detail regarding the audit process is provided below.

As indicated above, end users 3310 in this example will periodically contact the VDE repository to transmit content usage information (e.g. related to consumption of budget, recording of other usage activities, etc.), replenish their budgets, modify their account profile, access usage analysis information, and perform other administrative and information exchange activities. In some cases, an end user may wish to contact the repository to obtain additional control structures. For example, if an end user has requested and obtained a VDE content container from the repository, that container is typically shipped to the end user along with control structures appropriate to the content, the author's requirements and account profile, the end user's account profile, the content catalog 3322, and/or the circumstances of the delivery (e.g. the first delivery from a particular author, a subscription, a marketing promotion, presence and/or absence of certain advertising materials, requests formulated on behalf of the user by the user's local VDE instance, etc.) Even though, in this example, the repository may have attempted to deliver all relevant control structures, some containers may include controls structures that allow for options

that the end user did not anticipate exercising (and the other criteria did not automatically select for inclusion in the container) that the end user nonetheless determines that they would like to exercise. In this case, the end user may wish to contact the repository and request any additional control information (including, for example, control structures) that they will need in order to make use of the content under such option.

For example, if an end user has obtained a VDE content container with an overall control structure that includes an option that records of the number of times that certain types of accesses are made to the container and further bases usage fees on the number of such accesses, and another option within the overall control structure allows the end user to base the fees paid for access to a particular container based on the length of time spent using the content of the container, and the end user did not originally receive controls that would support this latter form of usage, the repository may deliver such controls at a later time and when requested by the user. In another example, an author may have made changes to their control structures (e.g. to reflect a sale, a new discounting model, a modified business strategy, etc.) which a user may or must receive in order to use the content container with the changed control structures. For example, one or more control structures associated with a certain VDE content

container may require a "refresh" for continued authorization to employ such structures, or the control structures may expire. This allows (if desired) a VDE content provider to periodically modify and/or add to VDE control information at an end user's site (employing the local VDE secure subsystem).

Audit information (related to usage of content received from the repository) in this example is securely received from end users 3310 by the receipt system 3362 of the clearinghouse. As indicated above, this system may process the audit information and pass some or all of the output of such a process to the billing system and/or transmit such output to appropriate content authors. Such passing of audit information employs secure VDE pathway of reporting information handling techniques. Audit information may also be passed to the analysis system in order to produce analysis results related to end user content usage for use by the end user, the repository, third party market researchers, and/or one or more authors. Analysis results may be based on a single audit transmission, a portion of an audit transmission, a collection of audit transmissions from a single end user and/or multiple end users 3310, or some combination of audit transmissions based on the subject of analysis (e.g. usage patterns for a given content element or collection of elements, usage of certain categories of

content, payment histories, demographic usage patterns, etc.) The response system 3364 is used to send information to the end user to, for example, replenish a budget, deliver usage controls, update permissions information, and to transmit certain other information and/or messages requested and/or required by an end user in the course of their interaction with the clearinghouse. During the course of an end user's connections and transmissions to and from the clearinghouse, certain transactions (e.g. time, date, and/or purpose of a connection and/or transmission) may be recorded by the transaction system of the audit system to reflect requirements of the repository and/or authors.

Certain audit information may be transmitted to authors. For example, author 3306A may require that certain information gathered from an end user be transmitted to author 3306A with no processing by the audit system. In this case, the fact of the transmission may be recorded by the audit system, but author 3306A may have elected to perform their own usage analysis rather than (or in addition to) permitting the repository to access, otherwise process and/or otherwise use this information. The repository in this example may provide author 3306A with some of the usage information related to the repository's budget method received from one or more end users 3310 and generated

by the payment of fees associated with such users' usage of content provided by author 3306A . In this case, author 3306A may be able to compare certain usage information related to content with the usage information related to the repository's budget method for the content to analyze patterns of usage (e.g. to analyze usage in light of fees, detect possible fraud, generate user profile information, etc.) Any usage fees collected by the clearinghouse associated with author 3306A's content that are due to author 3306A will be determined by the disbursement system of the clearinghouse. The disbursement system may include usage information (in complete or summary form) with any payments to author 3306A resulting from such a determination. Such payments and information reporting may be an entirely automated sequence of processes occurring within the VDE pathway from end user VDE secure subsystems, to the clearinghouse secure subsystem, to the author's secure subsystem.

In this example, end users 3310 may transmit VDE permissions and/or other control information to the repository 3302 permitting and/or denying access to usage information collected by the audit system for use by the analysis system. This, in part, may help ensure end user's privacy rights as it relates to the usage of such information. Some containers may

require, as an aspect of their control structures, that an end user make usage information available for analysis purposes. Other containers may give an end user the option of either allowing the usage information to be used for analysis, or denying some or all such uses of such information. Some users may elect to allow analysis of certain information, and deny this permission for other information. End users 3310 in this example may, for example, elect to limit the granularity of information that may be used for analysis purposes (e.g. an end user may allow analysis of the number of movies viewed in a time period but disallow use of specific titles, an end user may allow release of their ZIP code for demographic analysis, but disallow use of their name and address, etc.) Authors and/or the repository 3302 may, for example, choose to charge end users 3310 smaller fees if they agree to release certain usage information for analysis purposes.

In this example, the repository 3302 may receive content produced by more than one author. For example, author B, author C, and author D may each create portions of content that will be delivered to end users 3310 in a single container. For example, author B may produce a reference work. Author C may produce a commentary on author B's reference work, and author D may produce a set of illustrations for author B's reference work and author C's commentary. Author B may collect together

author C's and author D's content and add further content (e.g. the reference work described above) and include such content in a single container which is then transmitted to the repository 3302. Alternatively, each of the authors may transmit their works to the repository 3302 independently, with an indication that a template should be used to combine their respective works prior to shipping a container to an end user. Still alternatively, a container reflecting the overall content structure may be transmitted to the repository 3302 and some or all of the content may be referenced in the content references rather than delivered to the repository 3302 for storage in content storage.

When an end user makes use of container content, their content usage information may, for example, be segregated in accordance with control structures that organize usage information based at least in part on the author who created that segment. Alternatively, the authors and/or the VDE repository 3302 may negotiate one or more other techniques for securely dividing and/or sharing usage information in accordance with VDE control information. Furthermore, control structures associated with a container may implement models that differentiate any usage fees associated with portions of content based on usage of particular portions, overall usage of the container, particular patterns of usage, or other mechanism

negotiated (or otherwise agreed to) by the authors. Reports of usage information, analysis results, disbursements, and other clearinghouse processes may also be generated in a manner that reflects agreements reached by repository 3302 participants (authors, end users 3310 and/or the repository 3302) with respect to such processes. These agreements may be the result of a VDE control information negotiation amongst these participants.

In this example, one type of author is a publisher 3308. The publisher 3308 in this example communicates over an "internal" network with a VDE based local repository 3302 and over the network described above with the public repository 3302. The publisher 3308 may create or otherwise provide content and/or VDE control structure templates that are delivered to the local repository 3302 for use by other participants who have access to the "internal" network. These templates may be used to describe the structure of containers, and may further describe whom in the publisher 3308's organization may take which actions with respect to the content created within the organization related to publication for delivery to (and/or referencing by) the repository 3302. For example, the publisher 3308 may decide (and control by use of said temple) that a periodical publication will have a certain format with respect to the structure of its content and the types

of information that may be included (e.g. text, graphics, multimedia presentations, advertisements, etc.), the relative location and/or order of presentation of its content, the length of certain segments, etc. Furthermore, the publisher 3308 may, for example, determine (through distribution of appropriate permissions) that the publication editor is the only party that may grant permissions to write into the container, and that the organization librarian is the only party that may index and/or abstract the content. In addition, the publisher 3308 may, for example, allow only certain one or more parties to finalize a container for delivery to the repository 3302 in usable form (e.g. by maintaining control over the type of permissions, including distribution permissions, that may be required by the repository 3302 to perform subsequent distribution activities related to repository end users 3310).

In this example, author 3306E is connected directly to the publisher 3308, such that the publisher 3308 can provide templates for that author that establish the character of containers for author 3306E's content. For example, if author 3306E creates books for distribution by the publisher 3308, the publisher 3308 may define the VDE control structure template which provides control method options for author 3306E to select from and which provides VDE control structures for securely

distributing author 3306E's works. Author 3306E and the publisher 3308 may employ VDE negotiations for the template characteristics, specific control structures, and/or parameter data used by author 3306E. Author 3306E may then use the template(s) to create control structures for their content containers. The publisher 3308 may then deliver these works to the repository 3302 under a VDE extended agreement comprising electronic agreements between author 3306E and the publisher 3308 and the repository 3302 and the publisher 3308.

In this example, the publisher 3308 may also make author 3306E's work available on the local repository 3302. The editor may authorize (e.g. through distribution of appropriate permissions) author F to create certain portions of content for a publication. In this example, the editor may review and/or modify author F's work and further include it in a container with content provided by author 3306E (available on the local repository 3302). The editor may or may not have permissions from the publisher 3308 to modify author 3306E's content (depending on any negotiation(s) that may have occurred between the publisher 3308 and author 3306E, and the publisher 3308's decision to extend such rights to the editor if permissions to modify author 3306E's content are held in redistributable form by the publisher 3308). The editor may also include content from

other authors by (a) using a process of granting permissions to authors to write directly into the containers and/or (b) retrieving containers from the local repository 3302 for inclusion. The local repository 3302 may also be used for other material used by the publisher 3308's organization (e.g. databases, other reference works, internal documents, draft works for review, training videos, etc.), such material may, given appropriate permissions, be employed in VDE container collections of content created by the editor.

The librarian in this example has responsibility for building and/or editing inverted indexes, keyword lists (e.g. from a restricted vocabulary), abstracts of content, revision histories, etc. The publisher 3308 may, for example, grant permissions to only the librarian for creating this type of content. The publisher 3308 may further require that this building and/or editing occur prior to release of content to the repository 3302.

Example -- Evolution and Transformation of VDE Managed Content and Control Information

The VDE content control architecture allows content control information (such as control information for governing content usage) to be shaped to conform to VDE control information requirements of multiple parties. Formulating such

multiple party content control information normally involves securely deriving control information from control information securely contributed by parties who play a role in a content handling and control model (e.g. content creator(s), provider(s), user(s), clearinghouse(s), etc.). Multiple party control information may be necessary in order to combine multiple pieces of independently managed VDE content into a single VDE container object (particularly if such independently managed content pieces have differing, for example conflicting, content control information). Such secure combination of VDE managed pieces of content will frequently require VDE's ability to securely derive content control information which accommodates the control information requirements, including any combinatorial rules, of the respective VDE managed pieces of content and reflects an acceptable agreement between such plural control information sets.

The combination of VDE managed content pieces may result in a VDE managed composite of content. Combining VDE managed content must be carried out in accordance with relevant content control information associated with said content pieces and processed through the use of one or more secure VDE sub-system PPEs 650. VDE's ability to support the embedding, or otherwise combining, of VDE managed content pieces, so as to

create a combination product comprised of various pieces of VDE content, enables VDE content providers to optimize their VDE electronic content products. The combining of VDE managed content pieces may result in a VDE content container which "holds" consolidated content and/or concomitant, separate, nested VDE content containers.

VDE's support for creation of content containers holding distinct pieces of VDE content portions that were previously managed separately allows VDE content providers to develop products whose content control information reflects value propositions consistent with the objectives of the providers of content pieces, and further are consistent with the objectives of a content aggregator who may be producing a certain content combination as a product for commercial distribution. For example, a content product "launched" by a certain content provider into a commercial channel (such as a network repository) may be incorporated by different content providers and/or end-users into VDE content containers (so long as such incorporation is allowed by the launched product's content control information). These different content providers and/or end-users may, for example, submit differing control information for regulating use of such content. They may also combine in different combinations a certain portion of launched content with

content received from other parties (and/or produced by themselves) to produce different content collections, given appropriate authorizations.

VDE thus enables copies of a given piece of VDE managed content to be securely combined into differing consolidations of content, each of which reflects a product strategy of a different VDE content aggregator. VDE's content aggregation capability will result in a wider range of competitive electronic content products which offer differing overall collections of content and may employ differing content control information for content that may be common to such multiple products. Importantly, VDE securely and flexibly supports editing the content in, extracting content from, embedding content into, and otherwise shaping the content composition of, VDE content containers. Such capabilities allow VDE supported product models to evolve by progressively reflecting the requirements of "next" participants in an electronic commercial model. As a result, a given piece of VDE managed content, as it moves through pathways of handling and branching, can participate in many different content container and content control information commercial models.

VDE content, and the electronic agreements associated with said content, can be employed and progressively manipulated in commercial ways which reflect traditional business practices for non-electronic products (though VDE supports greater flexibility and efficiency compared with most of such traditional models). Limited only by the VDE control information employed by content creators, other providers, and other pathway of handling and control participants, VDE allows a "natural" and unhindered flow of, and creation of, electronic content product models. VDE provides for this flow of VDE products and services through a network of creators, providers, and users who successively and securely shape and reshape product composition through content combining, extracting, and editing within a Virtual Distribution Environment.

VDE provides means to securely combine content provided at different times, by differing sources, and or representing differing content types. These types, timings, and/or different sources of content can be employed to form a complex array of content within a VDE content container. For example, a VDE content container may contain a plurality of different content container objects, each containing different content whose usage can be controlled, at least in part, by its own container's set of VDE content control information.

A VDE content container object may, through the use of a secure VDE sub-system, be "safely" embedded within a "parent" VDE content container. This embedding process may involve the creation of an embedded object, or, alternatively, the containing, within a VDE content container, of a previously independent and now embedded object by, at minimum, appropriately referencing said object as to its location.

An embedded content object within a parent VDE content container:

(1) may have been a previously created VDE content container which has been embedded into a parent VDE content container by securely transforming it from an independent to an embedded object through the secure processing of one or more VDE component assemblies within a VDE secure sub-system PPE 650. In this instance, an embedded object may be subject to content control information, including one or more permissions records associated with the parent container, but may not, for example, have its own content control information other than content identification information, or the embedded object may be more extensively controlled by its own content control information (e.g. permissions records).

(2) may include content which was extracted from another VDE content container (along with content control information, as may be applicable) for inclusion into a parent VDE content container in the form of an embedded VDE content container object. In this case, said extraction and embedding may use one or more VDE processes which run securely within a VDE secure sub-system PPE 650 and which may securely remove (or copy) the desired content from a source VDE content container and place such content in a new or existing container object, either of which may be or become embedded into a parent VDE content container.

(3) may include content which was first created and then placed in a VDE content container object. Said receiving container may already be embedded in a parent VDE content container and may already contain other content. The container in which such content is placed may be specified using a VDE aware application which interacts with content and a secure VDE subsystem to securely create such VDE container and place such content therein followed by securely embedding such container into the destination, parent container. Alternatively, content may be specified without the use of a VDE aware

application, and then manipulated using a VDE aware application in order to manage movement of the content into a VDE content container. Such an application may be a VDE aware word processor, desktop and/or multimedia publishing package, graphics and/or presentation package, etc. It may also be an operating system function (e.g. part of a VDE aware operating system or mini-application operating with an O/S such as a Microsoft Windows compatible object packaging application) and movement of content from "outside" VDE to within a VDE object may, for example, be based on a "drag and drop" metaphor that involves "dragging" a file to a VDE container object using a pointing device such as a mouse. Alternatively, a user may "cut" a portion of content and "paste" such a portion into a VDE container by first placing content into a "clipboard," then selecting a target content object and pasting the content into such an object. Such processes may, at the direction of VDE content control information and under the control of a VDE secure subsystem, put the content automatically at some position in the target object, such as at the end of the object or in a portion of the object that corresponds to an identifier carried by or with the content such as a field identifier, or the embedding process might pop-up a user interface that allows a user to browse

a target object's contents and/or table of contents and/or other directories, indexes, etc. Such processes may further allow a user to make certain decisions concerning VDE content control information (budgets limiting use, reporting pathway(s), usage registration requirements, etc.) to be applied to such embedded content and/or may involve selecting the specific location for embedding the content, all such processes to be performed as transparently as practical for the application.

(4) may be accessed in conjunction with one or more operating system utilities for object embedding and linking, such as utilities conforming to the Microsoft OLE standard. In this case, a VDE container may be associated with an OLE "link." Accesses (including reading content from, and writing content to) to a VDE protected container may be passed from an OLE aware application to a VDE aware OLE application that accesses protected content in conjunction with control information associated with such content.

A VDE aware application may also interact with component assemblies within a PPE to allow direct editing of the content of a VDE container, whether the content is in a parent or

embedded VDE content container. This may include the use of a VDE aware word processor, for example, to directly edit (add to, delete, or otherwise modify) a VDE container's content. The secure VDE processes underlying VDE container content editing may be largely or entirely transparent to the editor (user) and may transparently enable the editor to securely browse through (using a VDE aware application) some or all of the contents of, and securely modify one or more of the VDE content containers embedded in, a VDE content container hierarchy.

The embedding processes for all VDE embedded content containers normally involves securely identifying the appropriate content control information for the embedded content. For example, VDE content control information for a VDE installation and/or a VDE content container may securely, and transparently to an embedder (user), apply the same content control information to edited (such as modified or additional) container content as is applied to one or more portions (including all, for example) of previously "in place" content of said container and/or securely apply control information generated through a VDE control information negotiation between control sets, and/or it may apply control information previously applied to said content. Application of control information may occur regardless of whether the edited content is in a parent or embedded container.

This same capability of securely applying content control information (which may be automatically and/or transparently applied), may also be employed with content that is embedded into a VDE container through extracting and embedding content, or through the moving, or copying and embedding, of VDE container objects. Application of content control information normally occurs securely within one or more VDE secure sub-system PPEs 650. This process may employ a VDE template that enables a user, through easy to use GUI user interface tools, to specify VDE content control information for certain or all embedded content, and which may include menu driven, user selectable and/or definable options, such as picking amongst alternative control methods (e.g. between different forms of metering) which may be represented by different icons picturing (symbolizing) different control functions and apply such functions to an increment of VDE secured content, such as an embedded object listed on an object directory display.

Extracting content from a VDE content container, or editing or otherwise creating VDE content with a VDE aware application, provides content which may be placed within a new VDE content container object for embedding into said parent VDE container, or such content may be directly placed into a previously existing content container. All of these processes may

be managed by processing VDE content control information within one or more VDE installation secure sub-systems.

VDE content container objects may be embedded in a parent object through control information referenced by a parent object permissions record that resolves said embedded object's location and/or contents. In this case, little or no change to the embedded object's previously existing content control information may be required. VDE securely managed content which is relocated to a certain VDE content container may be relocated through the use of VDE sub-system secure processes which may, for example, continue to maintain relocated content as encrypted or otherwise protected (e.g. by secure tamper resistant barrier 502) during a relocation/embedding process.

Embedded content (and/or content objects) may have been contributed by different parties and may be integrated into a VDE container through a VDE content and content control information integration process securely managed through the use of one or more secure VDE subsystems. This process may, for example, involve one or more of:

(1.) securely applying instructions controlling the embedding and/or use of said submitted content, wherein said

instructions were securely put in place, at least in part, by a content provider and/or user of said VDE container. For example, said user and/or provider may interact with one or more user interfaces offering a selection of content embedding and/or control options (e.g. in the form of a VDE template). Such options may include which, and/or whether, one or more controls should be applied to one or more portions of said content and/or the entry of content control parameter data (such a time period before which said content may not be used, cost of use of content, and/or pricing discount control parameters such as software program suite sale discounting). Once required and/or optional content control information is established by a provider and/or user, it may function as content control information which may be, in part or in full, applied automatically to certain, or all, content which is embedded in a VDE content container.

(2.) secure VDE managed negotiation activities, including the use of a user interface interaction between a user at a receiving VDE installation and VDE content control information associated with the content being submitted for embedding. For example, such associated control information may propose certain content information and the content receiver may, for example, accept, select from a plurality, reject, offer alternative control information, and/or apply conditions to the use of certain

content control information (for example, accept a certain one or more controls if said content is used by a certain one or more users and/or if the volume of usage of certain content exceeds a certain level).

(3.) a secure, automated, VDE electronic negotiation process involving VDE content control information of the receiving VDE content container and/or VDE installation and content control information associated with the submitted content (such as control information in a permissions record of a contributed VDE object, certain component assemblies, parameter data in one or more UDEs and/or MDEs, etc.).

Content embedded into a VDE content container may be embedded in the form of:

(1.) content that is directly, securely integrated into previously existing content of a VDE content container (said container may be a parent or embedded content container) without the formation of a new container object. Content control information associated with said content after embedding must be consistent with any pre-embedding content control information controlling, at least in part, the establishment of control information required after embedding. Content control

information for such directly integrated, embedded content may be integrated into, and/or otherwise comprise a portion of, control information (e.g. in one or more permissions records containing content control information) for said VDE container, and/or

(2.) content that is integrated into said container in one or more objects which are nested within said VDE content container object. In this instance, control information for said content may be carried by either the content control information for the parent VDE content container, or it may, for example, be in part or in full carried by one or more permissions records contained within and/or specifically associated with one or more content containing nested VDE objects. Such nesting of VDE content containing objects within a parent VDE content container may employ a number of levels, that is a VDE content container nested in a VDE content container may itself contain one or more nested VDE content containers.

VDE content containers may have a nested structure comprising one or more nested containers (objects) that may themselves store further containers and/or one or more types of content, for example, text, images, audio, and/or any other type of electronic information (object content may be specified by content control information referencing, for example, byte offset

locations on storage media). Such content may be stored, communicated, and/or used in stream (such as dynamically accumulating and/or flowing) and/or static (fixed, such as predefined, complete file) form. Such content may be derived by extracting a subset of the content of one or more VDE content containers to directly produce one or more resulting VDE content containers. VDE securely managed content (e.g. through the use of a VDE aware application or operating system having extraction capability) may be identified for extraction from each of one or more locations within one or more VDE content containers and may then be securely embedded into a new or existing VDE content container through processes executing VDE controls in a secure subsystem PPE 650. Such extraction and embedding (VDE "exporting") involves securely protecting, including securely executing, the VDE exporting processes.

A VDE activity related to VDE exporting and embedding involves performing one or more transformations of VDE content from one secure form to one or more other secure forms. Such transformation(s) may be performed with or without moving transformed content to a new VDE content container (e.g. by component assemblies operating within a PPE that do not reveal, in unprotected form, the results or other output of such transforming processes without further VDE processes governing

use of at least a portion of said content). One example of such a transformation process may involve performing mathematical transformations and producing results, such as mathematical results, while retaining, none, some, or all of the content information on which said transformation was performed. Other examples of such transformations include converting a document format (such as from a WordPerfect format to a Word for Windows format, or an SGML document to a Postscript document), changing a video format (such as a QuickTime video format to a MPEG video format), performing an artificial intelligence process (such as analyzing text to produce a summary report), and other processing that derives VDE secured content from other VDE secured content.

Figure 79 shows an example of an arrangement of commercial VDE users. The users in this example create, distribute, redistribute, and use content in a variety of ways. This example shows how certain aspects of control information associated with content may evolve as control information passes through a chain of handling and control. These VDE users and controls are explained in more detail below.

Creator A in this example creates a VDE container and provides associated content control information that includes

references (amongst other things) to several examples of possible "types" of VDE control information. In order to help illustrate this example, some of the VDE control information passed to another VDE participant is grouped into three categories in the following more detailed discussion: distribution control information, redistribution control information, and usage control information. In this example, a fourth category of embedding control information can be considered an element of all three of the preceding categories. Other groupings of control information are possible (VDE does not require organizing control information in this way). The content control information associated with this example of a container created by creator A is indicated on Figure 80 as C_A. Figure 80 further shows the VDE participants who may receive enabling control information related to creator A's VDE content container. Some of the control information in this example is explained in more detail below.

Some of the distribution control information (in this example, control information primarily associated with creation, modification, and/or use of control information by distributors) specified by creator A includes: (a) distributors will compensate creator A for each active user of the content of the container at the rate of $10 per user per month, (b) distributors are budgeted such that they may allow no more than 100 independent users to

gain access to such content (i.e. may create no more than 100 permissions records reflecting content access rights) without replenishing this budget, and (c) no distribution rights may be passed on in enabling control information (e.g. permissions records and associated component assemblies) created for distribution to other participants.

Some of the content redistribution control information (in this example, control information produced by a distributor within the scope permitted by a more senior participant in a chain of handling and control and passed to user/providers (in this example, user/distributors) and associated with controls and/or other requirements associated with redistribution activities by such user/distributors) specified by creator A includes: (a) a requirement that control information enabling content access may be redistributed by user/distributors no more than 2 levels, and further requires that each redistribution decrease this value by one, such that a first redistributor is restricted to two levels of redistribution, and a second redistributor to whom the first redistributor delivers permissions will be restricted to one additional level of redistribution, and users receiving permissions from the second redistributor will be unable to perform further redistribution (such a restriction may be enforced, for example, by including as one aspect of a VDE

control method associated with creating new permissions a requirement to invoke one or more methods that: (i) locate the current level of redistribution stored, for example, as an integer value in a UDE associated with such one or more methods, (ii) compare the level of redistribution value to a limiting value, and (iii) if such level of redistribution value is less than the limiting value, increment such level of redistribution value by one before delivering such a UDE to a user as an aspect of content control information associated with VDE managed content, or fail the process if such value is equal to such a limiting value), and (b) no other special restrictions are placed on redistributors.

Some of the usage control information (in this example, control information that a creator requires a distributor to provide in control information passed to users and/or user/distributors) specified by creator A may include, for example: (a) no moves (a form of distribution explained elsewhere in this document) of the content are permitted, and (b) distributors will be required to preserve (at a minimum) sufficient metering information within usage permissions in order to calculate the number of users who have accessed the container in a month and to prevent further usage after a rental has expired (e.g. by using a meter method designed to report access usages to creator A through a chain of handling and

reporting, and/or the use of expiration dates and/or time-aged encryption keys within a permissions record or other required control information).

Some of the extracting and/or embedding control information specified by creator A in this example may include a requirement that no extracting and/or embedding of the content is or will be permitted by parties in a chain of handling and control associated with this control information, except for users who have no redistribution rights related to such VDE secured content provided by Creator A. Alternatively, or in addition, as regards different portions of said content, control information enabling certain extraction and/or embedding may be provided along with the redistribution rights described in this example for use by user/distributors (who may include user content aggregators, that is they may provide content created by, and/or received from, different sources so as to create their own content products).

Distributor A in this example has selected a basic approach that distributor A prefers when offering enabling content control information to users and/or user/distributors that favors rental of content access rights over other approaches. In this example, some of the control information provided by

creators will permit distributor A to fulfill this favored approach directly, and other control structures may disallow this favored approach (unless, for example, distributor A completes a successful VDE negotiation allowing such an approach and supporting appropriate control information). Many of the control structures received by distributor A, in this example, are derived from (and reflect the results of) a VDE negotiation process in which distributor A indicates a preference for distribution control information that authorizes the creation of usage control information reflecting rental based usage rights. Such distribution control information may allow distributor A to introduce and/or modify control structures provided by creators in such a way as to create control information for distribution to users and/or user/distributors that, in effect, "rent" access rights. Furthermore, distributor A in this example services requests from user/distributors for redistribution rights, and therefore also favors distribution control information negotiated (or otherwise agreed to) with creators that permits distributor A to include such rights as an aspect of control information produced by distributor A.

In this example, distributor A and creator A may use VDE to negotiate (for example, VDE negotiate) for a distribution relationship. Since in this example creator A has produced a

VDE content container and associated control information that indicates creator A's desire to receive compensation based on rental of usage rights, and such control information further indicates that creator A has placed acceptable restrictions in redistribution control information that distributor A may use to service requests from user/distributors, distributor A may accept creator A's distribution control information without any negotiated changes.

After receiving enabling distribution control information from creator A, distributor A may manipulate an application program to specify some or all of the particulars of usage control information for users and/or user/distributors enabled by distributor A (as allowed, or not prevented, by senior control information). Distributor A may, for example, determine that a price of $15 per month per user would meet distributor A's business objectives with respect to payments from users for creator A's container. Distributor A must specify usage control information that fulfill the requirements of the distribution control information given to distributor A by creator A. For example, distributor A may include any required expiration dates and/or time-aged encryption keys in the specification of control information in accordance with creator A's requirements. If distributor A failed to include such information (or to meet

other requirements) in their specification of control information, the control method(s) referenced in creator A's permissions record and securely invoked within a PPE 650 to actually create this control information would, in this example, fail to execute in the desired way (e.g. based on checks of proposed values in certain fields, a requirement that certain methods be included in permissions, etc.) until acceptable information were included in distributor A's control information specification.

In this example, user A may have established an account with distributor A such that user A may receive VDE managed content usage control information from distributor A. User A may receive content usage control information from distributor A to access and use creator A's content. Since the usage control information has passed through (and been added to, and/or modified by) a chain of handling including distributor A, the usage control information requested from distributor A to make use of creator A's content will, in this example, reflect a composite of control information from creator A and distributor A. For example, creator A may have established a meter method that will generate an audit record if a user accesses creator A's VDE controlled content container if the user has not previously accessed the container within the same calendar month (e.g. by storing the date of the user's last access in a UDE associated

with an open container event referenced in a method core of such a meter method and comparing such a date upon subsequent access to determine if such access has occurred within the same calendar month). Distributor A may make use of such a meter method in a control method (e.g. also created and/or provided by creator A, or created and/or provided by distributor A) associated with opening creator A's container that invokes one or more billing and/or budget methods created, modified, referenced in one or more permissions records and/or parameterized by distributor A to reflect a charge for monthly usage as described above. If distributor A has specified usage and/or redistribution control information within the boundaries permitted by creator A's senior control information, a new set of control information (shown as $D_A(C_A)$ in Figure 80) may be associated with creator A's VDE content container when control information associated with that container by distributor A are delivered to users and/or user/distributors (user A, user B, and user/distributor A in this example).

In this example, user A may receive control information related to creator A's VDE content container from distributor A. This control information may represent an extended agreement between user A and distributor A (e.g. regarding fees associated with use of content, limited redistribution rights, etc.) and

distributor A and creator A (e.g. regarding the character, extent, handling, reporting, and/or other aspects of the use and/or creation of VDE controlled content usage information and/or content control information received, for example, by distributor A from creator A, or vice versa, or in other VDE content usage information handling). Such an extended agreement is enforced by processes operating within a secure subsystem of each participant's VDE installation. The portion of such an extended agreement representing control information of creator A as modified by distributor A in this example is represented by $D_A(C_A)$, including, for example, (a) control structures (e.g. one or more component assemblies, one or more permissions records, etc.), (b) the recording of usage information generated in the course of using creator A's content in conformance with requirements stated in such control information, (c) making payments (including automatic electronic credit and/or currency payments "executed" in response to such usage) as a consequence of such usage (wherein such consequences may also include electronically, securely and automatically receiving a bill delivered through use of VDE, wherein such a bill is derived from said usage), (d) other actions by user A and/or a VDE secure subsystem at user A's VDE installation that are a consequence of such usage and/or such control information.

In addition to control information $D_A(C_A)$, user A may enforce her own control information on her usage of creator A's VDE content container (within the limits of senior content control information). This control information may include, for example, (a) transaction, session, time based, and/or other thresholds placed on usage such that if such thresholds (e.g. quantity limits, for example, self imposed limits on the amount of expenditure per activity parameter) are exceeded user A must give explicit approval before continuing, (b) privacy requirements of user A with respect to the recording and/or transmission of certain usage related details relating to user A's usage of creator A's content, (c) backup requirements that user A places on herself in order to help ensure a preservation of value remaining in creator A's content container and/or local store of electronic credit and/or currency that might otherwise be lost due to system failure or other causes. The right to perform in some or all of these examples of user A's control information, in some examples, may be negotiated with distributor A. Other such user specified control information may be enforced independent of any control information received from any content provider and may be set in relationship to a user's, or more generally, a VDE installation's, control information for one or more classes, or for all classes, of content and/or electronic appliance usage. The entire set of VDE control information that may be in place

during user A's usage of creator A's content container is referred to on Figure 80 as $U_A(D_A(C_A))$. This set may represent the control information originated by creator A, as modified by distributor A, as further modified by user A, all in accordance with control information from value chain parties providing more senior control information, and therefore constitutes, for this example, a "complete" VDE extended agreement between user A, distributor A, and creator A regarding creator A's VDE content container. User B may, for example, also receive such control information $D_A(C_A)$ from distributor A, and add her own control information in authorized ways to form the set $U_B(D_A(C_A))$.

User/distributor A may also receive VDE control information from distributor A related to creator A's VDE content container. User/distributor A may, for example, both use creator A's content as a user and act as a redistributor of control information. In this example, control information $D_A(C_A)$ both enables and limits these two activities. To the extent permitted by $D_A(C_A)$, user/distributor A may create their own control information based on $D_A(C_A)$ -- $UD_A(D_A(C_A))$ -- that controls both user/distributor A's usage (in a manner similar to that described above in connection with user A and user B), and control information redistributed by user/distributor A (in a manner similar to that described above in connection with distributor A).

For example, if user/distributor A redistributes $UD_A(D_A(C_A))$ to user/distributor B, user/distributor B may be required to report certain usage information to user/distributor A that was not required by either creator A or distributor A. Alternatively or in addition, user/distributor B may, for example, agree to pay user/distributor A a fee to use creator A's content based on the number of minutes user/distributor B uses creator A's content (rather than the monthly fee charged to user/distributor A by distributor A for user/distributor B's usage).

In this example, user/distributor A may distribute control information $UD_A(D_A(C_A))$ to user/distributor B that permits user/distributor B to further redistribute control information associated with creator A's content. User/distributor B may make a new set of control information $UD_B(UD_A(D_A(C_A)))$. If the control information $UD_A(D_A(C_A))$ permits user/distributor B to redistribute, the restrictions on redistribution from creator A in this example will prohibit the set $UD_B(UD_A(D_A(C_A)))$ from including further redistribution rights (e.g. providing redistribution rights to user B) because the chain of handling from distributor A to user/distributor A (distribution) and the continuation of that chain from user/distributor A to user/distributor B (first level of redistribution) and the further continuation of that chain to another user represents two levels

of redistribution, and, therefore, a set $UD_B(UD_A(D_A(C_A)))$ may not, in this example, include further redistribution rights.

As indicated in Figure 79, user B may employ content from both user/distributor B and distributor A (amongst others). In this example, as illustrated in Figure 80, user B may receive control information associated with creator A's content from distributor A and/or user/distributor B. In either case, user B may be able to establish their own control information on $D_A(C_A)$ and/or $UD_B(UD_A(D_A(C_A)))$, respectively (if allowed by such control information. The resulting set(s) of control information, $U_B(D_A(C_A))$ and/or $U_B(UD_B(UD_A(D_A(C_A))))$ respectively, may represent different control scenarios, each of which may have benefits for user B. As described in connection with an earlier example, user B may have received control information from user/distributor B along a chain of handling including user/distributor A that bases fees on the number of minutes that user B makes use of creator A's content (and requiring user/distributor A to pay fees of $15 per month per user to distributor A regardless of the amount of usage by user B in a calendar month). This may be more favorable under some circumstances than the fees required by a direct use of control information provided by distributor A, but may also have the disadvantage of an exhausted chain of redistribution and, for

example, further usage information reporting requirements included in $UD_B(UD_A(D_A(C_A)))$. If the two sets of control information $D_A(C_A)$ and $UD_B(UD_A(D_A(C_A)))$ permit (e.g. do not require exclusivity enforced, for example, by using a registration interval in an object registry used by a secure subsystem of user B's VDE installation to prevent deregistration and reregistration of different sets of control information related to a certain container (or registration of plural copies of the same content having different control information and/or being supplied by different content providers) within a particular interval of time as an aspect of an extended agreement for a chain of handling and control reflected in $D_A(C_A)$ and/or $UD_B(UD_A(D_A(C_A)))$, user B may have both sets of control information registered and may make use of the set that they find preferable under a given usage scenario.

In this example, creator B creates a VDE content container and associates a set of VDE control information with such container indicated in Figure 81 as C_B. Figure 81 further shows the VDE participants who may receive enabling control information related to creator B's VDE content container. In this example, control information may indicate that distributors of creator B's content: (a) must pay creator B $0.50 per kilobyte of information decrypted by users and/or user/distributors

authorized by such a distributor, (b) may allow users and/or user/distributors to embed their content container in another container while maintaining a requirement that creator B receive $0.50 per kilobyte of content decrypted, (c) have no restrictions on the number of enabling control information sets that may be generated for users and/or user/distributors, (d) must report information concerning the number of such distributed control information sets at certain time intervals (e.g. at least once per month), (e) may create control information that allows users and/or user/distributors to perform up to three moves of their control information, (f) may allow redistribution of control information by user/distributors up to three levels of redistribution, (g) may allow up to one move per user receiving redistributed control information from a user/distributor.

In this example, distributor A may request control information from creator B that enables distributor A to distribute control information to users and/or user/distributors that is associated with the VDE container described above in connection with creator B. As stated earlier, distributor A has established a business model that favors "rental" of access rights to users and user/distributors receiving such rights from distributor A. Creator B's distribution control information in this example does not force a model including "rental" of rights,

but rather bases payment amounts on the quantity of content decrypted by a user or user/distributor. In this example, distributor A may use VDE to negotiate with creator B to include a different usage information recording model allowed by creator B. This model may be based on including one or more meter methods in control structures associated with creator B's container that will record the number of bytes decrypted by end users, but not charge users a fee based on such decryptions; rather distributor A proposes, and creator B's control information agrees to allow a "rental" model to charge users, and determines the amount of payments to creator B based on information recorded by the bytes decrypted meter methods and/or collections of payment from users.

Creator B may, for example, (a) accept such a new control model with distributor A acting as the auditor (e.g. trusting a control method associated with processing audit information received by distributor A from users of creator B's content using a VDE secure subsystem at distributor A's site, and further to securely calculate amounts owed by distributor A to creator B and, for example, making payments to creator B using a mutually acceptable budget method managing payments to creator B from credit and/or currency held by distributor A), (b) accept such a new control model based on distributor A's

acceptance of a third party to perform all audit functions associated with this content, (c) may accept such a model if information associated with the one or more meter methods that record the number of bytes decrypted by users is securely packaged by distributor B's VDE secure subsystem and is securely, employing VDE communications techniques, sent to creator B in addition to distributor A, and/or (d) other mutually acceptable conditions. Control information produced by distributor A based on modifications performed by distributor A as permitted by C_B are referred to in this example as $D_A(C_B)$.

User A may receive a set of control information $D_A(C_B)$ from distributor A. As indicated above in connection with content received from creator A via a chain of handling including distributor A, user A may apply their own control information to the control information $D_A(C_B)$, to the extent permitted by $D_A(C_B)$, to produce a set of control information $U_A(D_A(C_B))$. The set of control information $D_A(C_B)$ may include one or more meter methods that record the number of bytes of content from creator B's container decrypted by user A (in order to allow correct calculation of amounts owed by distributor A to creator B for user A's usage of creator B's content in accordance with the control information of C_B that requires payment of $0.50 per kilobyte of decrypted information), and a further meter method

associated with recording usage such that distributor A may gather sufficient information to securely generate billings associated with user A's usage of creator B's content and based on a "rental" model (e.g. distributor A may, for example, have included a meter method that records each calendar month that user A makes use of creator B's content, and relates to further control information that charges user A $10 per month for each such month during which user A makes use of such content.)

User/distributor A may receive control information C_B directly from creator B. In this case, creator B may use VDE to negotiate with user/distributor A and deliver a set of control information C_B that may be the same or differ from that described above in connection with the distribution relationship established between creator B and distributor A. For example, user/distributor A may receive control information C_B that includes a requirement that user/distributor A pay creator B for content decrypted by user/distributor A (and any participant receiving distributed and/or redistributed control information from user/distributor A) at the rate of $0.50 per kilobyte. As indicated above, user/distributor A also may receive control information associated with creator B's VDE content container from distributor A. In this example, user/distributor A may have a choice between paying a "rental" fee through a chain of

handling passing through distributor A, and a fee based on the quantity of decryption through a chain of handling direct to creator B. In this case, user/distributor A may have the ability to choose to use either or both of C_B and $D_A(C_B)$. As indicated earlier in connection with a chain of handling including creator A and distributor A, user/distributor A may apply her own control information to the extent permitted by C_B and/or $D_A(C_B)$ to form the sets of control information $UD_A(C_B)$ and $UD_A(D_A(C_B))$, respectively.

As illustrated in Figure 81, in this example, user B may receive control information associated with creator B's VDE content container from six different sources: C_B directly from creator B, $D_A(C_B)$ from distributor A, $UD_B(UD_A(D_A(C_B)))$ and/or $UD_B(UD_A(C_B))$ from user/distributor B, $D_C(C_B)$ from distributor C, and/or $D_B(D_C(C_B))$ from distributor B. This represents six chains of handling through which user B may enter into extended agreements with other participants in this example. Two of these chains pass through user/distributor B. Based on a VDE negotiation between user/distributor B and user B, an extended agreement may be reached (if permitted by control information governing both parties) that reflects the conditions under which user B may use one or both sets of control information. In this example, two chains of handling and control

may "converge" at user/distributor B, and then pass to user B (and if control information permits, later diverge once again based on distribution and/or redistribution by user B).

In this example, creator C produces one or more sets of control information C_C associated with a VDE content container created by creator C, as shown in Figure 82. Figure 82 further shows the VDE participants who may receive enabling control information related to creator C's VDE content container. The content in such a container is, in this example, organized into a set of text articles. In this example control information may include one or more component assemblies that describe the articles within such a container (e.g. one or more event methods referencing map tables and/or algorithms that describe the extent of each article). C_C may further include, for example: (a) a requirement that distributors ensure that creator C receive $1 per article accessed by users and/or user/distributors, which payment allows a user to access such an article for a period of no more than six months (e.g. using a map-type meter method that is aged once per month, time aged decryption keys, expiration dates associated with relevant permissions records, etc.), (b) control information that allows articles from creator C's container to be extracted and embedded into another container for a one time charge per extract/embed of $10, (c) prohibits

extracted/embedded articles from being reextracted, (d) permits distributors to create enabling control information for up to 1000 users or user/distributors per month, (e) requires that information regarding the number of users and user/distributors enabled by a distributor be reported to creator C at least once per week, (f) permits distributors to enable users or user/distributors to perform up to one move of enabling control information, and (g) permits up to 2 levels of redistribution by user/distributors.

In this example, distributor B may establish a distribution relationship with creator C. Distributor B in this example may have established a business model that favors the distribution of control information to users and user/distributors that bases payments to distributor B based on the number of accesses performed by such VDE participants. In this example, distributor B may create a modified set $D_B(C_C)$ of enabling control information for distribution to users and/or user/distributors. This set $D_B(C_C)$ may, for example, be based on a negotiation using VDE to establish a fee of $0.10 per access per user for users and/or user/distributors who receive control information from distributor B. For example, if one or more map-type meter methods have been included in C_C to ensure that adequate information may be gathered from users and/or user/distributors to ensure correct payments to creator C by

distributor B based on C_C, such methods may be preserved in the set $D_B(C_C)$, and one or more further meter methods (and any other necessary control structures such as billing and/or budget methods) may be included to record each access such that the set $D_B(C_C)$ will also ensure that distributor B will receive payments based on each access.

The client administrator in this example may receive a set of content control information $D_B(C_C)$ that differs, for example, from control information received by user B from distributor B. For example, the client administrator may use VDE to negotiate with distributor B to establish a set of control information for content from all creators for whom distributor B may provide enabling content control information to the client administrator. For example, the client administrator may receive a set of control information $D_B(C_C)$ that reflects the results of a VDE negotiation between the client administrator and distributor B. The client administrator may include a set of modifications to $D_B(C_C)$ and form a new set $CA(D_B(C_C))$ that includes control information that may only be available to users and user/distributors within the same organization as the client administrator (e.g. coworkers, employees, consultants, etc.) In order to enforce such an arrangement, $CA(D_B(C_C))$ may, for example, include control structures that examine name services information associated

with a user or user/distributor during registration, establish a new budget method administered by the client administrator and required for use of the content, etc.

A distributor may provide redistribution rights to a client administrator which allows said administrator to redistribute rights to create permissions records for certain content (redistribute rights to use said content) only within the administrator's organization and to no other parties. Similarly, such administrator may extend such a "limited" right to redistribute to department and/or other administrator within his organization such that they may redistribute such rights to use content based on one or more restricted lists of individuals and/or classes and/or other groupings of organization personnel as defined by said administrator. This VDE capability to limit redistribution to certain one or more parties and/or classes and/or other groupings of VDE users and/or installations can be applied to content by any VDE content provider, so long as such a control is allowed by senior control information.

User D in this example may receive control information from either the client administrator and/or user/distributor C. User/distributor C may, for example, distribute control information $UD_C(CA(D_B(C_C)))$ to user D that includes a

departmental budget method managed by user/distributor C to allow user/distributor C to maintain an additional level of control over the actions of user D. In this case, $UD_C(CA(D_B(C_C)))$ may include multiple levels of organizational controls (e.g. controls originating with the client administrator and further controls originating with user/distributor C) in addition to controls resulting from a commercial distribution channel. In addition or alternatively, the client administrator may refuse to distribute certain classes of control information to user D even if the client administrator has adequate control information (e.g. control information distributed to user/distributor C that allows redistribution to users such as user D) to help ensure that control information flows through the client administrator's organization in accordance with policies, procedures, and/or other administrative processes.

In this example, user E may receive control information from the client administrator and/or distributor B. For example, user E may have an account with distributor B even though some control information may be received from the client administrator. In this case, user E may be permitted to request and receive control information from distributor B without restriction, or the client administrator may have, as a matter of organizational policy, control information in place associated

with user E's electronic appliance that limits the scope of user E's interaction with distributor B. In the latter case, the client administrator may, for example, have limited user E to registering control information with the secure subsystem of user E's electronic appliance that is not available from the client administrator, is from one or more certain classes of distributors and/or creators, and/or has a cost for usage, such as a certain price point (e.g. $50 per hour of usage). Alternatively or in addition, the client administrator may, for example, limit user E to receiving control information from distributor B in which user E receives a more favorable price (or other control information criteria) than the price (or other criteria) available in control information from the client administrator.

In this example, creator D may create a VDE content container that is designed primarily for integration with other content (e.g. through use of a VDE extracting/embedding process), for example, content provided by creator B and creator C. Figure 83 shows the VDE participants who may receive enabling control information related a VDE content container produced by creator D. Control information associated with creator D's content (C_D in Figure 83) may include, for example: (a) a requirement that distributors make payment of either $1.50 per open per user, or $25 per user for an unlimited number of

opens, (b) a discount of 20% for any user that has previously paid for an unlimited number of opens for certain other content created by creator D (e.g. implemented by including one or more billing methods that analyze a secure database of a user's VDE installation to determine if any of such certain other containers are registered, and further determines the character of rights held by a user purchasing rights to this container), (c) a requirement that distributors report the number of users and user/distributors enabled by control information produced in accordance with C_D after such number exceeds 1000, (d) a requirement that distributors limit the number of moves by users and/or user/distributors to no more than one, (e) a requirement that distributors limit user/distributors to no more than four levels of redistribution, and (f) that distributors may create enabling control information that permits other distributors to create control information as distributors, but may not pass this capability to such enabled distributors, and further requires that audit information associated with use of control information by such enabled distributors shall pass directly to creator D without processing by such enabling distributor and that creator D shall pay such an enabling distributor 10% of any payments received by creator D from such an enabled distributor.

In this example, distributor C may receive VDE content containers from creator B, creator C, and creator D, and associated sets of control information C_B, C_C, and C_D. Distributor C may use the embedding control information and other control information to produce a new container with two or more VDE objects received from creator B, creator C, and creator D. In addition or alternatively, distributor C may create enabling control information for distribution to users and/or user/distributors (or in the case of C_D, for distributors) for such received containers individually. For example, distributor C may create a container including content portions (e.g. embedded containers) from creator B, creator C, and creator D in which each such portion has control information related to its access and use that records, and allows an auditor to gather, sufficient information for each such creator to securely and reliably receive payments from distributor C based on usage activities related to users and/or user/distributors enabled by distributor C. Furthermore, distributor C may negotiate using VDE with some or all of such creators to enable a model in which distributor C provides overall control information for the entire container based on a "uniform" fee (e.g. calculated per month, per access, from a combined model, etc.) charged to users and/or user/distributors, while preserving the models of each such creator with respect to payments due to them by distributor C

based on C_B, C_C, and/or C_D, and, for example, resulting from each of their differing models for the collection of content usage information and any related (e.g. advertising) information.

In this example, distributor B may receive a VDE content container and associated content control information C_E from creator E as shown in Figure 83. If C_E permits, distributor B may extract a portion of the content in such a container. Distributor B may then, for example, embed this portion in a container received from distributor C that contains an aggregation of VDE objects created by creator B, creator C, and creator D. Depending on the particular restrictions and/or permissions in the sets of control information received from each creator and distributor C, distributor B may, for example, be able to embed such an extracted portion into the container received from distributor C as an independent VDE object, or directly into content of "in place" objects from creator B, creator C, and/or creator D. Alternatively, or in addition, distributor B may, if permitted by C_E, choose to distribute such an extracted portion of content as an independent VDE object.

User B may, in this example, receive a VDE content container from distributor C that is comprised of VDE objects created by creator B, creator C, and creator D. In addition, user

B may receive a VDE content container from distributor B that contains the same content created by creator B, creator C, and creator D in addition to one or more extracted/embedded portions of content created by creator E. User B may base decisions concerning which of such containers they choose to use (including which embedded containers she may wish to use), and under which circumstances, based on, for example, the character of such extracted/embedded portions (e.g. multimedia presentations illustrating potential areas of interest in the remainder of the content, commentary explaining and/or expositing other elements of content, related works, improved application software delivered as an element of content, etc.); the quality, utility, and/or price (or other attributes of control information) of such portions; and other considerations which distinguish the containers and/or content control information received, in this example, from distributor B and distributor C.

User B may receive content control information from distributor B for such a VDE content container that permits user B to add and/or modify content contained therein. User B may, for example, desire an ability to annotate content in such a container using a VDE aware word processor or other application(s). If permitted by senior control information, some or all of the content may be available to user B for modification

and/or additions. In this case, user B is acting as a VDE creator for added and/or modified content. User B may, for example, provide new control information for such content, or may be required (or desire to) make use of existing control information (or control information included by senior members of a chain of handling for this purpose) to manage such content (based on control information related to such a container and/or contained objects).

In this example, VDE 100 has been used to enable an environment including, for example, content distribution, redistribution, aggregation (extracting and/or embedding), reaggregation, modification, and usage. The environment in this example allows competitive models in which both control information and content may be negotiated for and have different particulars based on the chain of handling through which control information and/or content has been passed. Furthermore, the environment in this example permits content to be added to, and/or modified by, VDE participants receiving control information that enables such activities.

Example -- Content Distribution Through a Content VDE Chain of Handling

Figure 84 reflects certain aspects of a relatively simple model 3400 of VDE content distribution involving several categories of VDE participants. In this instance, and for simplicity of reference purposes, various portions of content are represented as discrete items in the form of VDE content container objects. One or more of such content portions may also be integrated together in a single object and may (as may the contents of any VDE content container object if allowed by content control information) be extracted in whole or part by a user. In this example, publishers of historical/educational multimedia content have created VDE content containers through the use of content objects available from three content resources:

- a Video Library 3402 product available to Publishers on optical discs and containing video clip VDE objects representing various historical situations,

- an Internet Repository 3404 which stores history information text and picture resources in VDE objects which are available for downloading to Publishers and other users, and

- an Audio Library 3406, also available on optical discs, and containing various pieces of musical performances and vocal performances (for example, historical narrations) which can be used alone or to accompany other educational historical materials.

The information provided in library 3402, repository 3404, and library 3406 may be provided to different publishers 3408(a), 3408(b), ..., 3408(n). Publishers 3408 may, in turn, provide some or all of the information they obtain to end users 3410.

In this example, the Video Library 3402 control information allows publishers to extract objects from the Video Library product container and content control information enabling use of each extracted object during a calendar year if the object has a license cost of $50 or less, and is shorter than 45 minutes in duration, and 20,000 copies of each of any other extracted objects, and further requires all video objects to be VDE fingerprinted upon decryption. The Audio Library 3404 has established similar controls that match its business model. The Internet Repository 3406 VDE containerizes, including encrypts, selected object content as it streams out of the Repository in response to an online, user request to download an object. The Repository 3406 may fingerprint the identification of the

receiving VDE installation into its content prior to encryption and communication to a publisher, and may further require user identification fingerprinting of their content when decrypted by said Publisher or other content user.

The Publishers 3408 in this example have selected, under terms and conditions VDE negotiated (or otherwise agreed to) with the providing resources, various content pieces which they combine together to form their VDE object container products for their teacher customers. Publisher 3408(A) has combined video objects extracted from the Video Library 3402 (as indicated by circles), text and image objects extracted from the Internet Repository 3404 (indicated by diamonds), and one musical piece and one historical narration extracted from the Audio Library 3406 (as indicated by rectangles). Publisher 3408(B) has extracted a similar array of objects to be combined into his product, and has further added graphical elements (indicated by a hexagon) created by Publisher 3408(B) to enhance the product. Publisher 3408(C) has also created a product by combining objects from the Internet Repository 3404 and the Audio Library 3406. In this example, all publisher products are delivered, on their respective optical discs, in the form of VDE content container objects with embedded objects, to a modern high school for installation on the high school's computer network.

In this particular example, End-Users 3410 are teachers who use their VDE node's secure subsystems to access the VDE installation on their high school server that supports the publishers' products (in an alternative example, the high school may maintain only a server based VDE installation). These teachers license the VDE products from one or more of the publishers and extract desired objects from the VDE product content containers and either download the extracted VDE content in the form of VDE content containers for storage on their classroom computers and/or as appropriate and/or efficient. The teachers may store extracted content in the form of VDE content containers on server mass storage (and/or if desired and available to an end-user, and further according to acceptable pricing and/or other terms and conditions and/or senior content control information, they may store extracted information in "clear" unencrypted form on their nodes' and/or server storage means). This allows the teachers to play, and/or otherwise use, the selected portions of said publishers' products, and as shown in two instances in this example, add further teacher and/or student created content to said objects. End-user 3410(2), for example, has selected a video piece 1 received from Publisher A, who received said object from the Video Library. End-user 3410(3) has also received a video piece 3 from the same Publisher 3408(A) wherein said piece was also available to her from

Publisher 3408(B), but perhaps under not as favorable terms and conditions (such as a support consultation telephone line). In addition, end-user 3410(3) has received an audio historical narration from Publisher 3408(B) which corresponds to the content of historical reference piece 7. End-user 3410(3) has also received a corresponding historical reference piece 7 (a book) from publisher 3408(2) who received said book from the Internet Repository 3404. In this instance, perhaps publisher 3408(2) charged less for said book because end-user 3410(3) has also licensed historical reference piece 7 from him, rather than publisher 3408(1), who also carried the same book. End-user 3410(3), as a teacher, has selected the items she considers most appropriate for her classes and, through use of VDE, has been able to flexibly extract such items from resources available to her (in this instance, extracting objects from various optical products provided by publishers and available on the local high school network server).

Example -- Distribution of Content Control Information Within an Organization

Figure 85 shows two VDE content containers, Container 300(A) and Container 300(B), that have been distributed to a VDE Client Administrator 3450 in a large organization. As shown in the figure, Container 300(A) and Container 300(B), as

they arrive at the corporation, carry certain control information specifying available usage rights for the organization. As can be further seen in Figure 85, the client administrator 3450 has distributed certain subsets of these rights to certain department administrators 3452 of her organization, such as Sales and Marketing Administrator 3452(1), Planning Administrator 3452(2), and Research and Development Administrator 3452(k). In each instance, the Client Administrator 3450 has decided which usage options and how much budget should be made available to each department.

Figure 85 is a simplified example and, for example, the Client Administrator 3450 could have added further VDE controls created by herself and/or modified and/or deleted in place controls (if allowed by senior content control information) and/or (if allowed by control information) she could have further divided the available monetary budget (or other budgets) among specific usage activities. In this example, departmental administrators have the same rights to determine the rights of departmental end-users as the client administrator has in regard to departments. In addition, in this example (but not shown in Figure 85) the client administrator 3450 and/or content provider(s) may also determine certain control information which must directly control (including providing rights related to) end-

user content usage and/or the consequences of said usage for all or certain classes of end-users. In the example shown in Figure 85, there are only three levels of VDE participants within the organization:

a Client Administrator 3450,

department administrators 3452, and

end-users 3454.

In other examples, VDE will support many levels of VDE administration (including overlapping groups) within an organization (e.g., division, department, project, network, group, end-users, etc). In addition, administrators in a VDE model may also themselves be VDE content users.

Within an organization, VDE installations may be at each end-user 3454 node, only on servers or other multiple user computers or other electronic appliances, or there may be a mixed environment. Determination as to the mix of VDE server and/or node usage may be based on organization and/or content provider security, performance, cost overhead, or other considerations.

In this example, communications between VDE participants in Figure 85 employs VDE secure communication techniques between VDE secure subsystems supporting PPEs

and other VDE secure system components at each VDE installation within the organization.

Example -- Another Content Distribution Example

Creators of VDE protected content may interact with other VDE participants in many different ways. A VDE creator 102 may, for example, distribute content and/or content control information directly to users, distribute content and/or content control information to commercial content repositories, distribute content and/or content control information to corporate content repositories, and/or distribute content and/or content control information to other VDE participants. If a creator 102 does not interact directly with all users of her content, she may transmit distribution permissions to other VDE participants that permit such participants to further distribute content and/or content control information. She may also allow further distribution of VDE content and/or content control information by, for example, not restricting redistribution of control information, or allowing a VDE participant to act as a "conduit" for one or more permissions records that can be passed along to another party, wherein said permissions record provides for including the identification of the first receiving party and/or the second receiving party.

Figure 86 shows one possible arrangement of VDE participants. In this example, creator 102 may employ one or more application software programs and one or more VDE secure subsystems to place unencrypted content into VDE protected form (i.e., into one or more VDE content containers). In addition, creator 102 may produce one or more distribution permissions 3502 and/or usage permissions 3500 as an aspect of control information associated with such VDE protected content. Such distribution and/or usage permissions 3500, 3502 may be the same (e.g., all distribution permissions may have substantively all the same characteristics), or they may differ based on the category and/or class of participant for whom they are produced, the circumstances under which they are requested and/or transmitted, changing content control models of either creator 102 or a recipient, etc.

In this example, creator 102 transmits (e.g., over a network, via broadcast, and/or through transfer of physical media) VDE protected content to user 112a, user 112b, and/or user 112c. In addition, creator 102 transmits, using VDE secure communications techniques, usage permissions to such users. User 112a, user 112b, and user 112c may use such VDE protected content within the restrictions of control information specified by usage permissions received from creator 102. In this

case, creator 102 may, for example, manage all aspects of such users activities related to VDE protected content transmitted to them by creator 102. Alternatively, creator 102 may, for example, include references to control information that must be available to users that is not provided by creator 102 (e.g., component assemblies managed by another party).

Commercial content repository 200g, in this example, may receive VDE protected (or otherwise securely delivered) content and distribution, permissions and/or other content usage control information from creator 102. Commercial content repository 200g may store content securely such that users may obtain such, when any required conditions are met, content from the repository 200g. The distribution permissions 3502 may, for example, permit commercial content repository 200g to create redistribution permissions and/or usage permissions 3500, 3502 using a VDE protected subsystem within certain restrictions described in content control information received from creator 102 (e.g., not to exceed a certain number of copies, requiring certain payments by commercial content repository 200g to creator 102, requiring recipients of such permissions to meet certain reporting requirements related to content usage information, etc.). Such content control information may be stored at the repository installation and be applied to

unencrypted content as it is transmitted from said repository in response to a user request, wherein said content is placed into a VDE container as a step in a secure process of communicating such content to a user. Redistribution permissions may, for example, permit a recipient of such permissions to create a certain number of usage permissions within certain restrictions (e.g., only to members of the same household, business other organization, etc.). Repository 200g may, for example, be required by control information received from creator 102 to gather and report content usage information from all VDE participants to whom the repository has distributed permissions.

In this example, power user 112d may receive VDE protected content and redistribution permissions from commercial content repository 200g using the desktop computer 3504. Power user 112d may, for example, then use application software in conjunction with a VDE secure subsystem of such desktop computer 3504 in order to produce usage permissions for the desktop computer 3504, laptop computer 3506 and/or settop appliance 3508 (assuming redistribution permissions received from commercial content repository 200g permit such activities). If permitted by senior control information (for example, from creator 102 as may be modified by the repository 200g), power user 112d may add her own restrictions to such usage

permissions (e.g., restricting certain members of power user 112d's household using the settop appliance to certain times of day, amounts of usage, etc. based on their user identification information). Power user 112d may then transmit such VDE protected content and usage permissions to the laptop computer 3506 and the settop appliance 3508 using VDE secure communications techniques. In this case, power user 112d has redistributed permissions from the desktop computer 3504 to the settop appliance 3508 and the laptop computer 3506, and periodically the settop appliance and the laptop computer may be required to report content usage information to the desktop computer, which in turn may aggregate, and/or otherwise process, and report user usage information to the repository 200g.

User 112e and/or user 112f may receive usage permissions and VDE protected content from commercial content repository 200g. These users may be able to use such content in ways authorized by such usage information. In contrast to power user 112d, these users may not have requested and/or received redistribution permissions from the repository 200g. In this case, these users may still be able to transfer some or all usage rights to another electronic appliance 600, and/or they may be permitted to move some of their rights to another electronic

appliance, if such transferring and/or moving is permitted by the usage permissions received from the repository 200g. In this case, such other appliances may be able to report usage information directly to the repository 200g.

In this example, corporate content repository 702 within corporation 700 may receive VDE protected content and distribution permissions from creator 102. The distribution permissions received by corporate repository 702 may, for example, include restrictions that limit repository 702 to distribution activities within corporation 700.

The repository 702 may, for example, employ an automated system operating in conjunction with a VDE secure subsystem to receive and/or transmit VDE protected content, and/or redistribution and/or usage permissions. In this case, an automated system may, for example, rely on criteria defined by corporate policies, departmental policies, and/or user preferences to determine the character of permissions and/or content delivered to various parties (corporation groups and/or individuals) within corporation 700. Such a system may, for example, automatically produce redistribution permissions for a departmental content repository 704 in response to corporation

700 receiving distribution permissions from creator 102, and/or produce usage permissions for user 112j and/or user 112k.

The departmental repository 704 may automatically produce usage permissions for user 112g, user 112h, and/or user 112i. Such users may access content from the corporate content repository 702, yet receive usage permissions from departmental repository 704. In this case, user 112g, user 112h, and/or user 112i may receive usage permissions from departmental repository 704 that incorporate departmental restrictions in addition to restrictions imposed by senior control information (in this example, from creator 102, as may be modified by corporate repository 702, as may be further modified by departmental repository 704, that reflect a VDE extended agreement incorporating commercial requirements of creator 102 and corporation 700 in addition to corporate and/or departmental policies and agreements with corporate personnel of corporation 700).

Example—"Virtual Silicon Container"

As discussed above, VDE in one example provides a "virtual silicon container" ("virtual black box") in that several different instances of SPU 500 may securely communicate together to provide an overall secure hardware environment that

"virtually" exists at multiple locations and multiple electronic appliances 600. Figure 87 shows one model 3600 of a virtual silicon container. This virtual container model 3600 includes a content creator 102, a content distributor 106, one or more content redistributors 106a, one or more client administrators 700, one or more client users 3602, and one or more clearinghouses 116. Each of these various VDE participants has an electronic appliance 600 including a protected processing environment 655 that may comprise, at least in part, a silicon-based semiconductor hardware element secure processing unit 500. The various SPUs 500 each encapsulate a part of the virtual distribution environment, and thus, together form the virtual silicon container 3600.

Example -- Testing/Examinations

A scheduled SAT examination for high school seniors is prepared by the Educational Testing Service. The examination is placed in a VDE container for scheduled release on November 15, 1994 at 1:00 PM Eastern Standard time. The SAT prepares one copy of the container for each school or other location which will conduct the examination. The school or other location ("test site") will be provided with a distributed examination container securely containing the VDE identification for the "administration" electronic appliance and/or test administrator

at the test site (such as, a testing organization) and a budget enabling, for example, the creation of 200 test VDE content containers. Each container created at the test site may have a permissions record containing secure identification information for each electronic appliance 600, on the test site's network, that will be used by a test taker, as well as, for example, an identification for the student who will take the test. The student identification could, for example, be in the form of a secure PIN password which is entered by the student prior to taking the test (a test monitor or administrator might verify the student identification by entering in a PIN password). Of course, identification might take the firm of automated voice recognition, handwriting recognition (signature recognition), fingerprint information, eye recognition, or similar one or more recognition forms which may be used either to confirm the identity of the test taker (and/or test monitor/administrator) and/or may be stored with the test results in a VDE container or the like or in a location pointed to by certain container information. This identification may be stored in encrypted or unencrypted form. If stored in encrypted or otherwise protected form, certain summary information, such as error correction information, may be stored with the identification information to authenticate the associated test as corresponding to the identification.

As the student takes the test using the computer terminal, the answers selected may be immediately securely stored (but may be changed by the student during the test session). Upon the completion of the test, the student's answers, along with a reference to the test, are securely stored in a VDE reporting object which is passed along to the network to the test administrator and the administration electronic appliance 600. All test objects for all students could then be placed in a VDE object 300 for communication to the Educational Testing Service, along with whatever other relevant information (which may also be secured by VDE 100), including summary information giving average and mean scores, and other information that might be desirable to summarize and/or act as an authentication of the test objects sent. For example, certain information might be sent separately from each student summary object containing information which helps validate the object as an "authentic" test object.

Applying VDE to testing scenarios would largely eliminate cheating resulting from access to tests prior to testing (normally the tests are stolen from a teacher or test administrator). At ETS, individuals who have access to tests could be limited to only a portion of the test to eliminate the risk of the theft of a "whole" test. Employing VDE would also ensure against processing

errors or other manipulation of test answers, since absolutely authentic test results can be archived for a reasonable period of time.

Overall, employing VDE 100 for electronic testing will enable the benefits of electronic testing to be provided without the substantial risks associated with electronic storing, communicating, and processing of test materials and testing results. Electronic testing will provide enormous efficiency improvements, significantly lowering the cost of conducting and processing tests by eliminating printing, shipping, handling, and human processing of tests. At the same time, electronic testing will allow users to receive a copy (encrypted or unencrypted) of their test results when they leave the test sessions. This will help protect the tested individual against lost of, or improperly processed, test results. Electronic testing employing VDE 100 may also ensure that timing related variables of testing (for example precise starting, duration, and stopping times) can be reliably managed. And, of course, proper use of VDE 100 for the testing process can prevent improper access to test contents prior to testing and ensure that test taking is properly audited and authenticated, that is which person took which test, at which time, on which electronic appliance, at which location. Retesting

due to lost, stolen, improperly timed, or other variables can be avoided or eliminated.

VDE assisted testing may, of course, be employed for many different applications including secure identification of individuals for security/authentication purposes, for employment (e.g. applying for jobs) applications, and for a full range of evaluation testing. For example, an airline pilot, or a truck, train, or bus driver might take a test immediately prior to departure or during travel, with the test evaluating alertness to test for fatigue, drug use, etc. A certain test may have a different order and/or combination of test activities each time, or each group of times, the test is taken. The test or a master test might be stored in a VDE container (the order of, and which, test questions might be determined by a process executed securely within an PPE 650). The test responses may be encrypted as they occur and either locally stored for aggregated (or other test result) transmission or dynamically transmitted (for example, to a central test administration computer). If the test taker "flunks" the test, perhaps he or she is then prevented from operating the vehicle, either by a local PPE 650 issuing control instructions to that effect on some portion of the vehicle's electronic control system or a local PPE failing to decrypt or

otherwise provide certain key information required for vehicle operation.

Example -- Appliance Rental

Through use of the present invention, electronic appliances can be "leased" or otherwise provided to customers who, rather than purchasing a given appliance for unlimited usage, may acquire the appliance (such as a VCR, television, microwave oven, etc.) and be charged according to one or more aspects of use. For example, the charge for a microwave might be for each time it is used to prepare an item and/or for the duration of time used. A telephone jack could be attached, either consistently or periodically, to an inexpensive modem operatively attached or within the microwave (the modem might alternatively be located at a location which services a plurality of items and/or functions -- such as burglar alarm, light and/or heat control). Alternatively, such appliances may make use of a network formed by the power cables in a building to transmit and receive signals.

At a periodic interval, usage information (in summary form and/or detailed) could be automatically sent to a remote information utility that collects information on appliance usage (the utility might service a certain brand, a certain type of appliance, and/or a collection of brands and/or types). The usage

information would be sent in VDE form (e.g. as a VDE object 300). The information utility might then distribute information to financial clearinghouse(s) if it did not itself perform the billing function, or the information "belonging" to each appliance manufacturer and/or lessor (retailer) might be sent to them or to their agents. In this way a new industry would be enabled of leased usage of appliances where the leases might be analogous to car leasing.

With VDE installed, appliances could also be managed by secure identification (PIN, voice or signature recognition, etc.). This might be required each time a unit is used, or on some periodic basis. Failure to use the secure identification or use it on a timely basis could disable an appliance if a PPE 650 issued one or more instructions (or failed to decrypt or otherwise provide certain information critical to appliance operation) that prevented use of a portion or all of the appliance's functions. This feature would greatly reduce the desirability of stealing an electronic appliance. A further, allied use of VDE is the "registration" of a VDE secure subsystem in a given appliance with a VDE secure subsystem at some control location in a home or business. This control location might also be responsible for VDE remote communications and/or centralized administration (including, for example, restricting your children from viewing R

rated movies either on television or videocassettes through the recognition of data indicating that a given movie, song, channel, game, etc. was R rated and allowing a parent to restrict viewing or listening). Such a control location may, for example, also gather information on consumption of water, gas, electricity, telephone usage, etc. (either through use of PPEs 650 integrated in control means for measuring and/or controlling such consumption, or through one or more signals generated by non-VDE systems and delivered to a VDE secure subsystem, for example, for processing, usage control (e.g. usage limiting), and/or billing), transmit such information to one or more utilities, pay for such consumption using VDE secured electronic currency and/or credit, etc.

In addition, one or more budgets for usage could be managed by VDE which would prevent improper, excessive use of a certain, leased appliance, that might, for example lead to failure of the appliance, such as making far more copies using a photocopier than specified by the duty cycle. Such improper use could result in a message, for example on a display panel or television screen, or in the form of a communication from a central clearinghouse, that the user should upgrade to a more robust model.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

WE CLAIM:

1. A rights management appliance including:

a user input device,

a user display device,

at least one processor, and

at least one element defining a protected processing environment,

characterized in that the protected processing environment stores and uses permissions, methods, keys, programs and/or other information to electronically manage rights.

2. In a rights management appliance including:

a user input device,

a user display device,

at least one processor, and

at least one element defining a protected processing environment,

a method of operating the appliance characterized by the step of storing and using permissions, methods, keys, programs and/or other information to electronically manage rights.

3. A rights management appliance including at least one processor element at least in part defining a protected processing

environment, characterized in that the protected processing environment stores and uses permissions, methods, keys, programs and/or other information to electronically manage rights.

4. In a rights management appliance including at least one processor element at least in part defining a protected processing environment, a method comprising storing and using permissions, methods, keys, programs and/or other information to electronically manage rights.

5. An electronic appliance arrangement containing at least one secure processing unit and at least one secure database operatively connected to at least one of said secure processing unit(s), said arrangement including means to monitor usage of at least one aspect of appliance usage and control said usage based at least in part upon protected appliance usage control information.

6. In an electronic appliance arrangement containing at least one secure processing unit and at least one secure database operatively connected to at least one of said secure processing unit(s), a method characterized by the steps of monitoring usage of at least one aspect of appliance usage and controlling said

usage based at least in part upon protected appliance usage control information.

7. An electronic appliance arrangement containing a protected processing environment and at least one secure database operatively connected to said protected processing environment, said arrangement including means to monitor usage of at least one aspect of an amount of appliance usage and control said usage based at least in part upon protected appliance usage control information processed at least in part through use of said protected processing environment.

8. In an electronic appliance arrangement containing a protected processing environment and at least one secure database operatively connected to said protected processing environment, a method characterized by the steps of monitoring usage of at least one aspect of appliance usage and controlling said usage based at least in part upon protected appliance usage control information processed at least in part through use of said protected processing environment.

9. An electronic appliance arrangement containing one or more CPUs wherein at least one of the CPUs incorporates an integrated secure processing unit, said arrangement storing

protected appliance usage control information designed to be securely processed by said integrated secure processing unit.

10. In an electronic appliance arrangement containing one or more CPUs wherein at least one of the CPUs incorporates an integrated secure processing unit, a method including the step of storing and securely processing protected modular component appliance usage control information with said integrated secure processing unit.

11. A method of compromising a distributed electronic rights management system comprising plural nodes having protected processing environments, characterized by the following steps:

(a) exposing a certification private key,

(b) passing at least one challenge/response protocol and/or exposing at least one external communication key based at least in part on the key exposed by the exposing step,

(c) creating a processing environment based at least in part on steps (a) and (b), and

participating in distributed rights management using the processing environment created by step (c).

12. A processing environment for compromising a distributed electronic rights management system comprising plural nodes having protected processing environments, characterized by the following:

protocol passing means including an exposed certification private key for passing at least one challenge/response protocol,

means coupled to the protocol passing means for at least one of (a) defeating an initialization challenge/response security, and/or (b) exposing external communication keys, and

means coupled to the security detecting means for participating in distributed rights management.

13. A method of compromising a distributed electronic rights management system comprising plural nodes having associated protected processing environments, characterized by the steps of:

compromising the permissions record of an electronic container, and

using the compromised permissions record to access and/or use electronic information.

14. A system for compromising a distributed electronic rights management system comprising plural nodes having associated protected processing environments, characterized by means for

using a compromised permissions record of an electronic container for accessing and/or using electronic information.

15. A method of tampering with a protected processing environment characterized by the steps of:

discovering at least one system-wide key, and

using the key to obtain access to content and/or administrative information without authorization.

16. An arrangement including means for using at least one compromised system-wide key to decrypt and compromise content and/or administrative information of a protected processing environment without authorization.

17. A combination general and secure processing computation element comprising:

a central processing unit;

at least one secure resource; and

a secure mode interface switch coupled between a centrla processing unit and the secure resource, the switch operable alternately in a secure mode and in a non secure mode, the switch blocking access by a central processing unit to the secure resource except when the switch is operating in the secure mode.

18. A secure printing method comprising:

downloading a decryption program to an intelligent printer;

sending an encrypted print stream to the printer;

decrypting the encrypted print stream within the printer using the decryption program; and

destroying the downloaded decryption program.

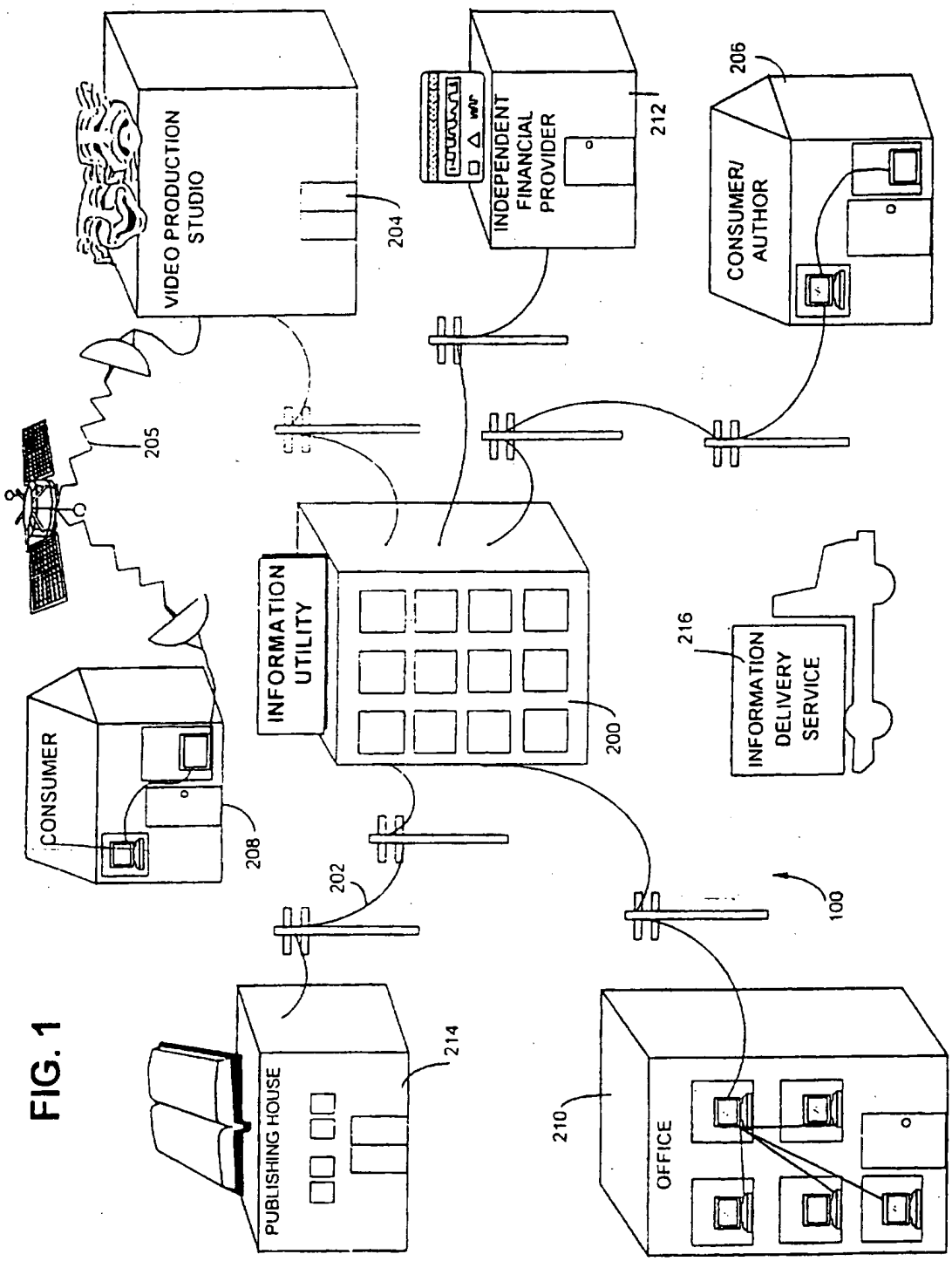


FIG. 1

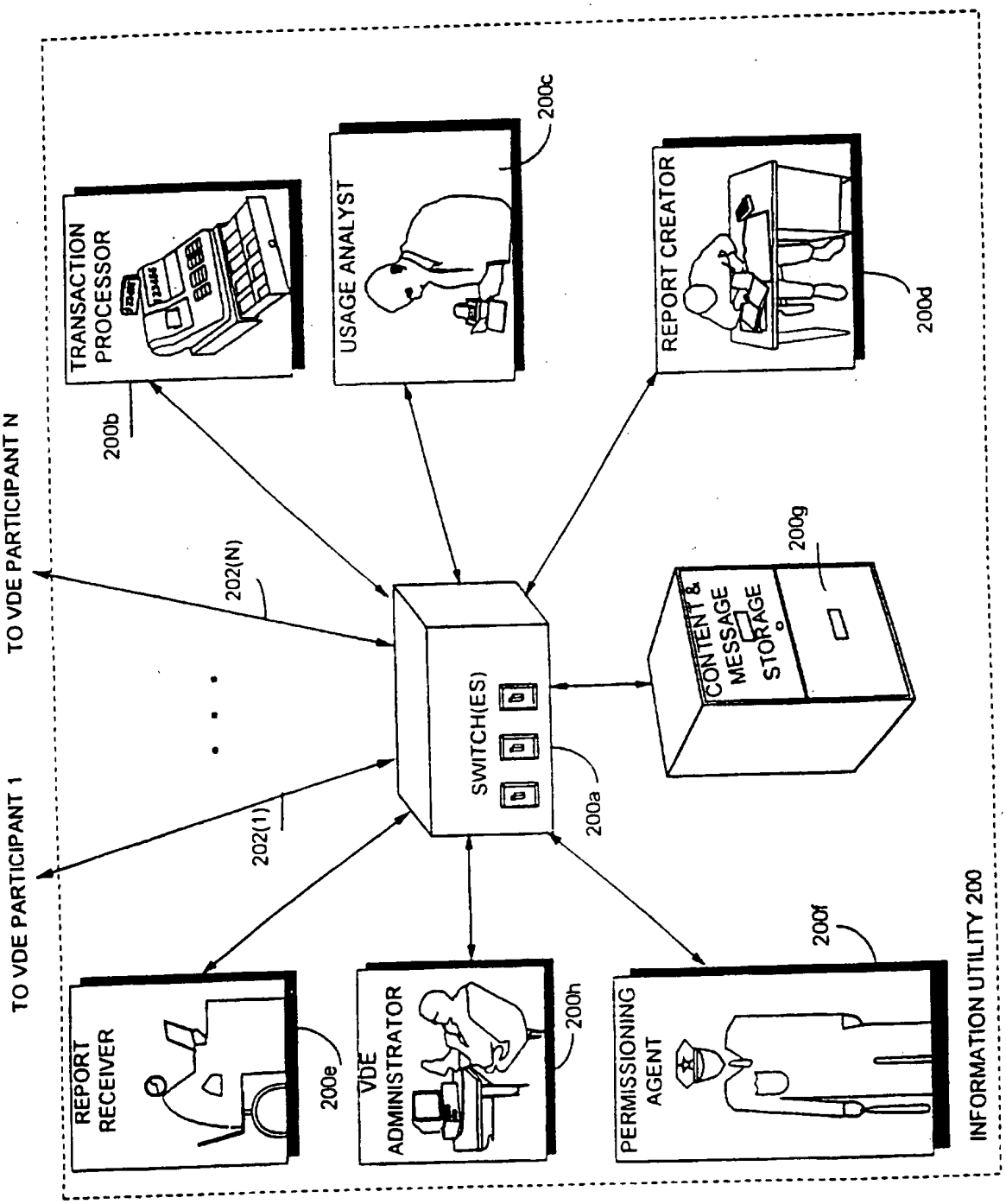


FIG. 1A

FIG. 2

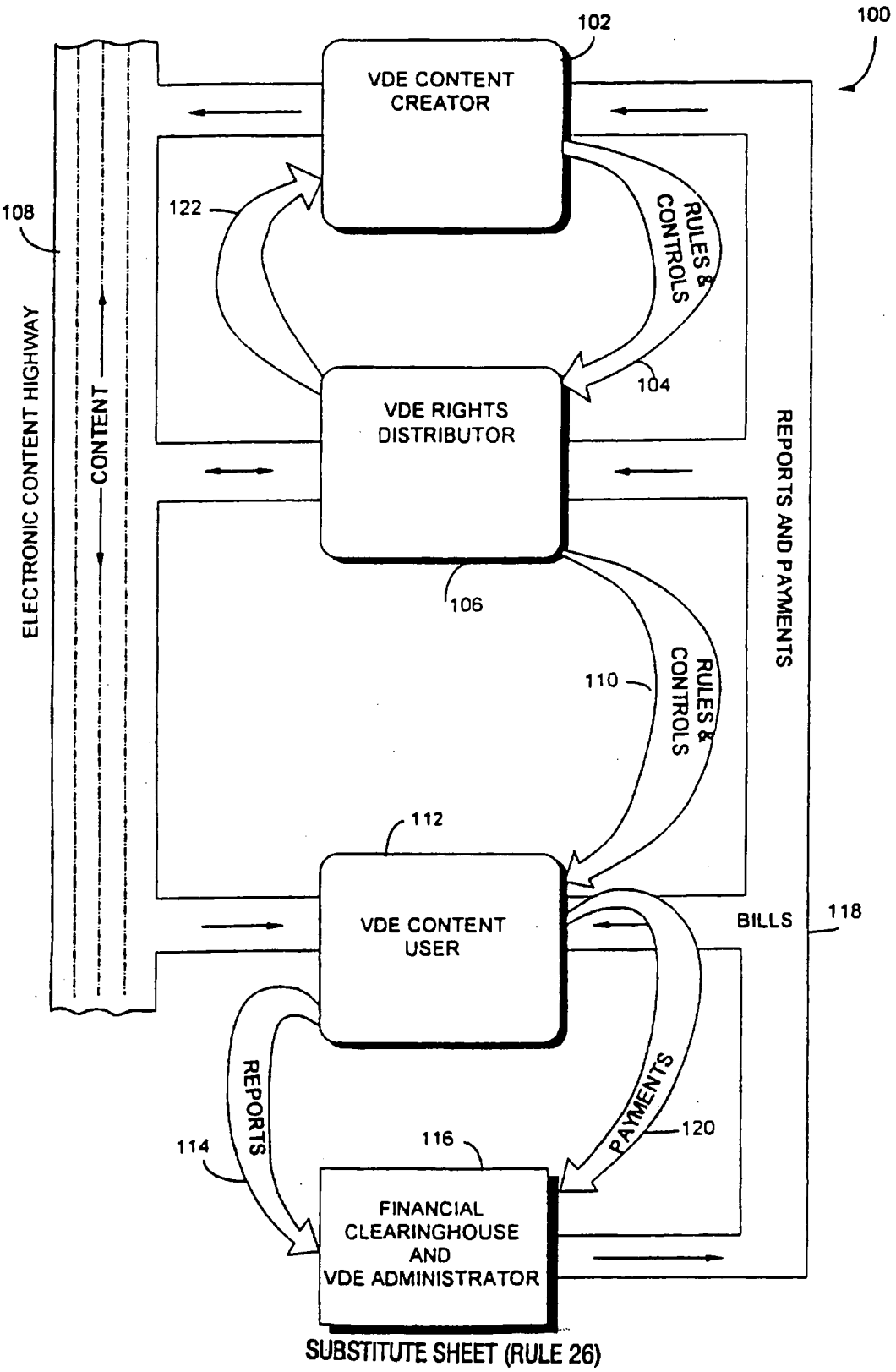


FIG. 2A

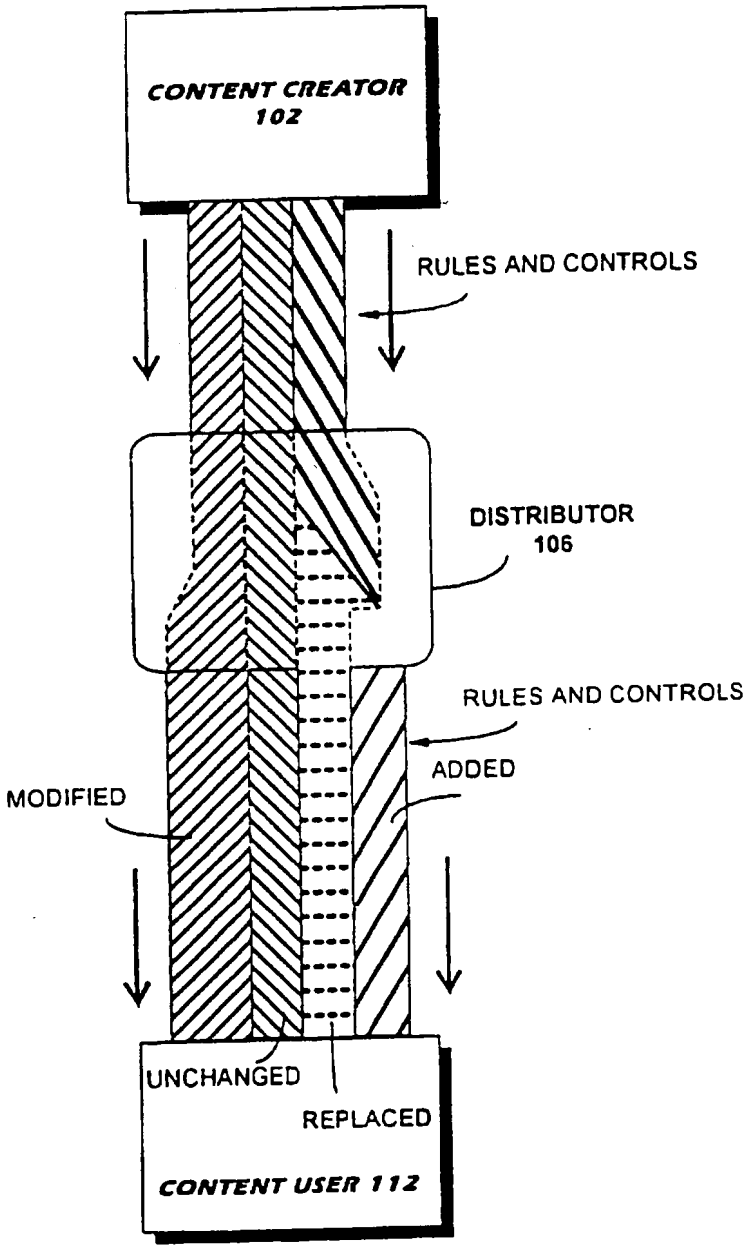
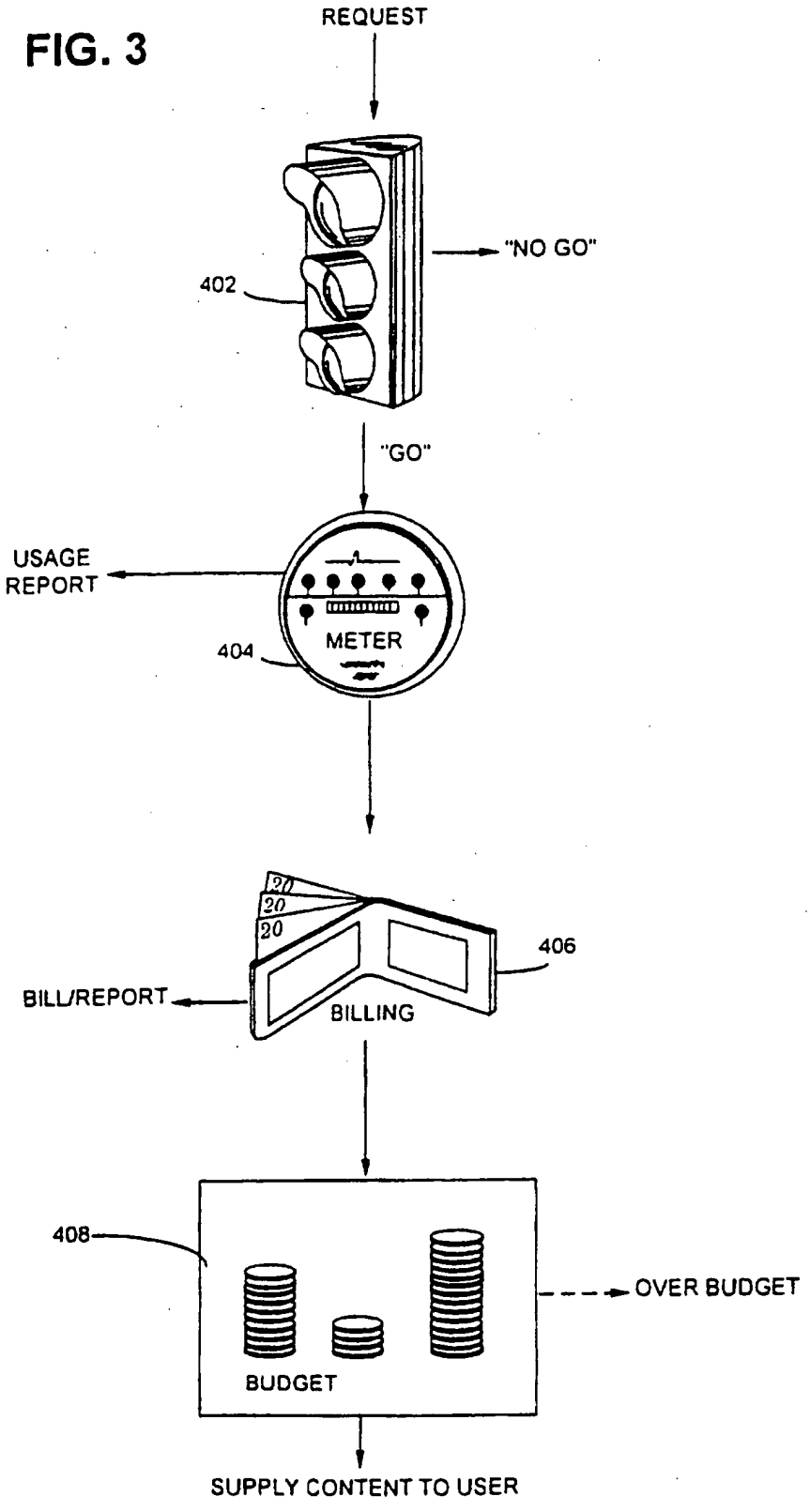


FIG. 3



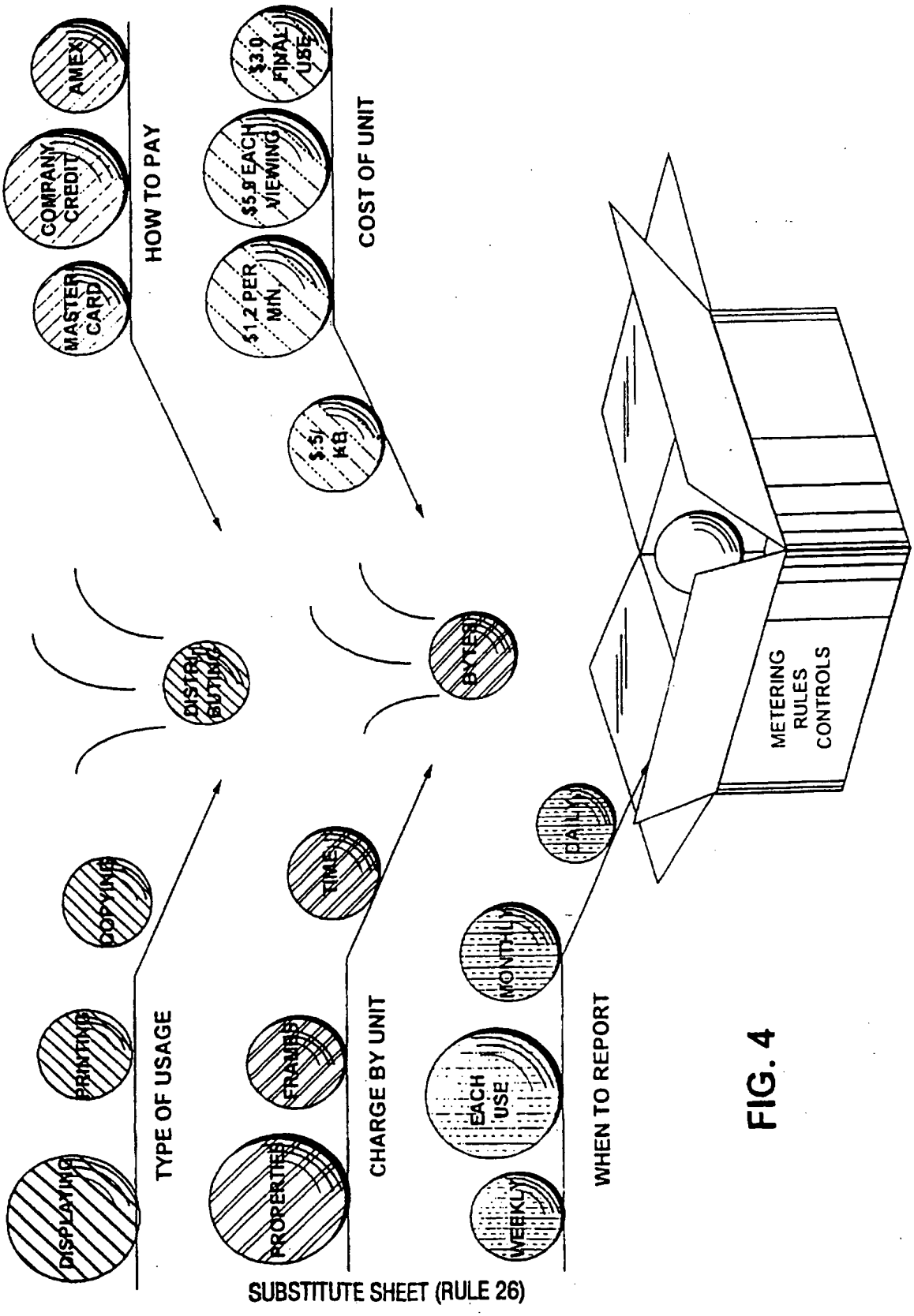
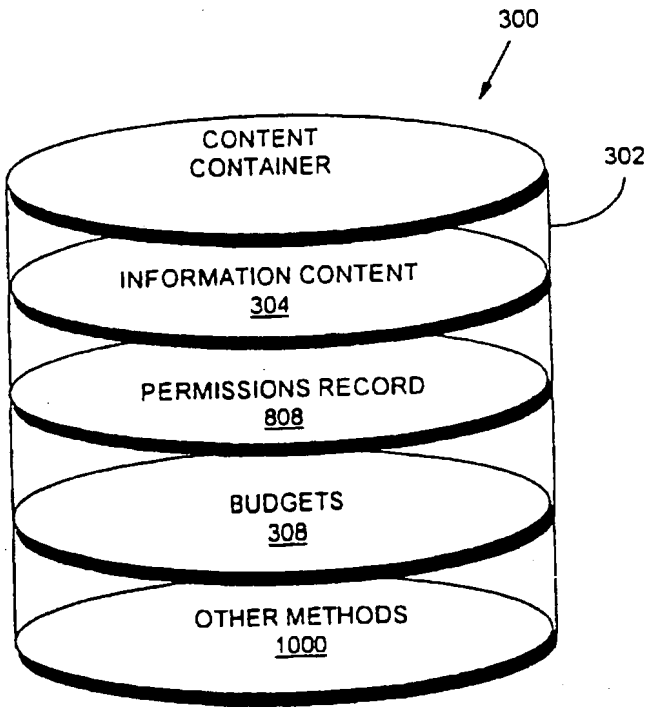


FIG. 4

FIG. 5A



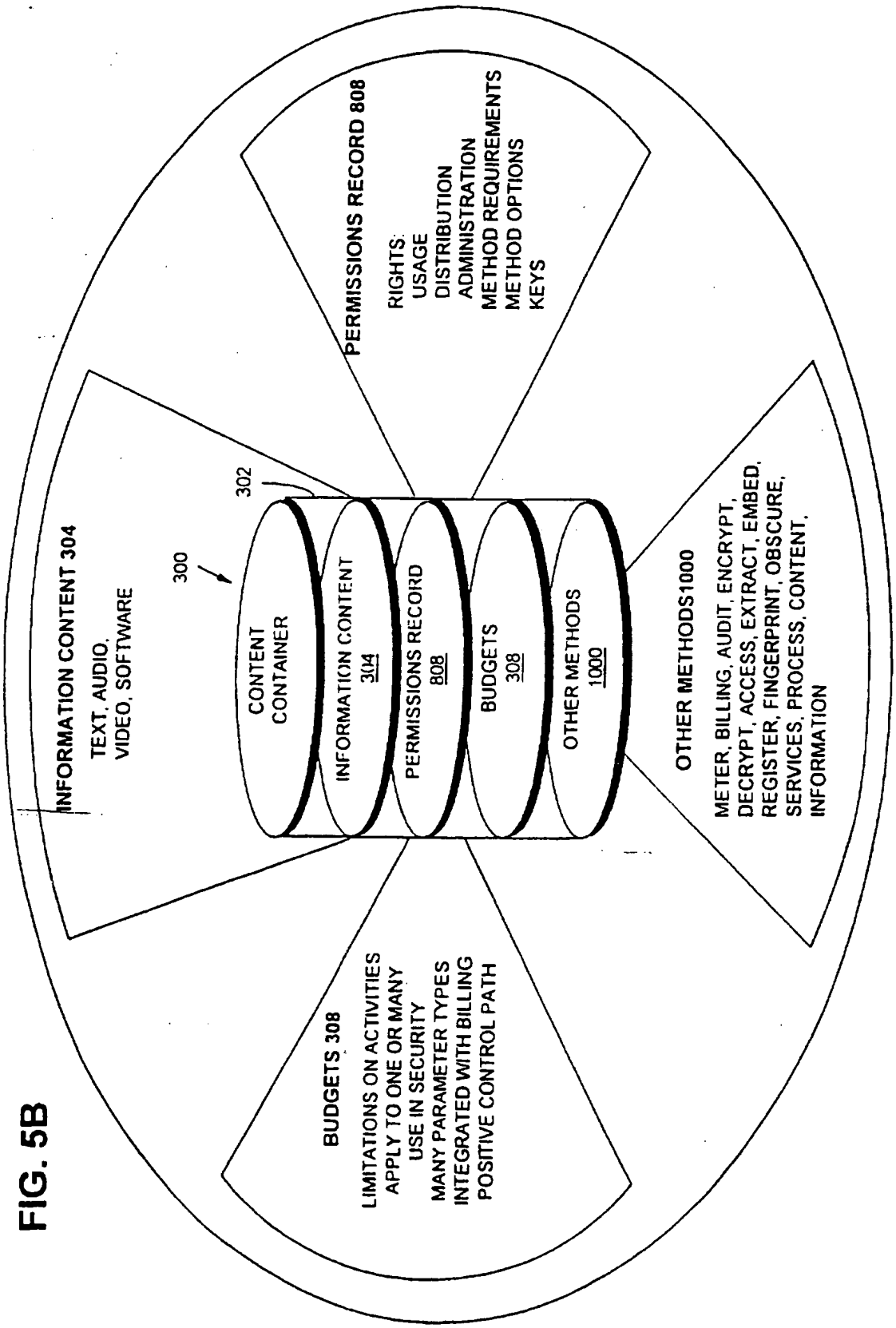
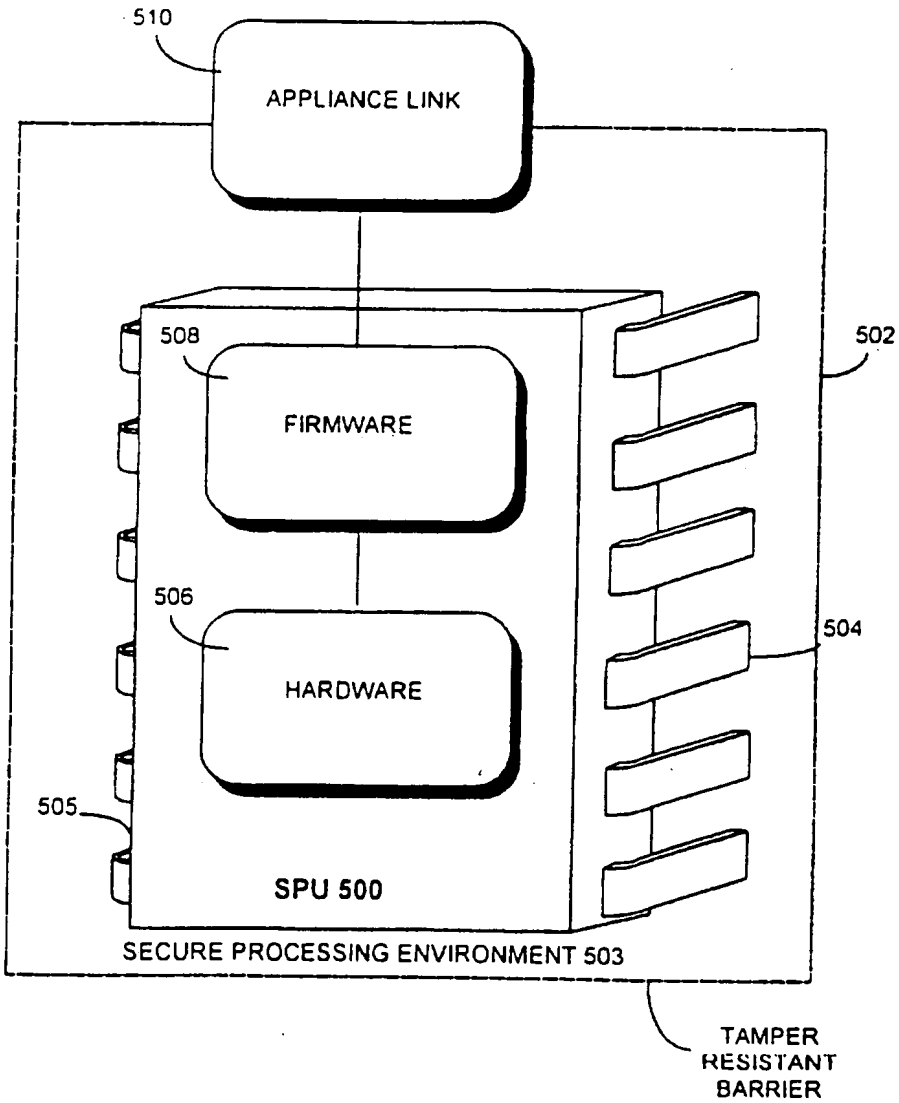


FIG. 5B

FIG. 6



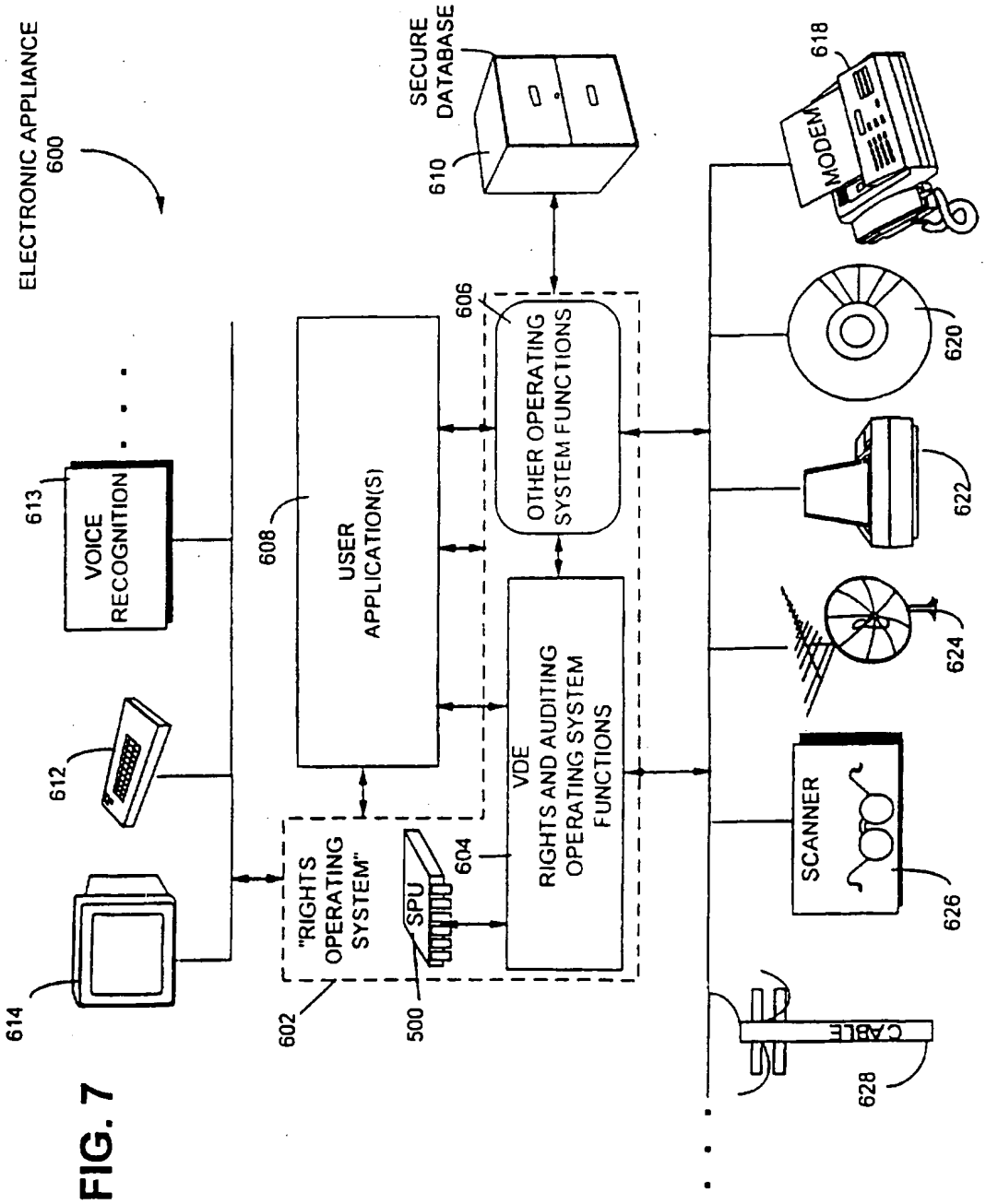
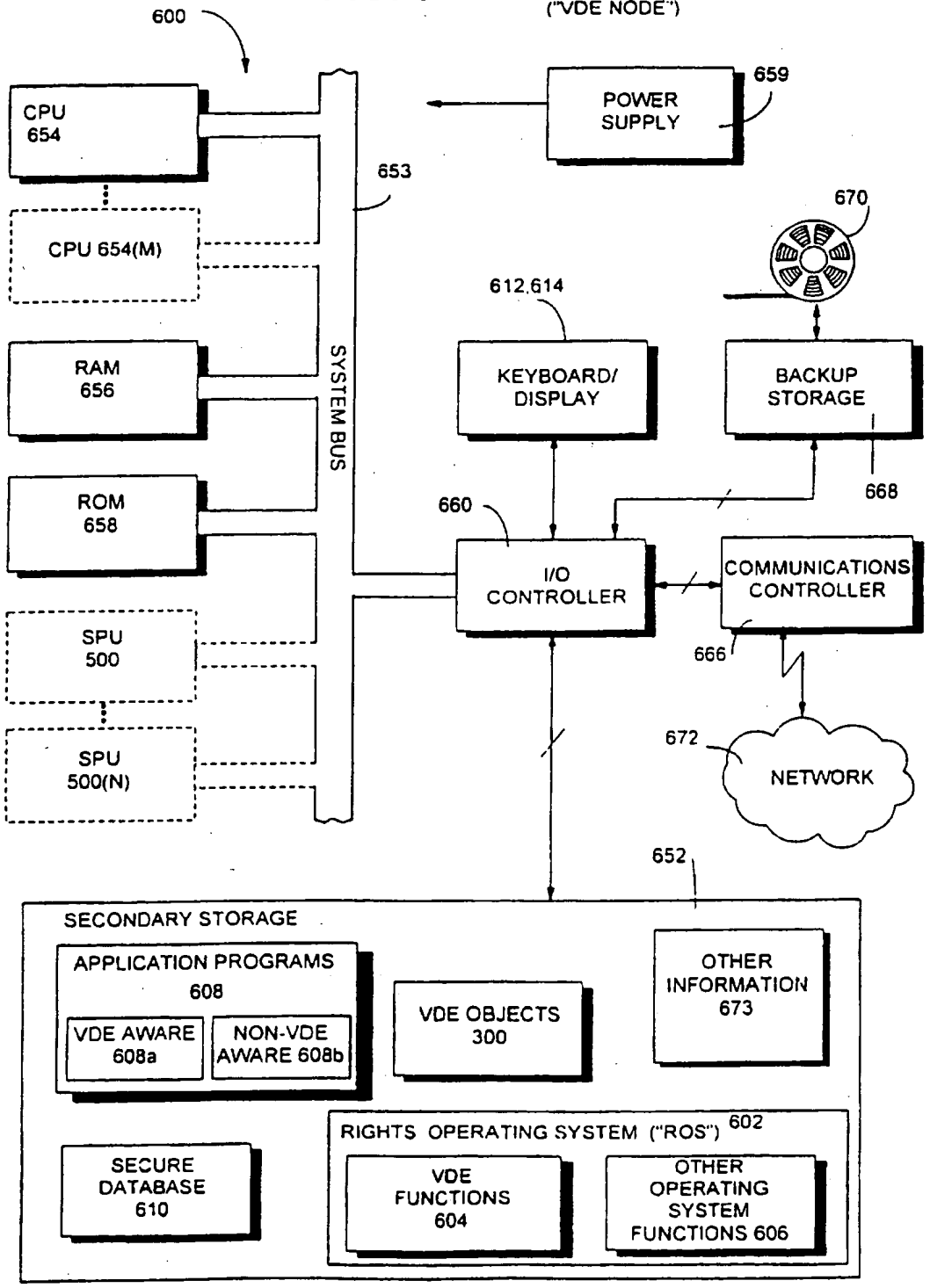


FIG. 7

FIG. 8 ELECTRONIC APPLIANCE 600 ("VDE NODE")



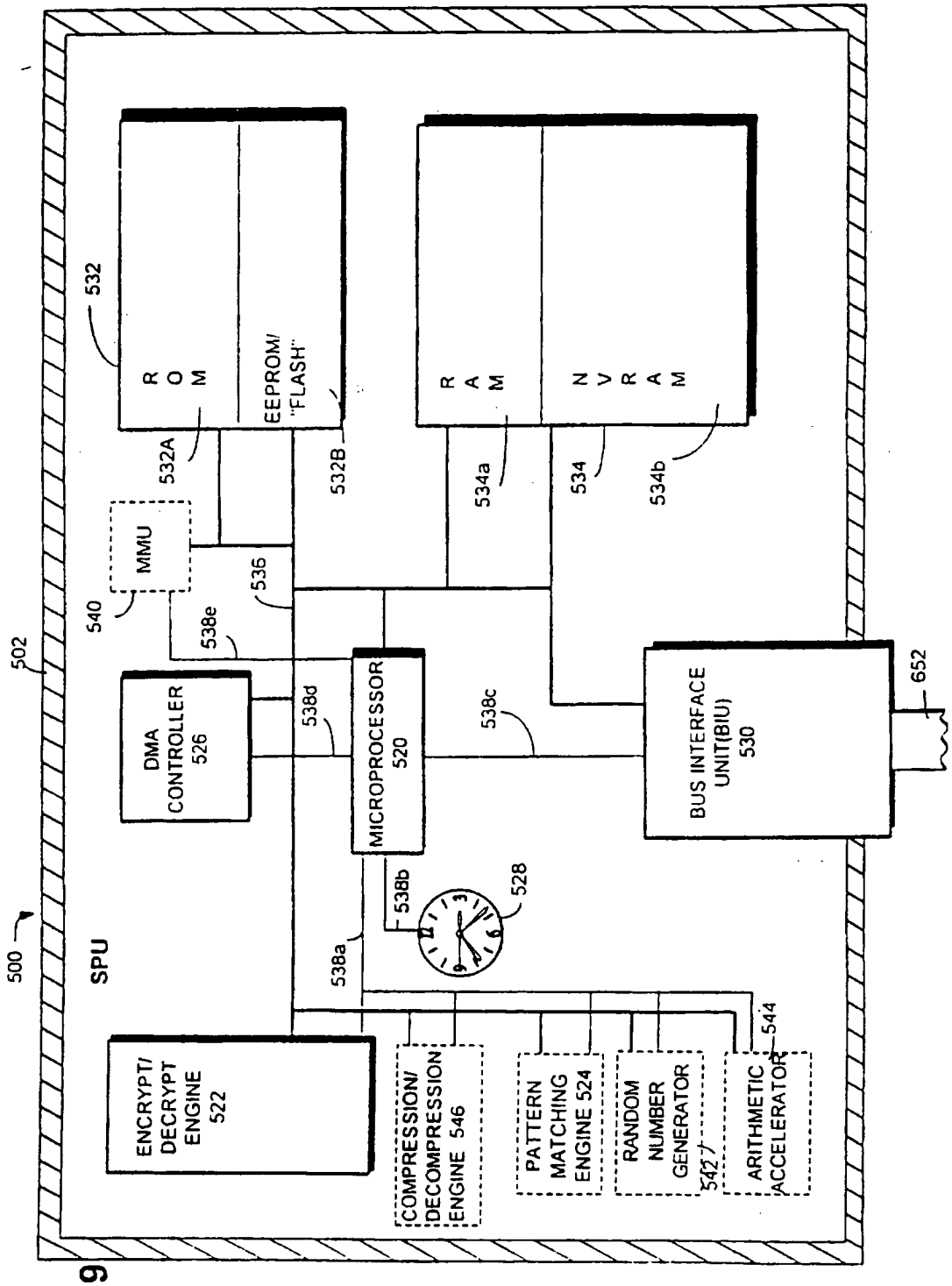


FIG. 9

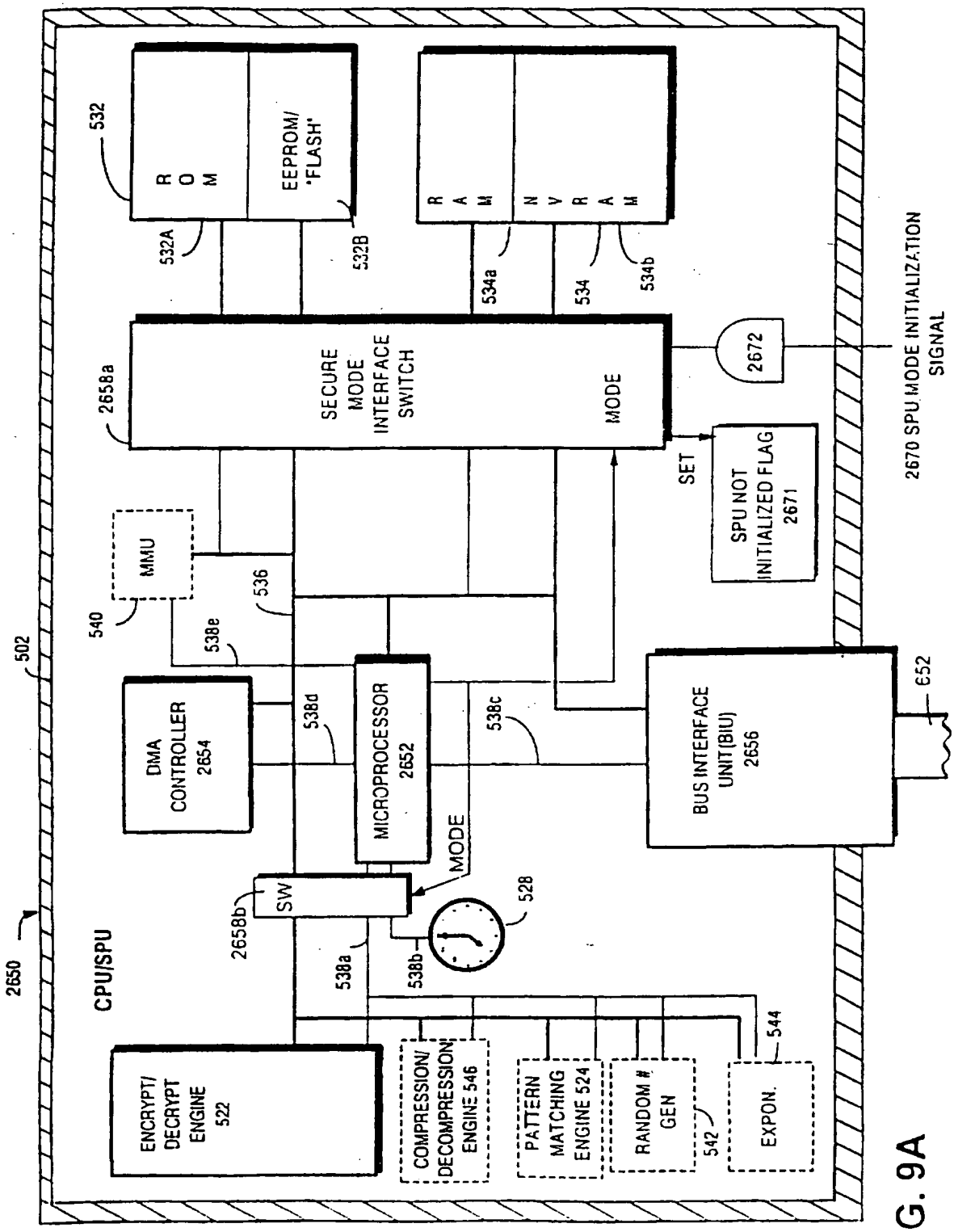


FIG. 9A

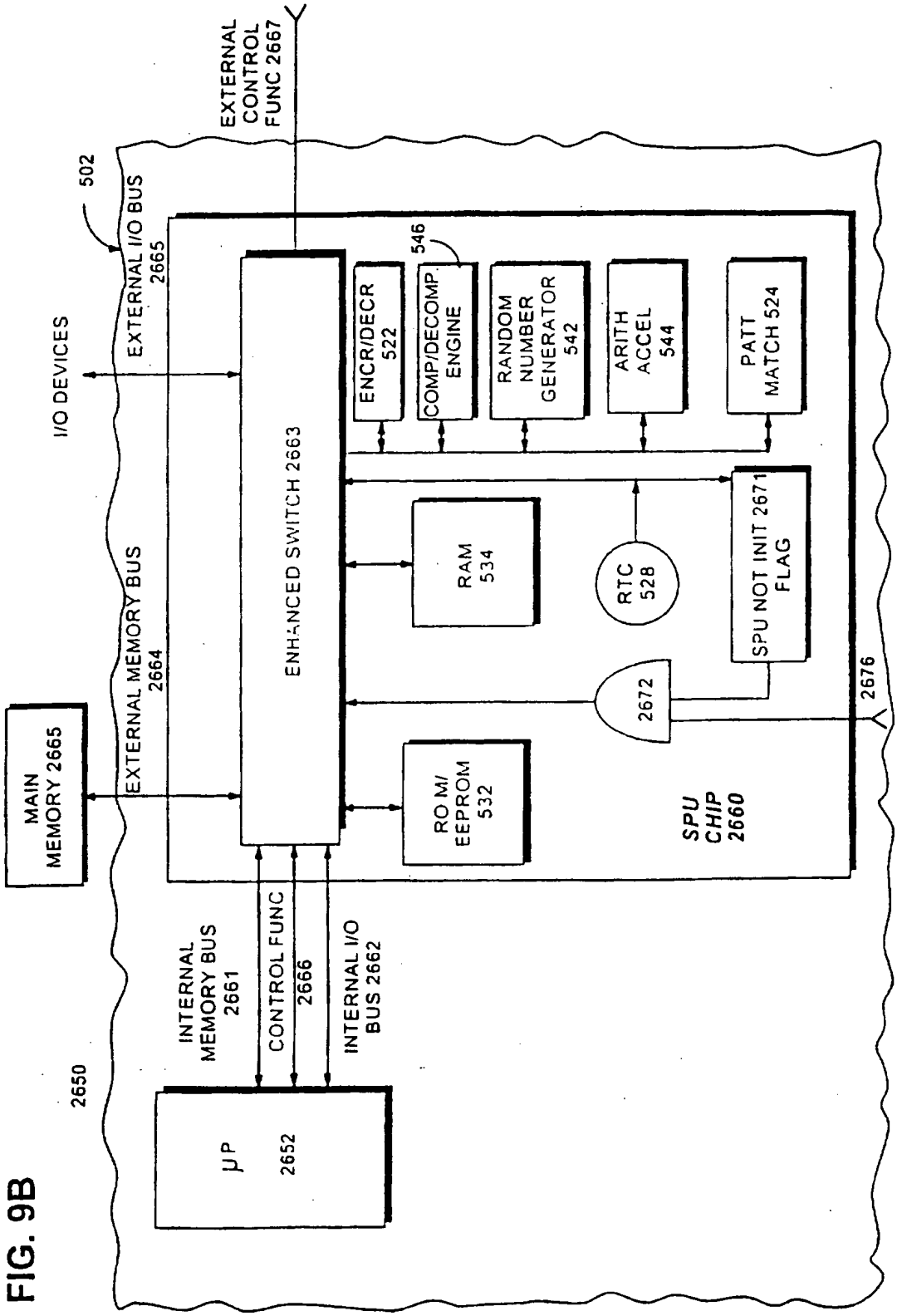


FIG. 9B

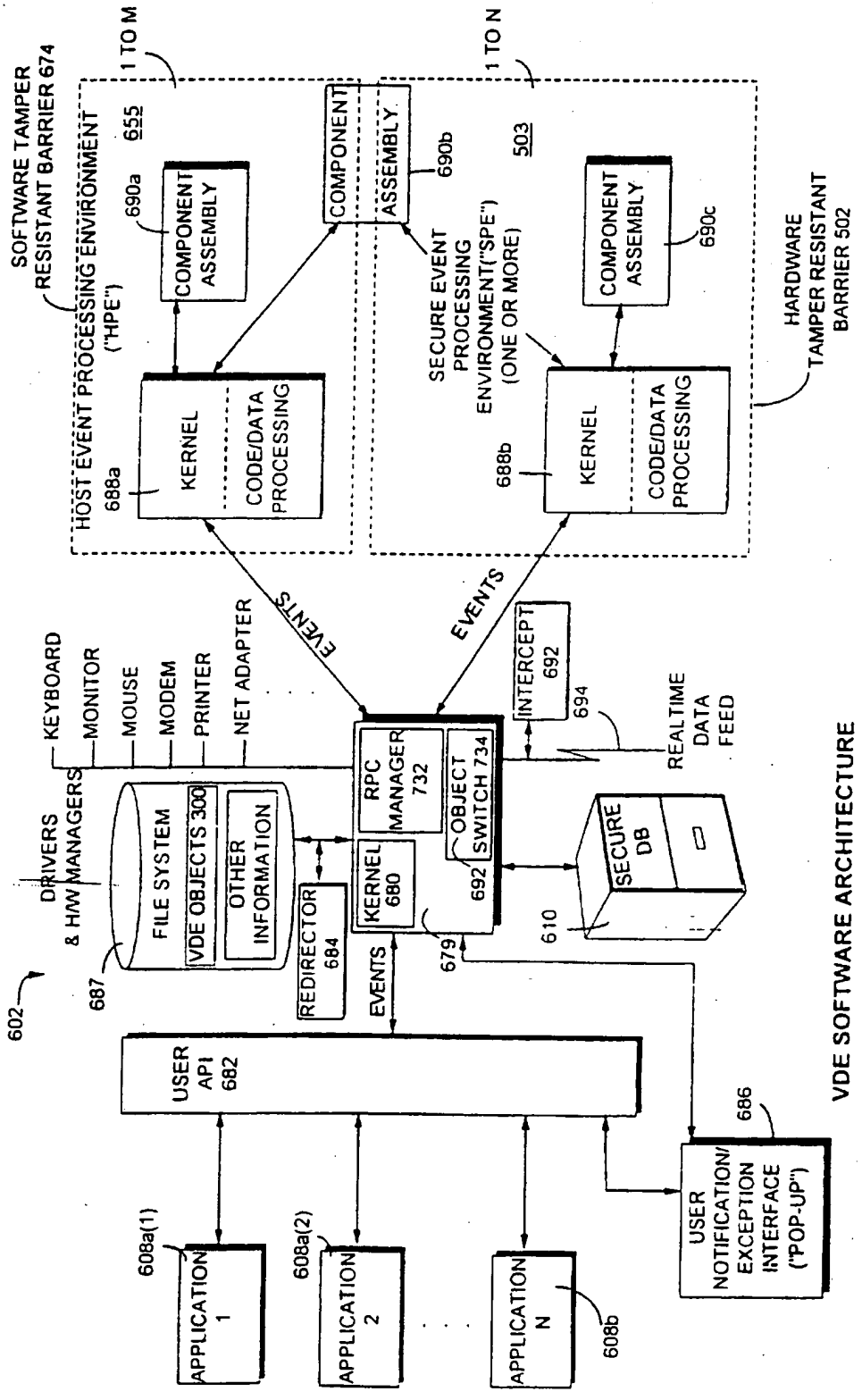


FIG. 10

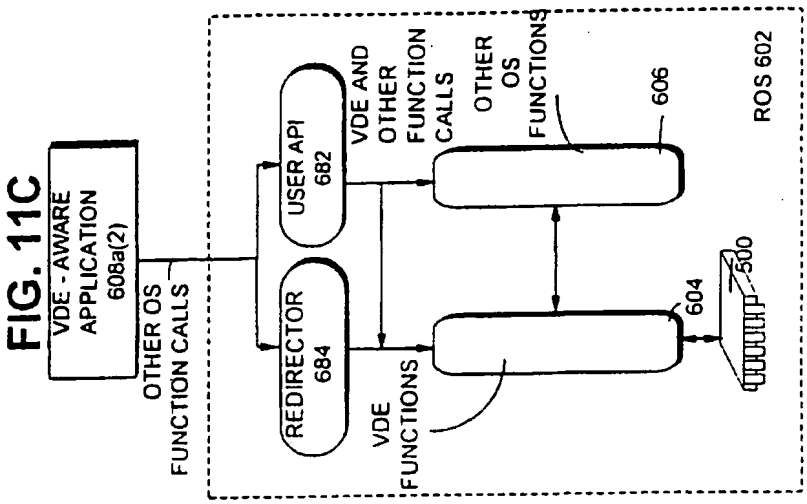


FIG. 11C

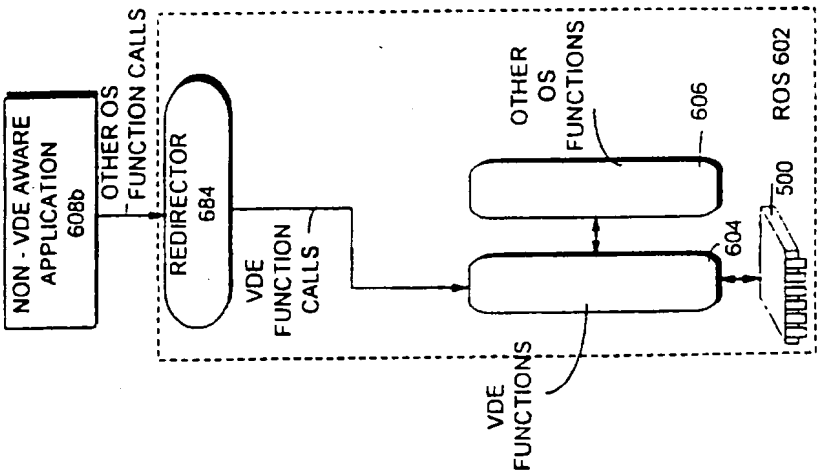


FIG. 11B

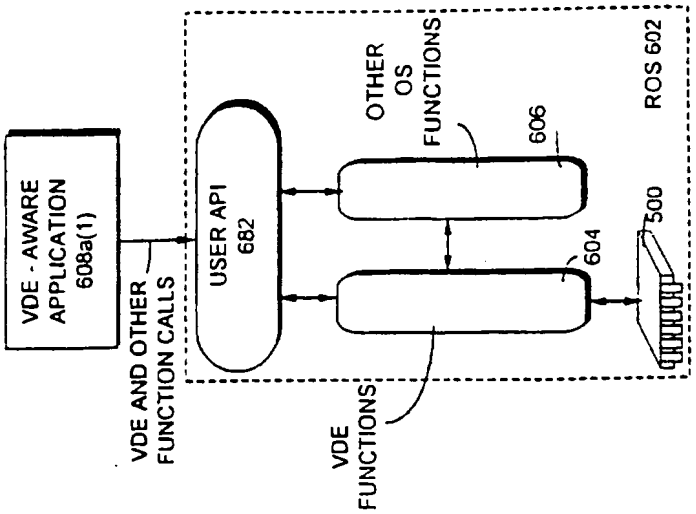


FIG. 11A

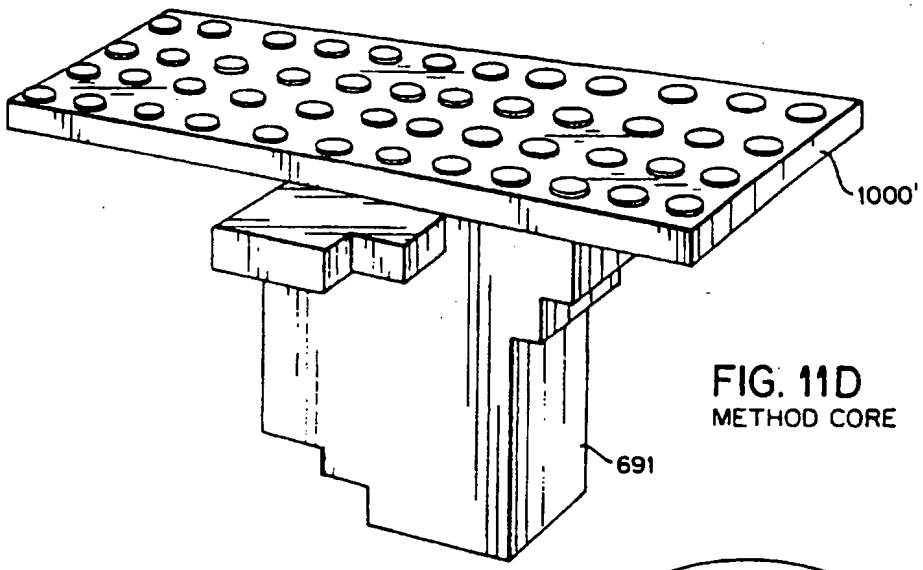


FIG. 11D
METHOD CORE

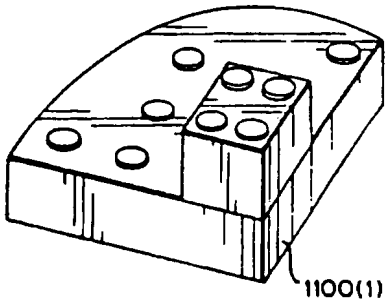


FIG. 11E
LOAD MODULE
WITH DTD

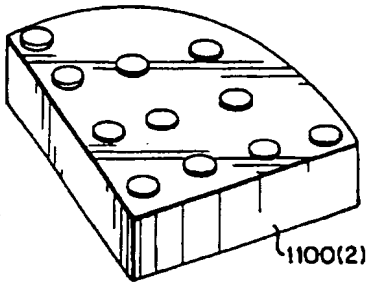


FIG. 11F
LOAD MODULE

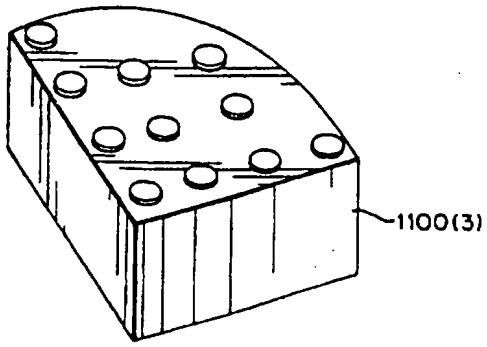


FIG. 11G
LOAD MODULE

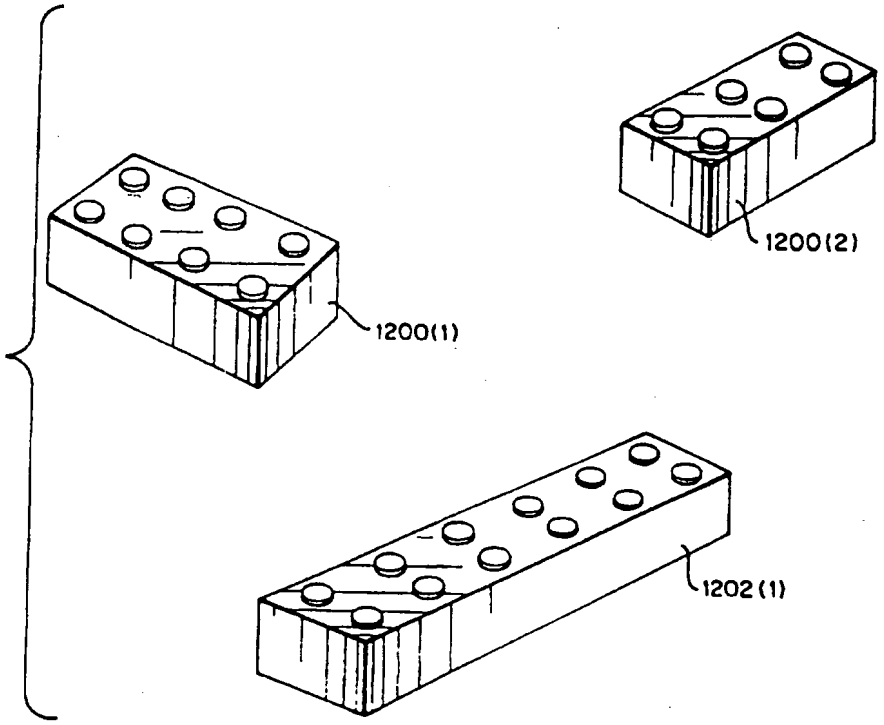
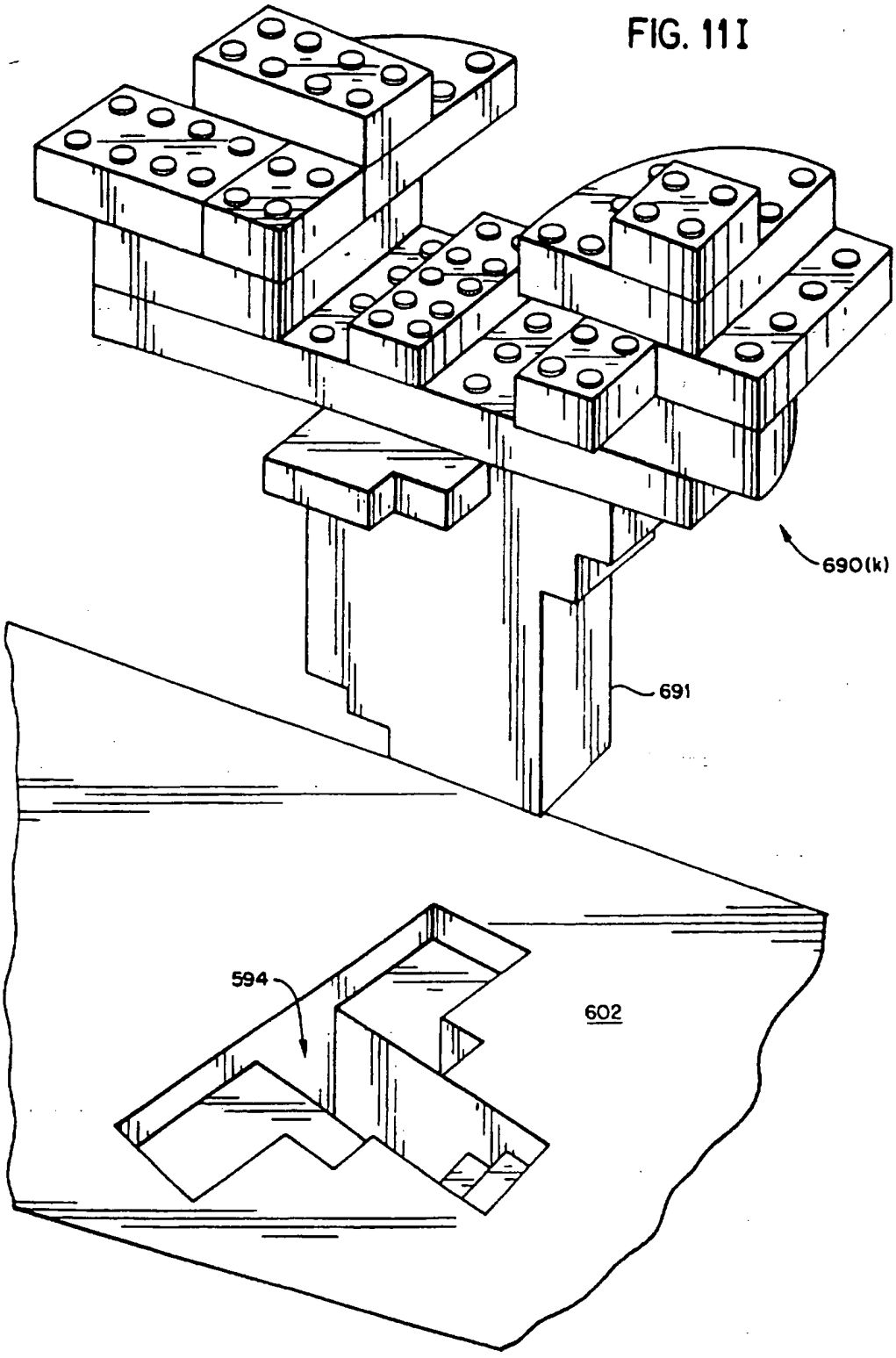


FIG. 11H
DATA STRUCTURES

FIG. 11I



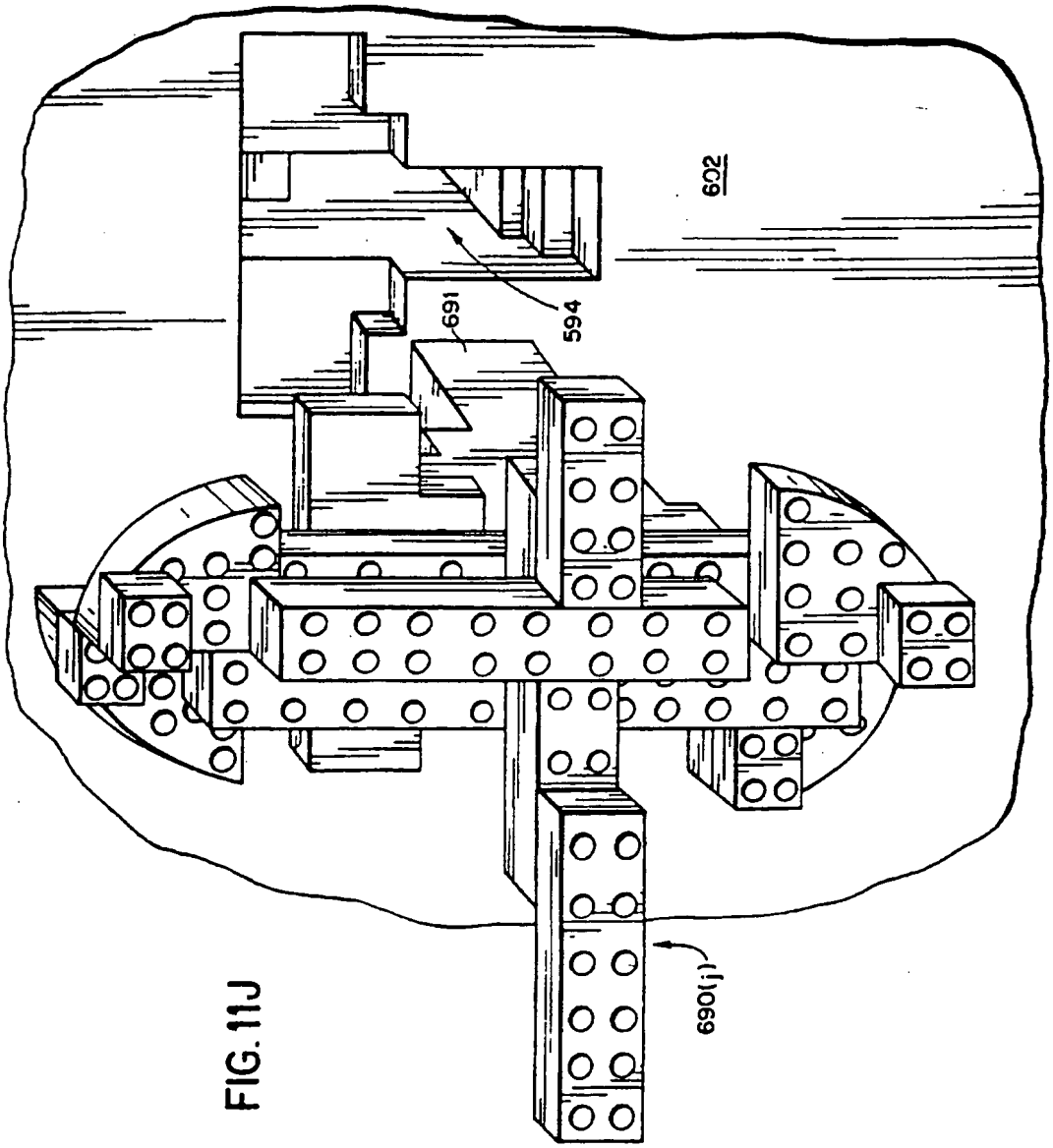


FIG. 11J

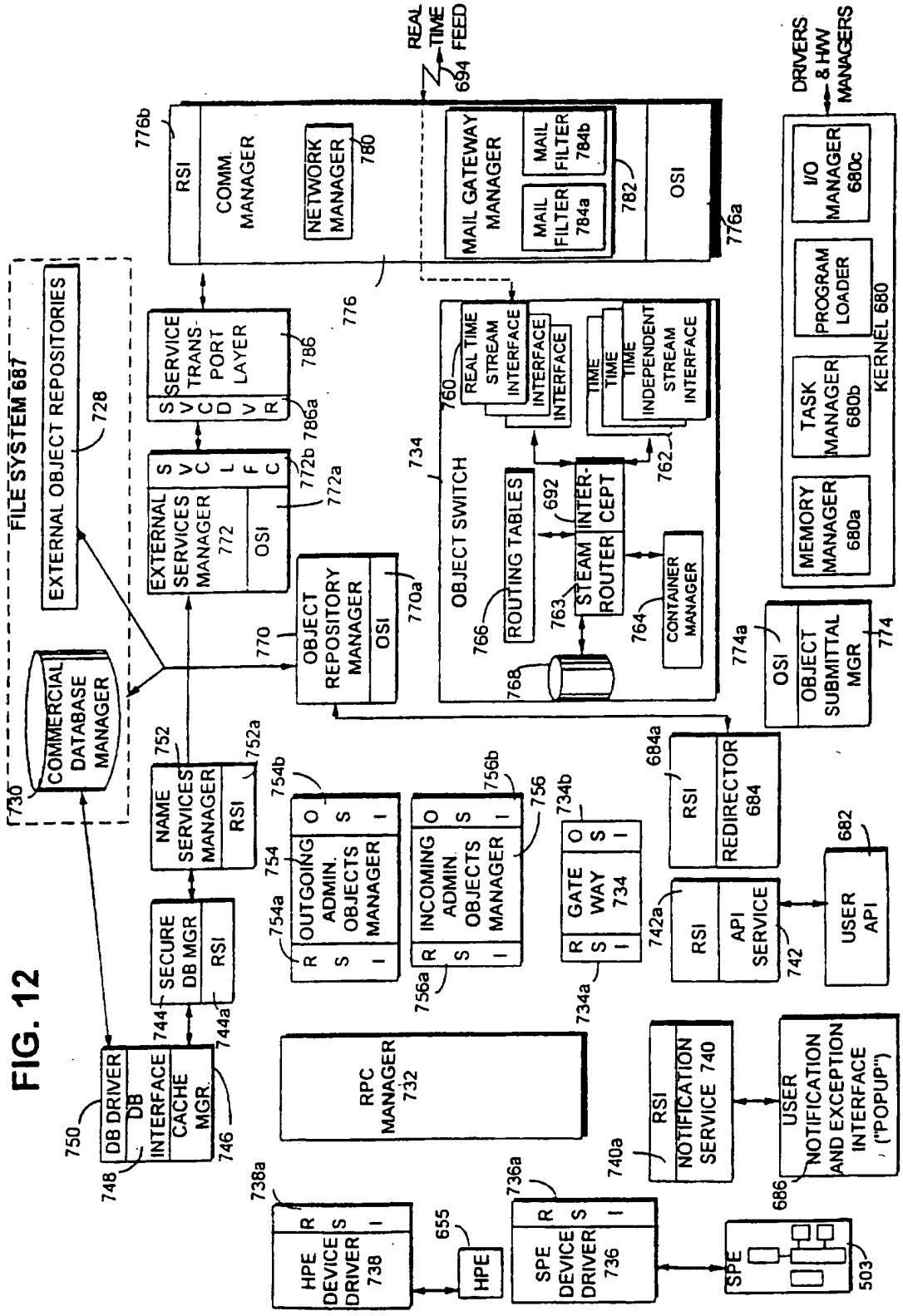


FIG. 12

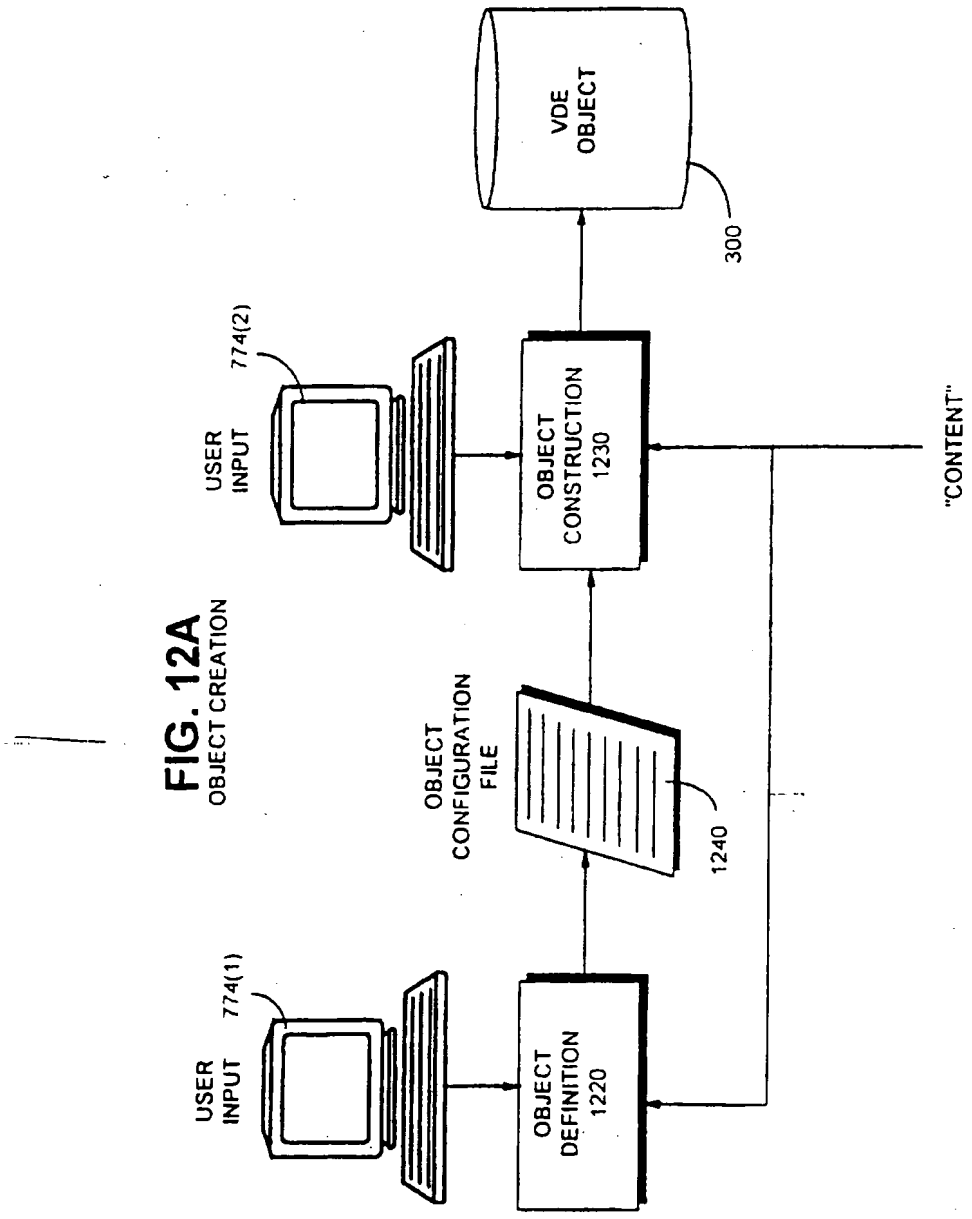


FIG. 12A
OBJECT CREATION

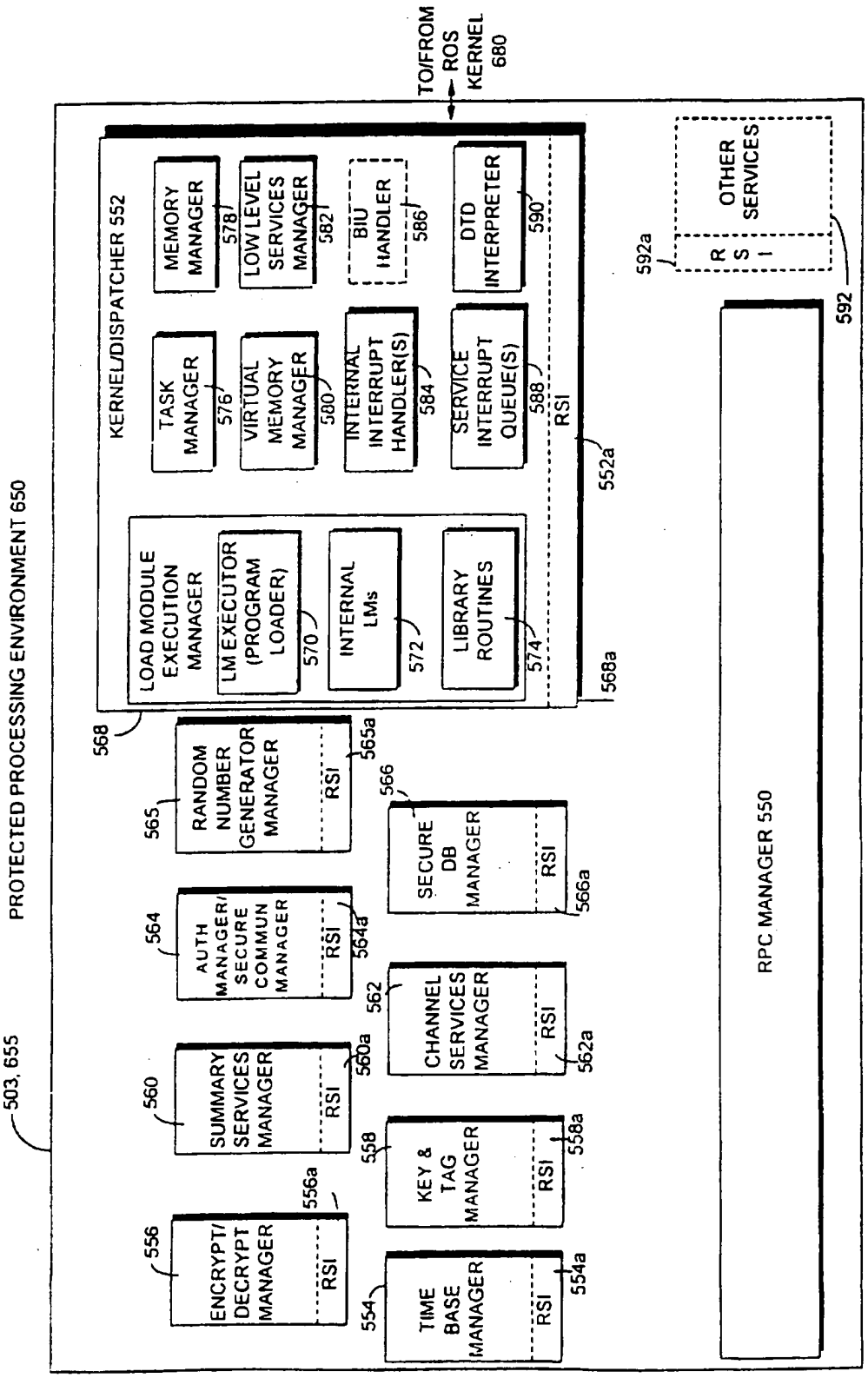


FIG. 13

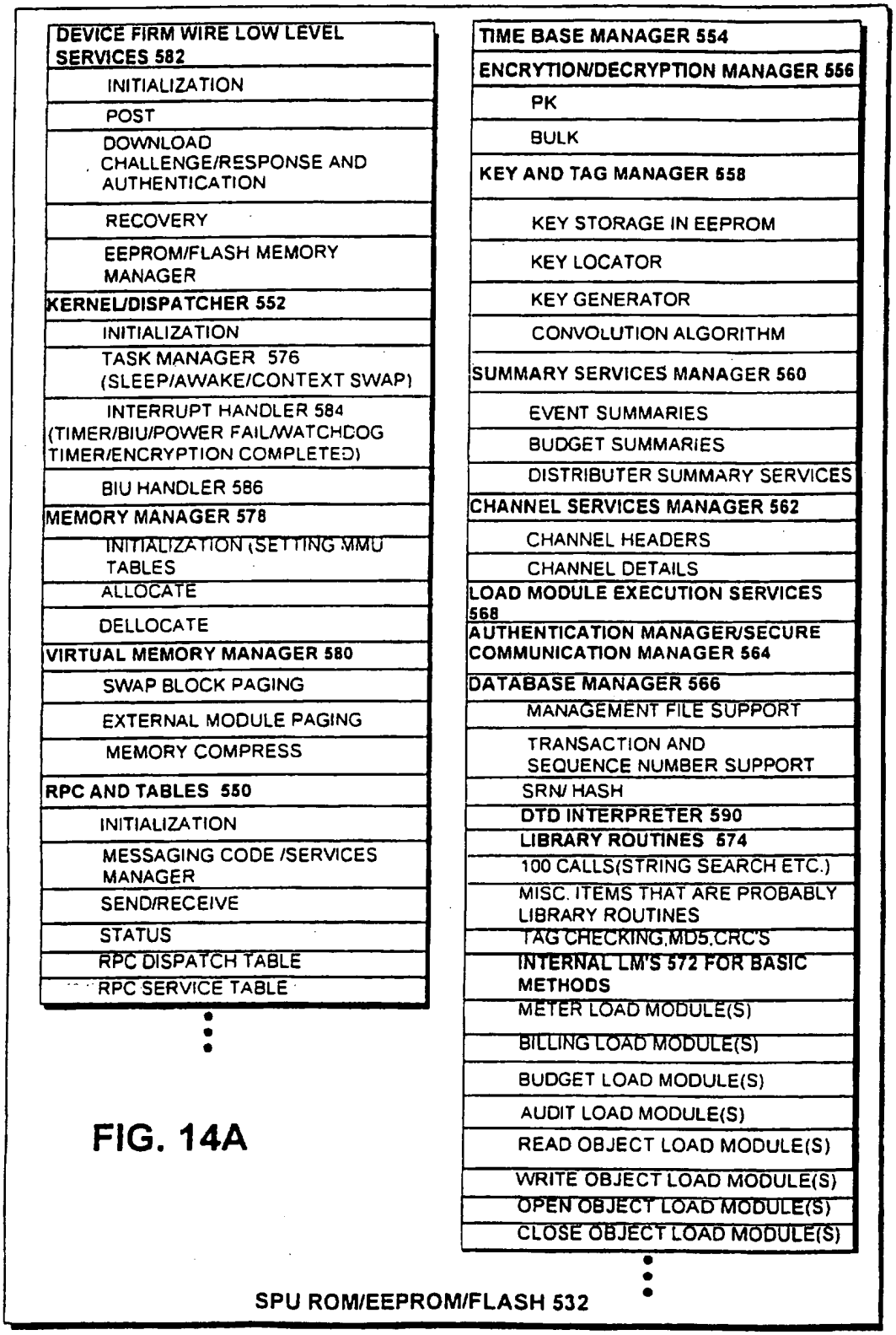


FIG. 14A

FIG. 14B

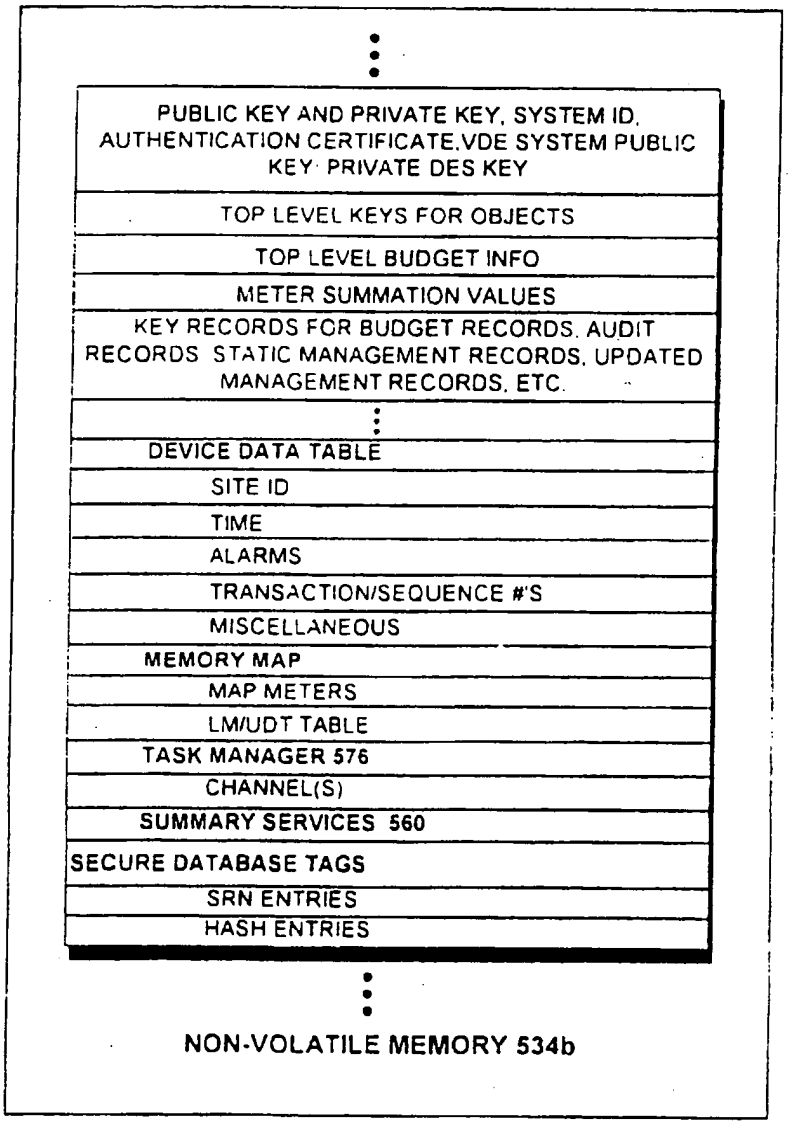


FIG. 14C

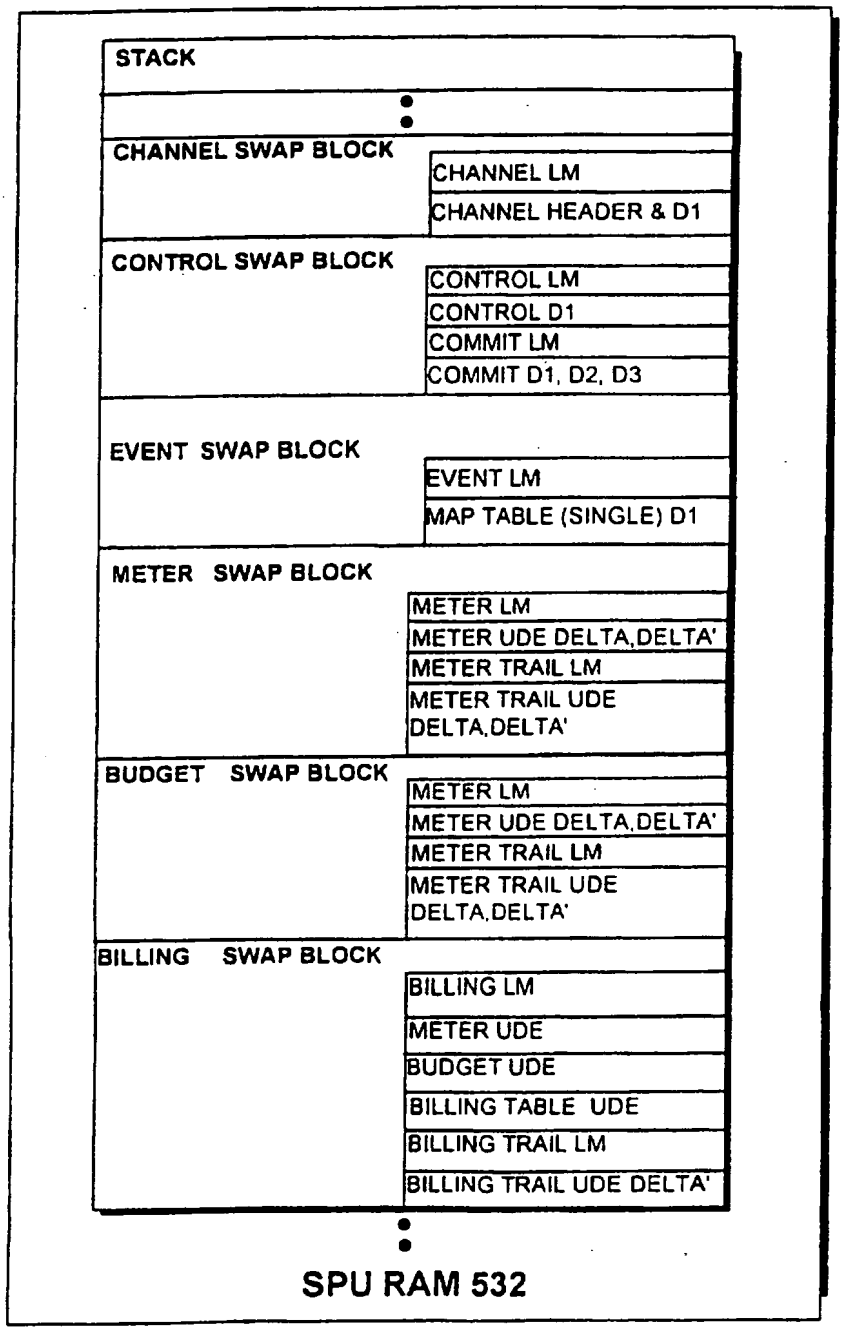


FIG. 15

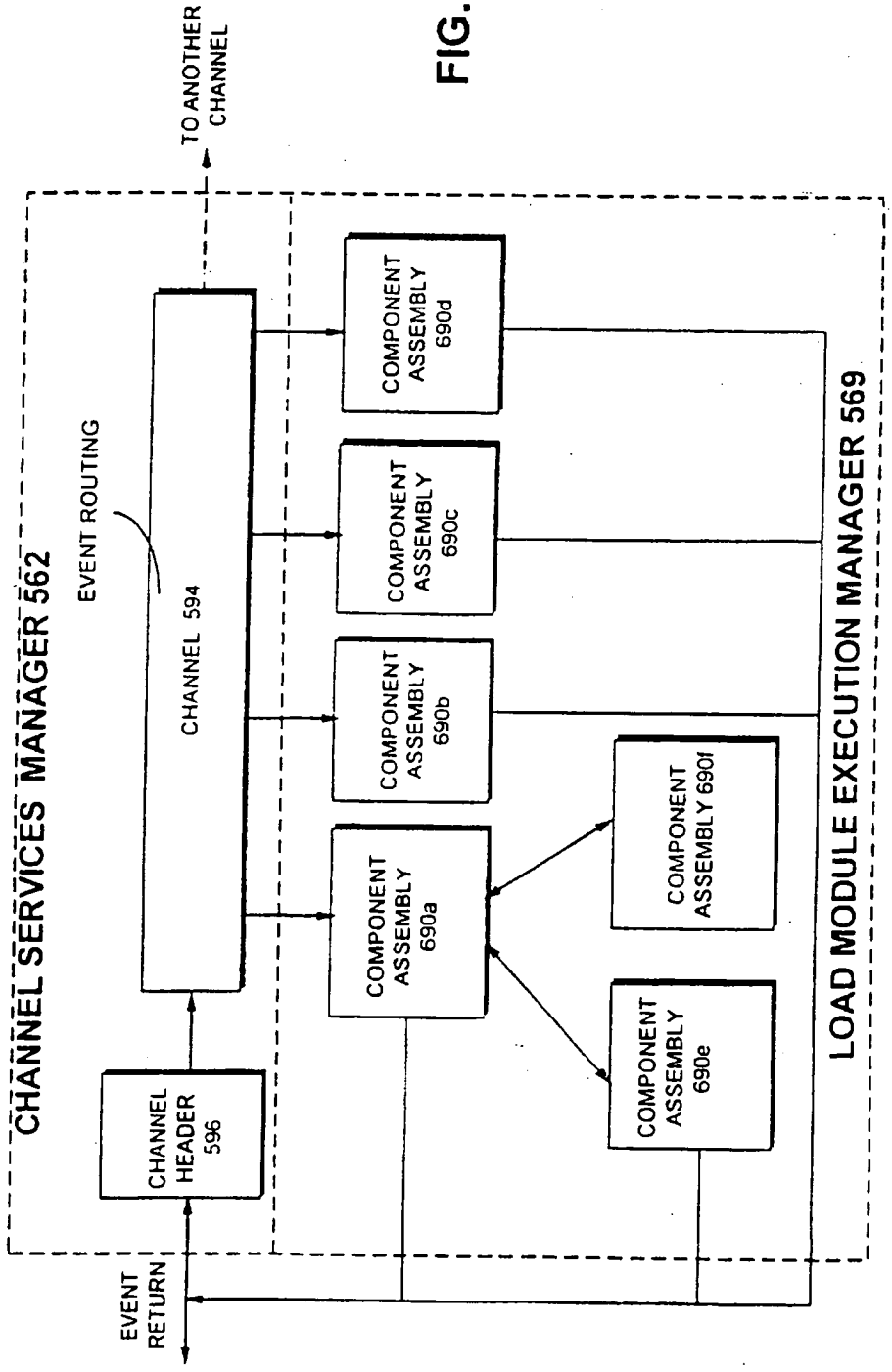


FIG. 15A

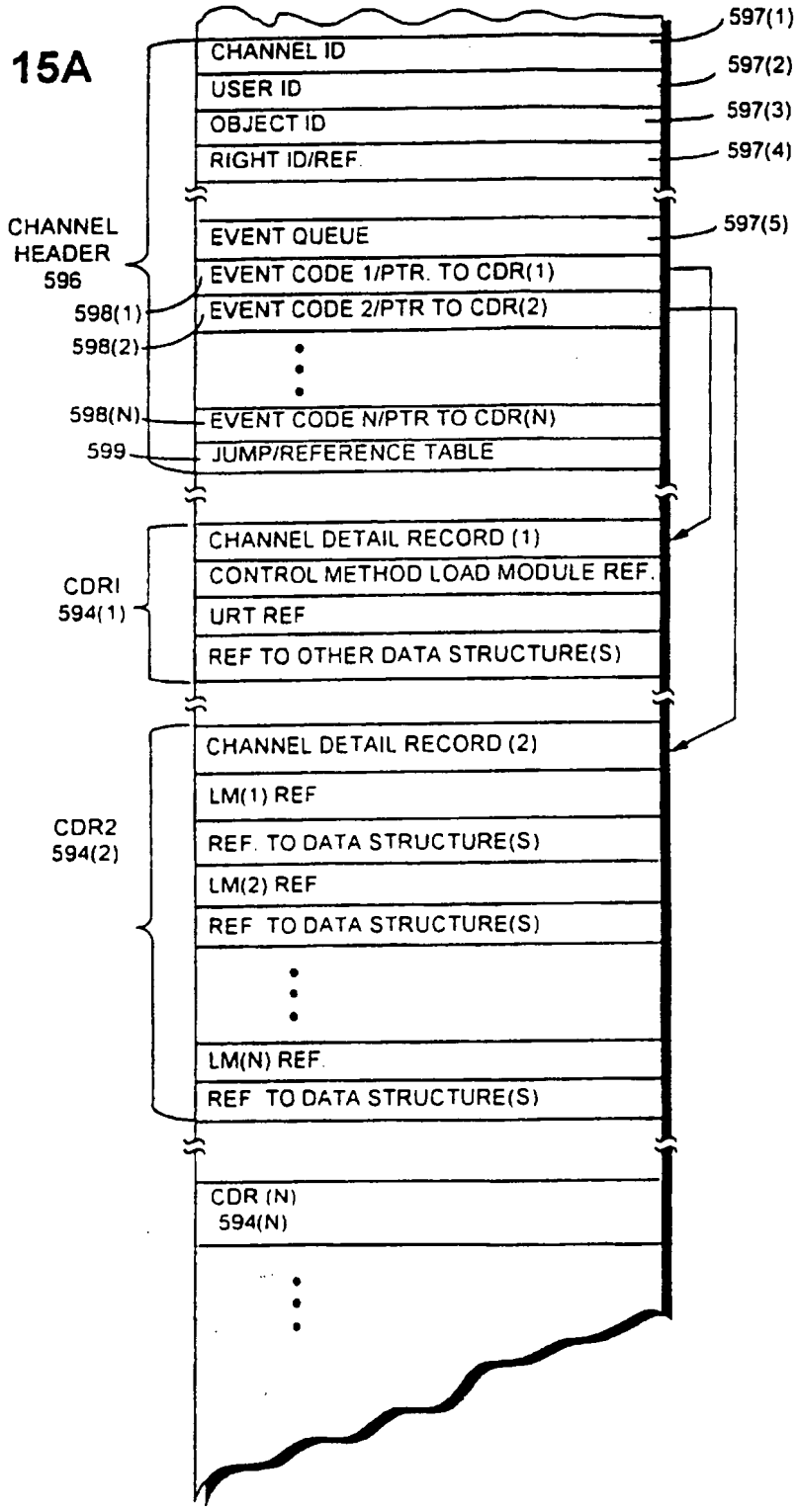


FIG. 15B

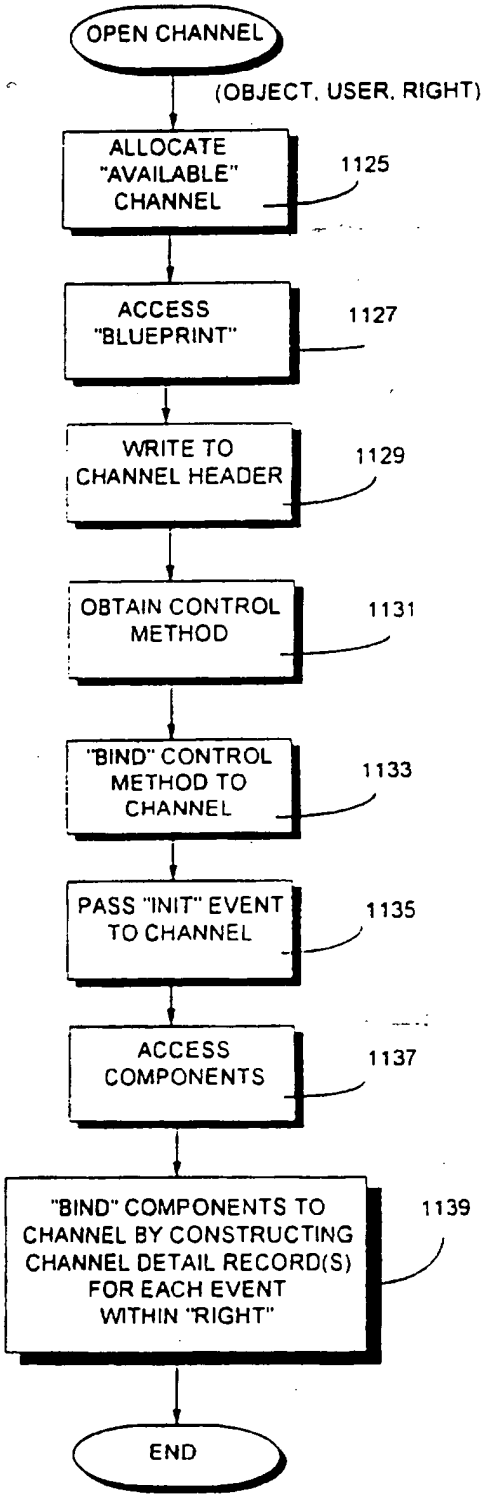
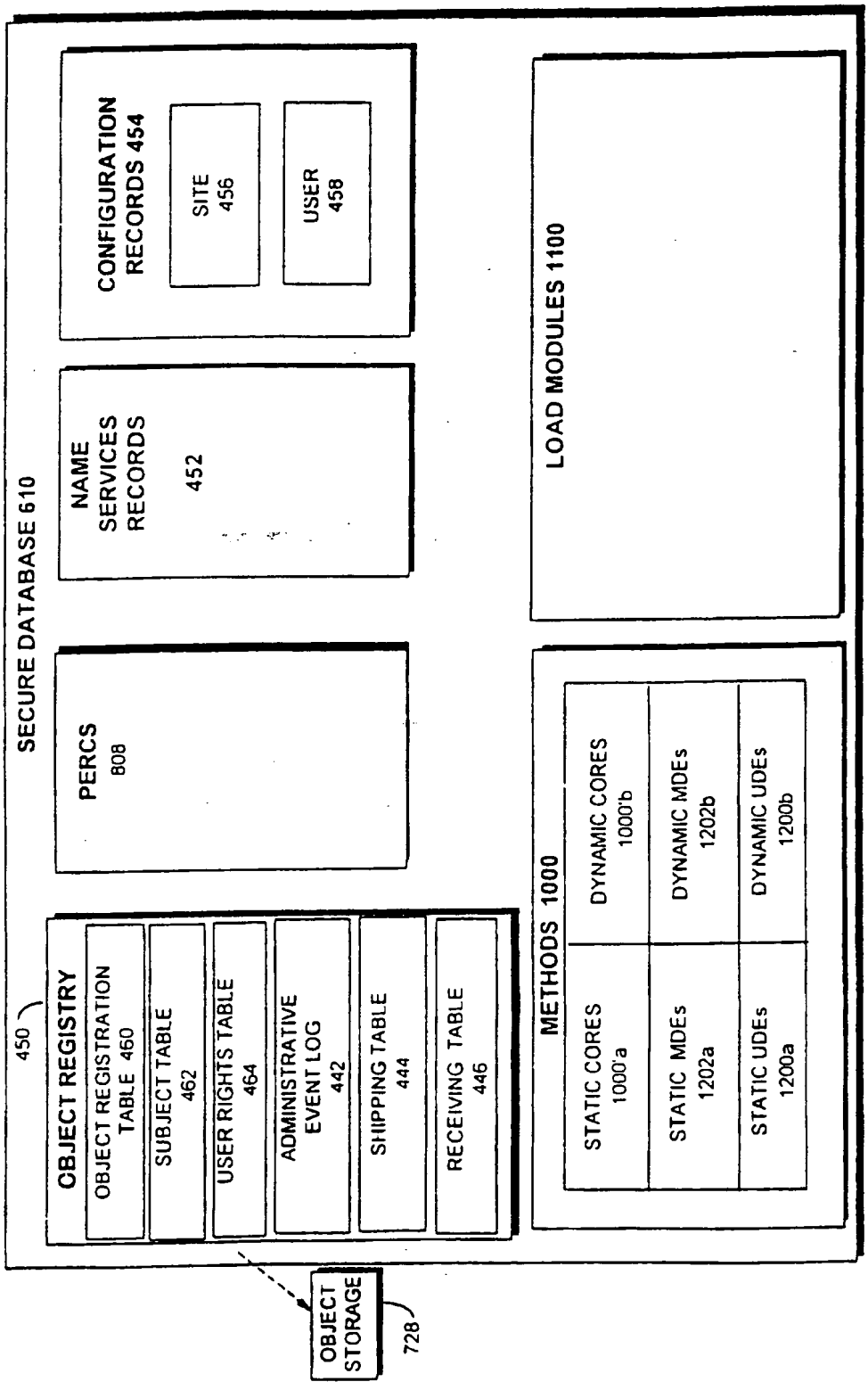


FIG. 16



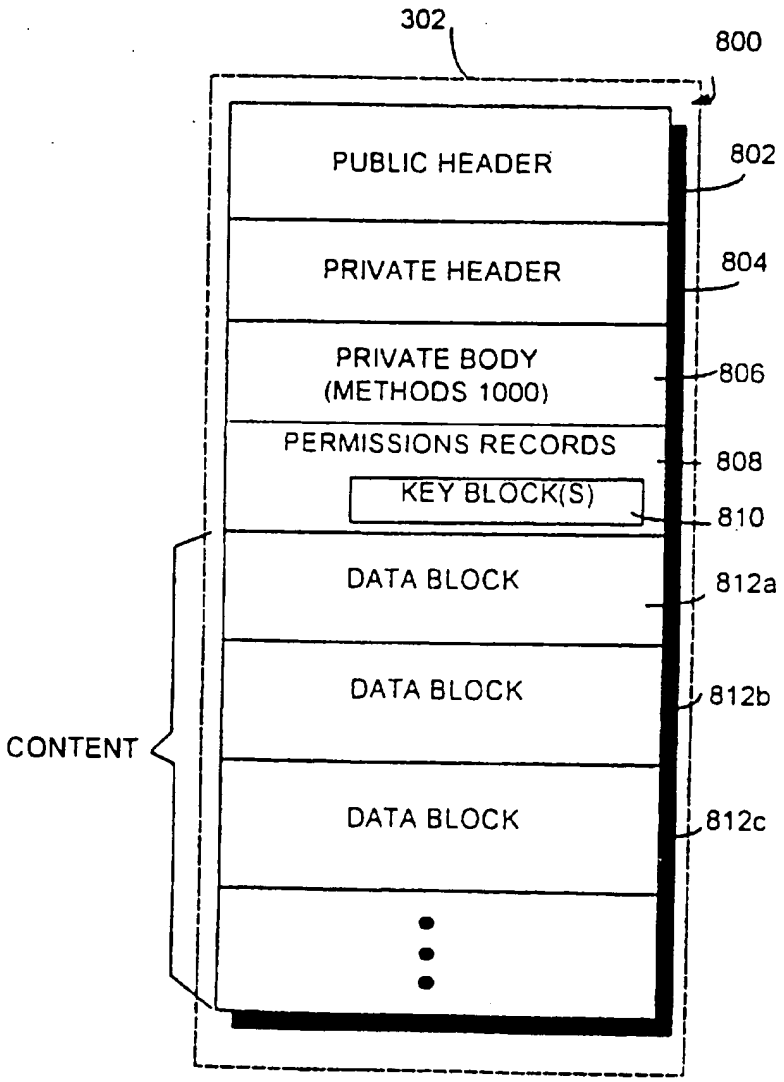


FIG. 17

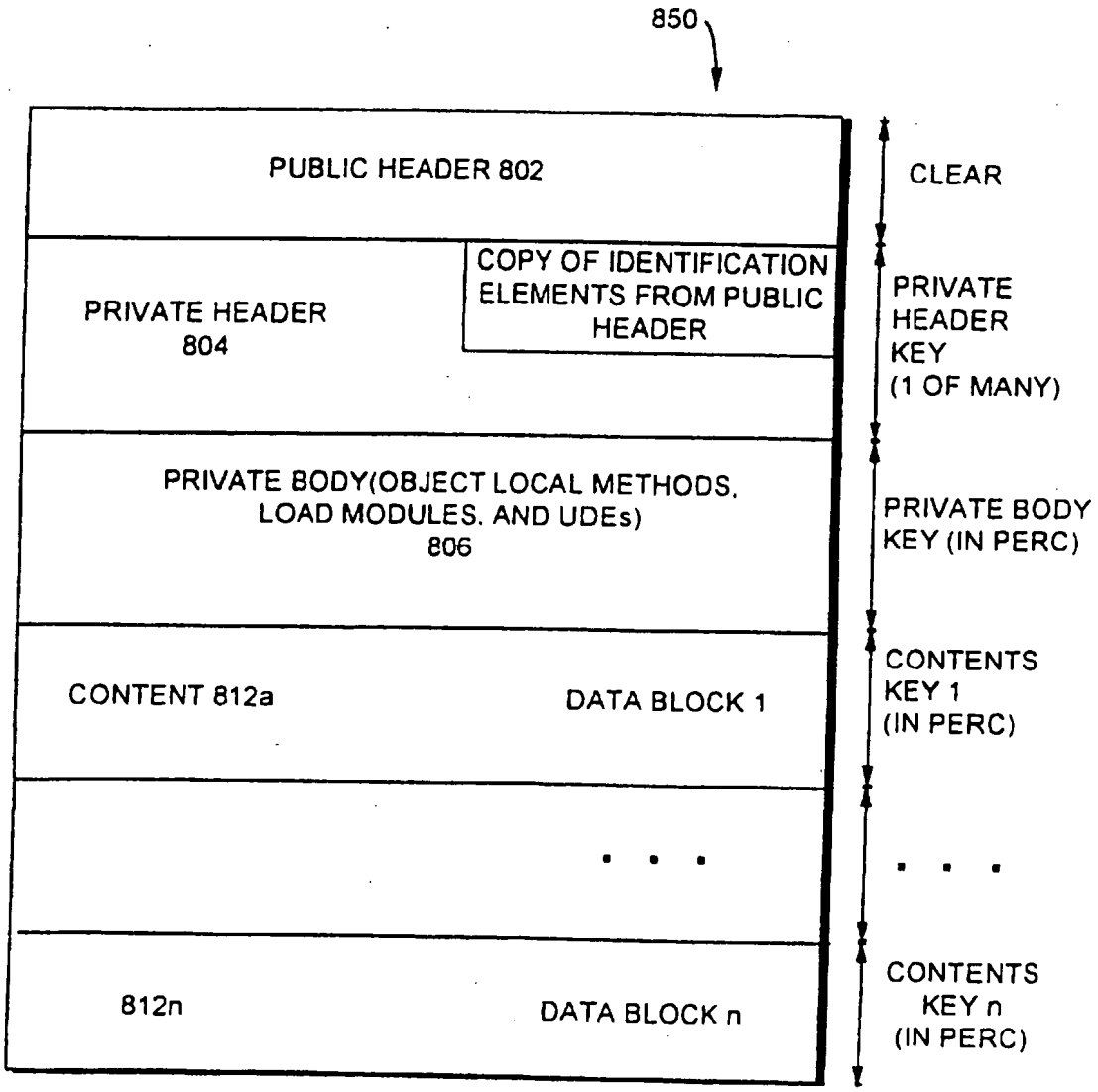
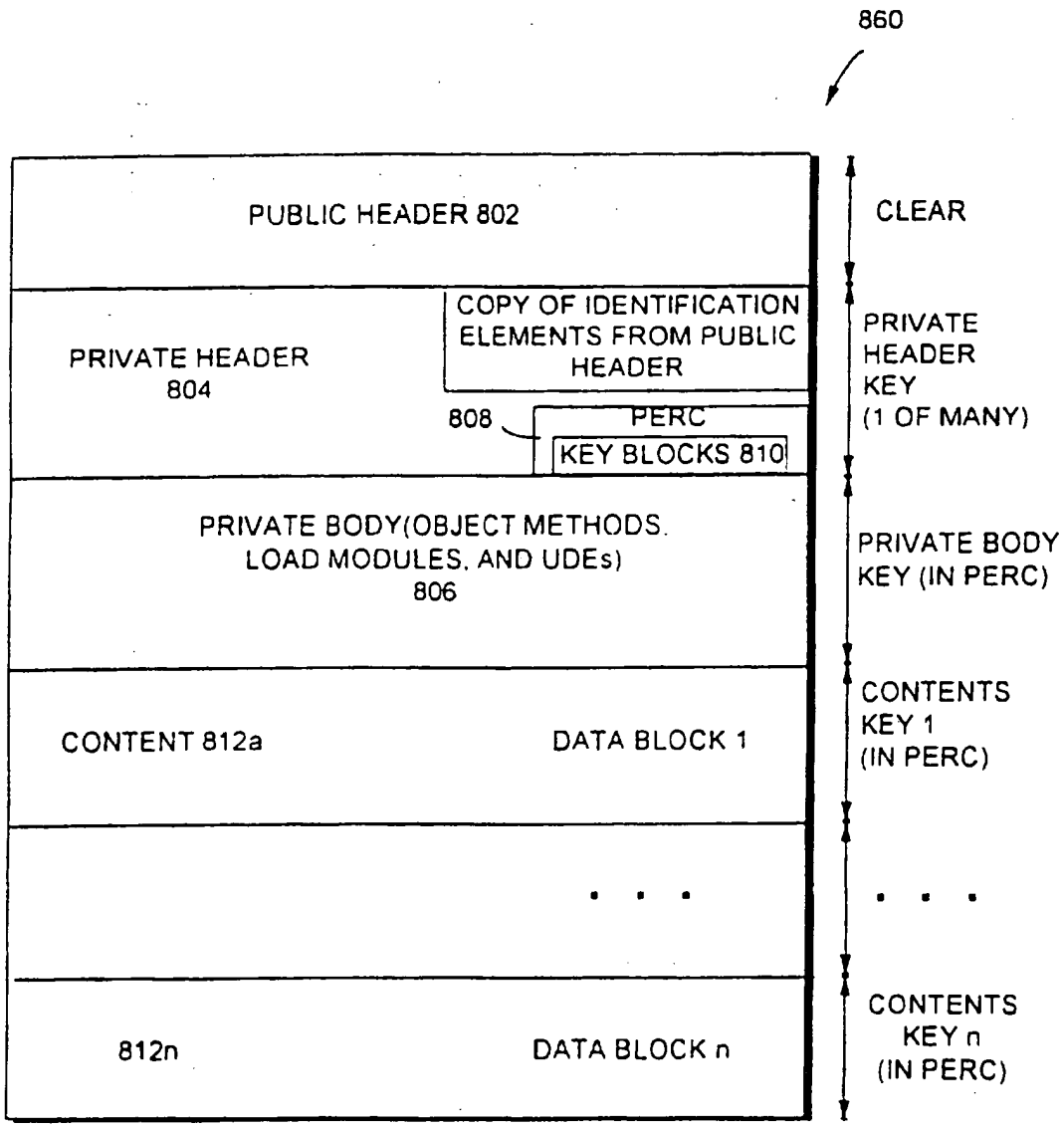
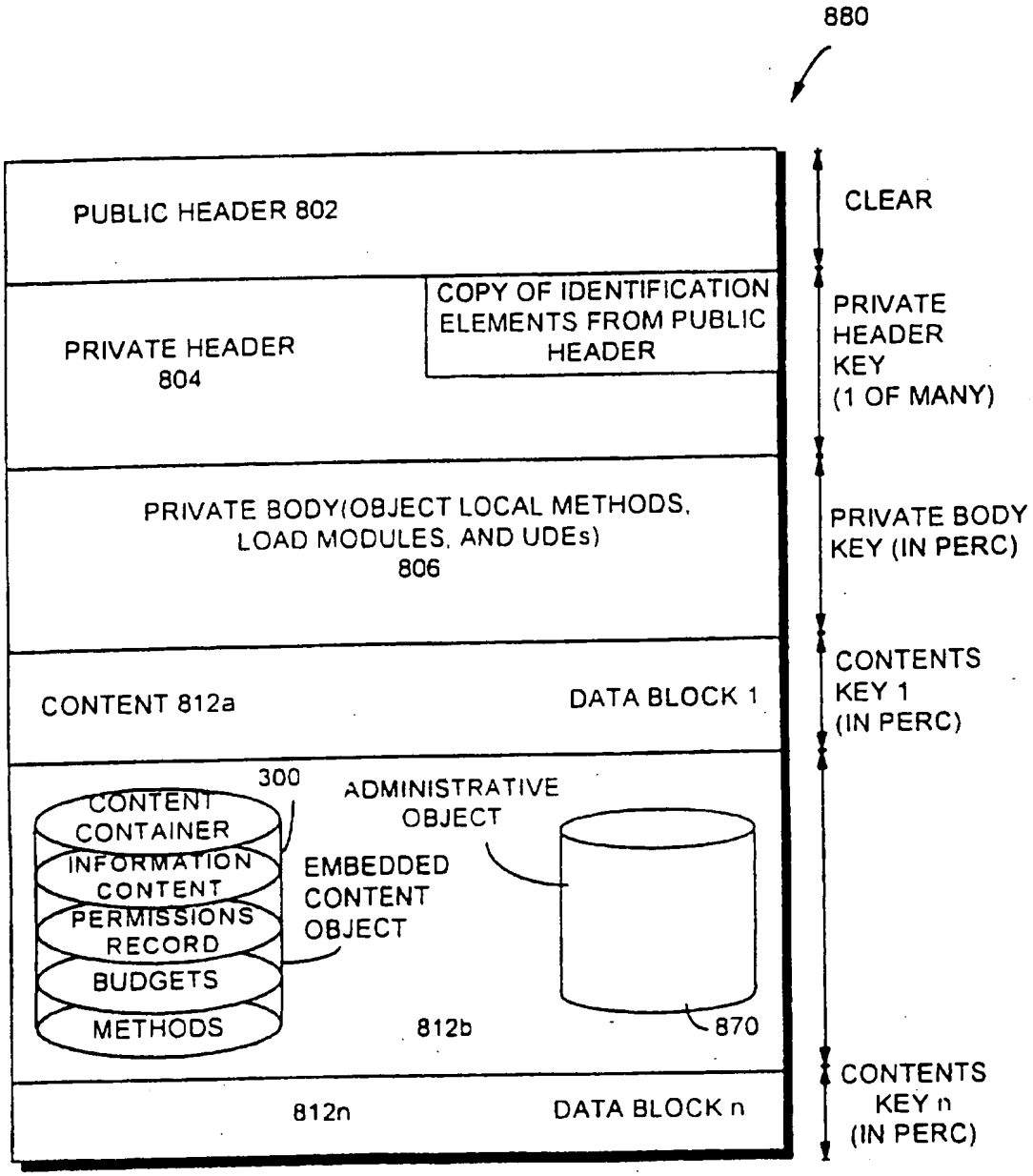


FIG. 18



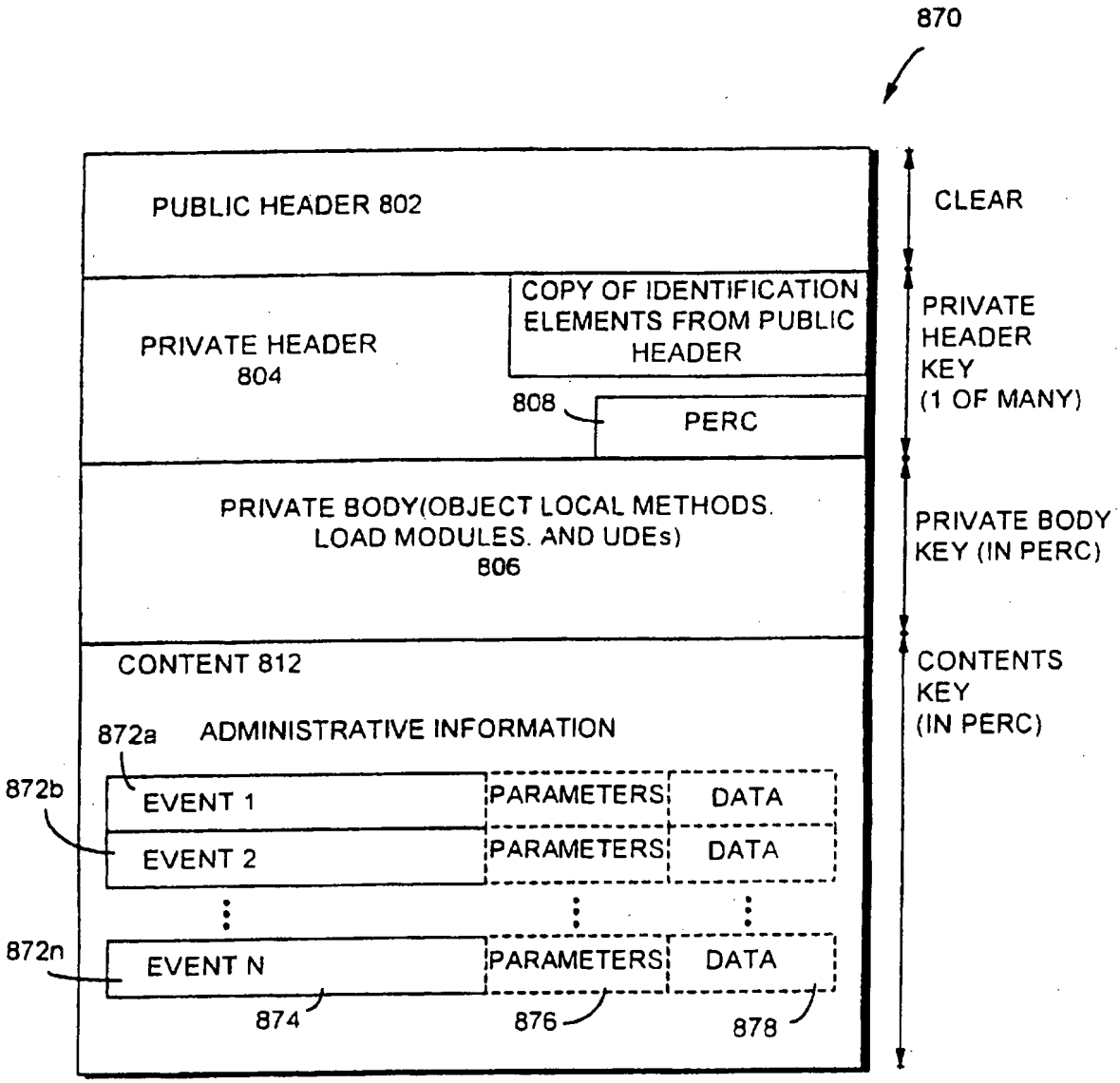
TRAVELING OBJECT

FIG. 19



CONTENT OBJECT

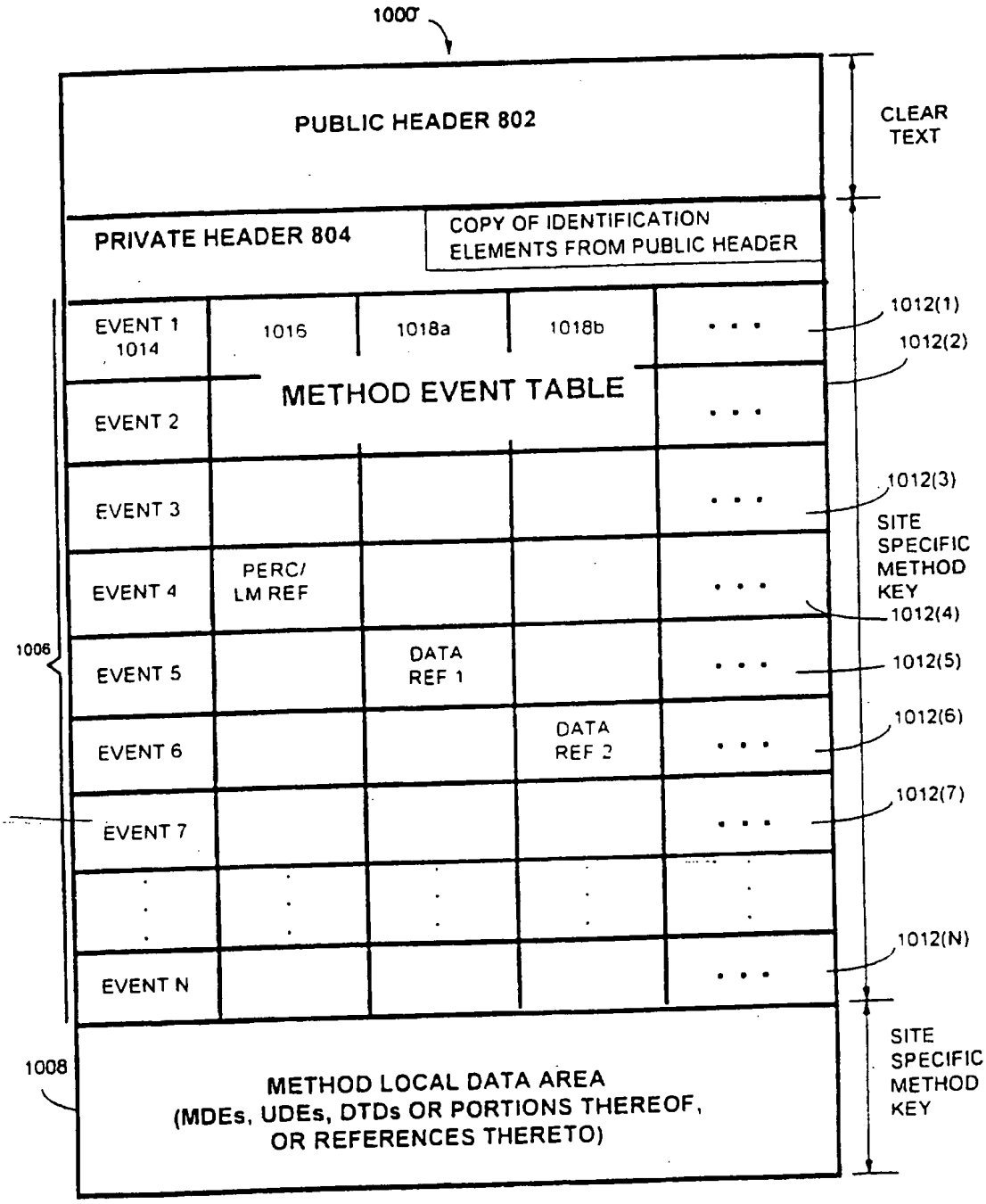
FIG. 20



ADMINISTRATIVE OBJECT

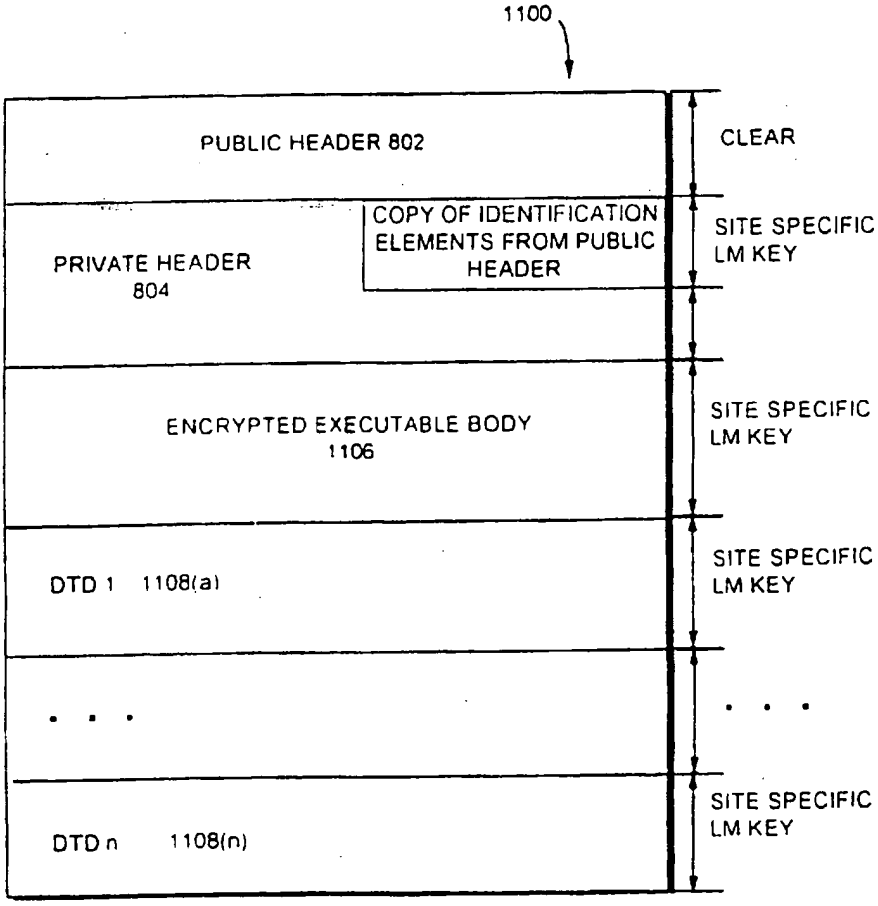
FIG. 21

FIG. 22



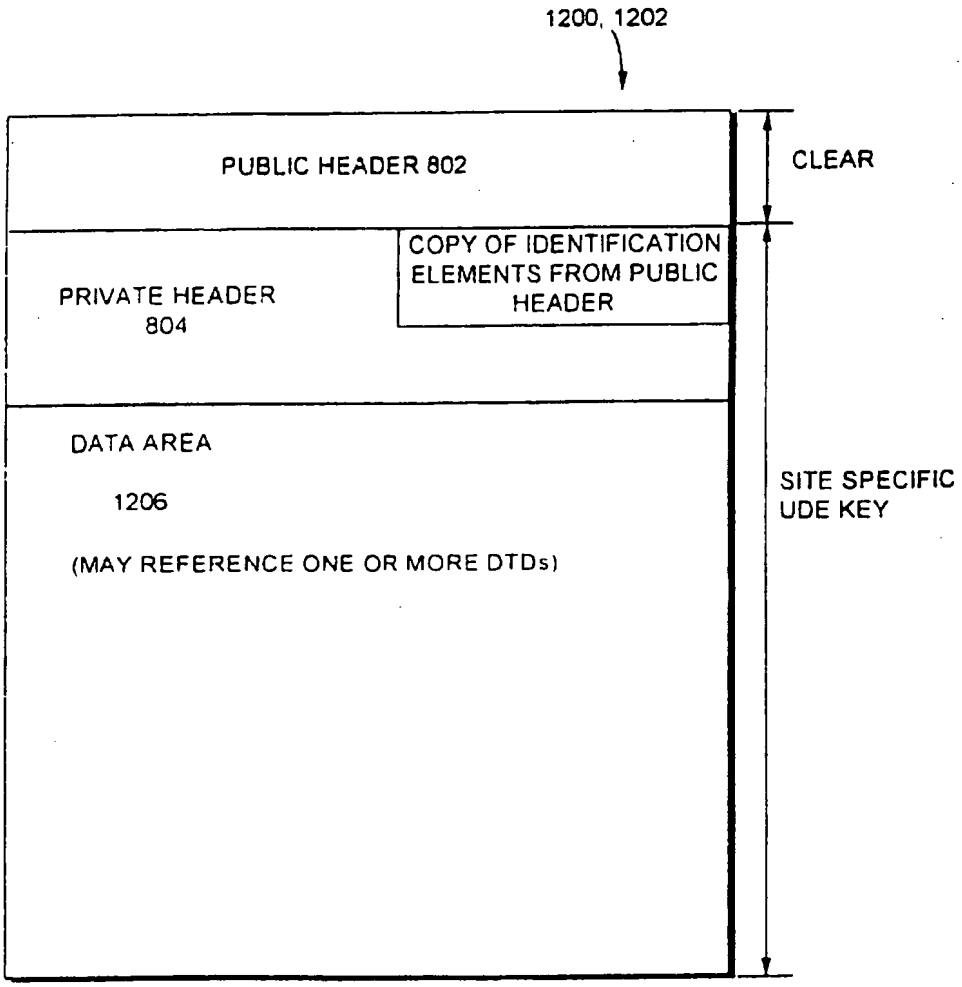
METHOD "CORE"

FIG. 23



LOAD MODULE

FIG. 24



UDE (MDE)

FIG. 25A

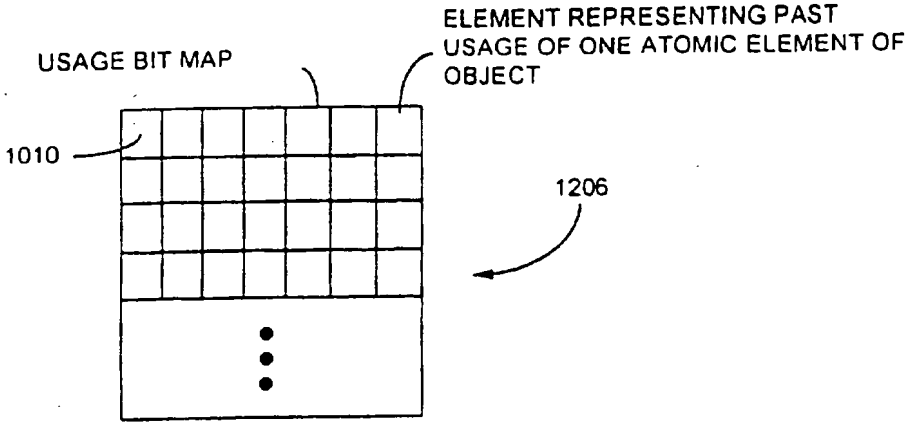


FIG. 25B

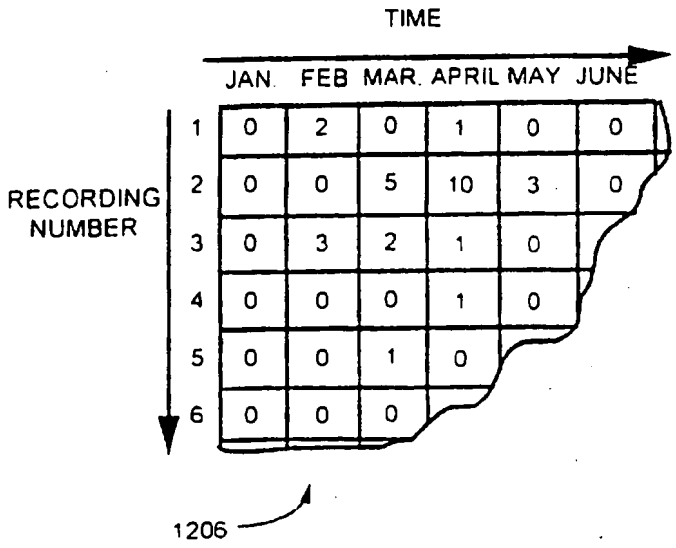


FIG. 25C

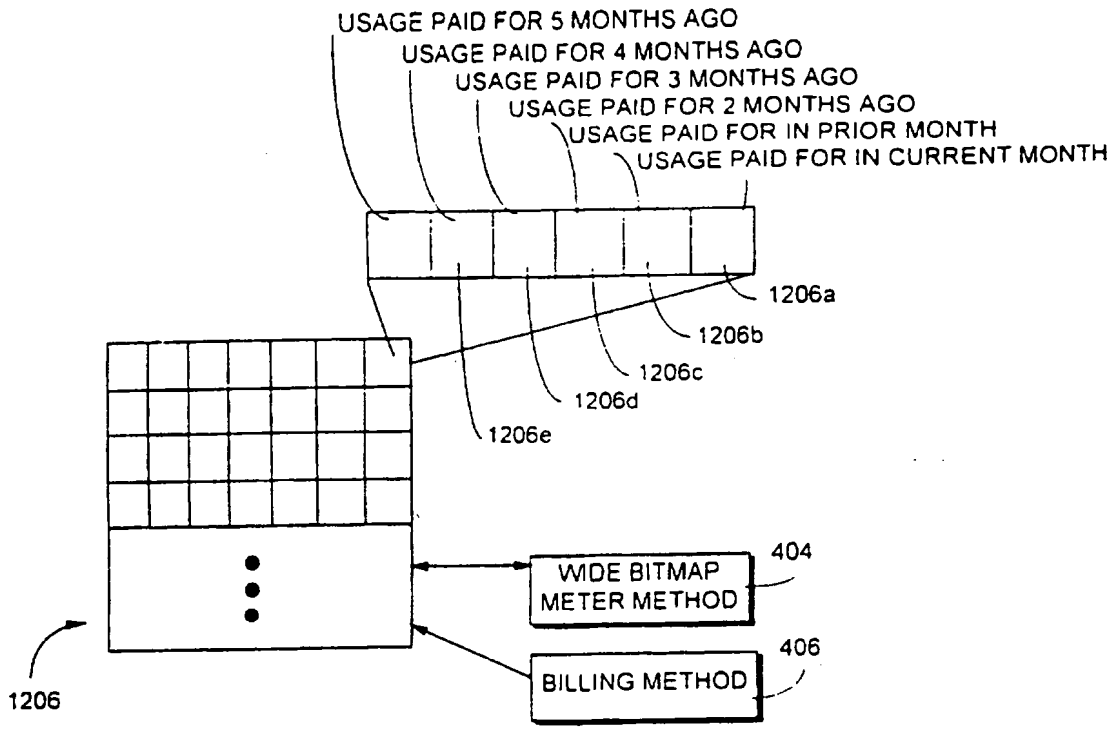


FIG. 26

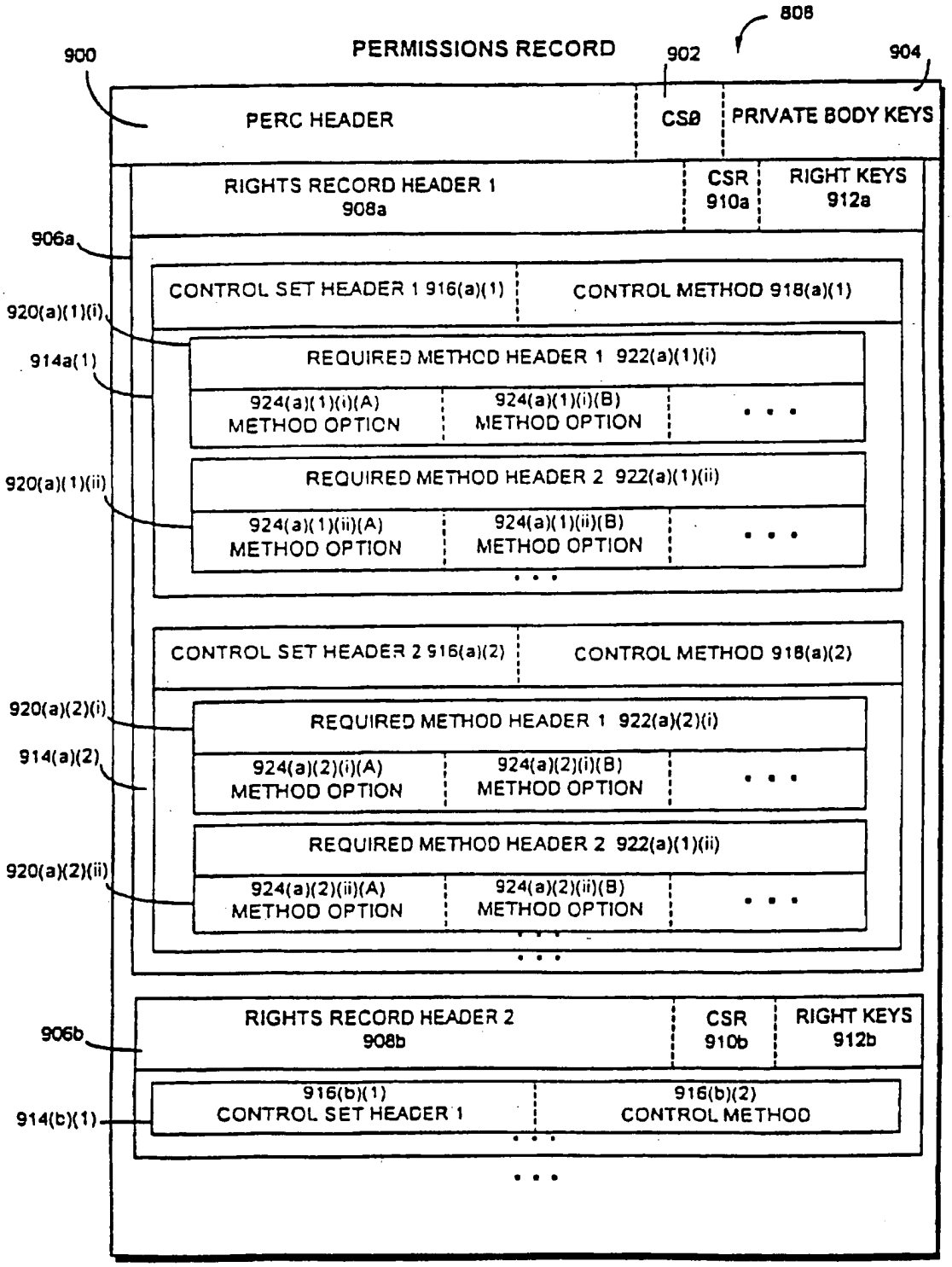
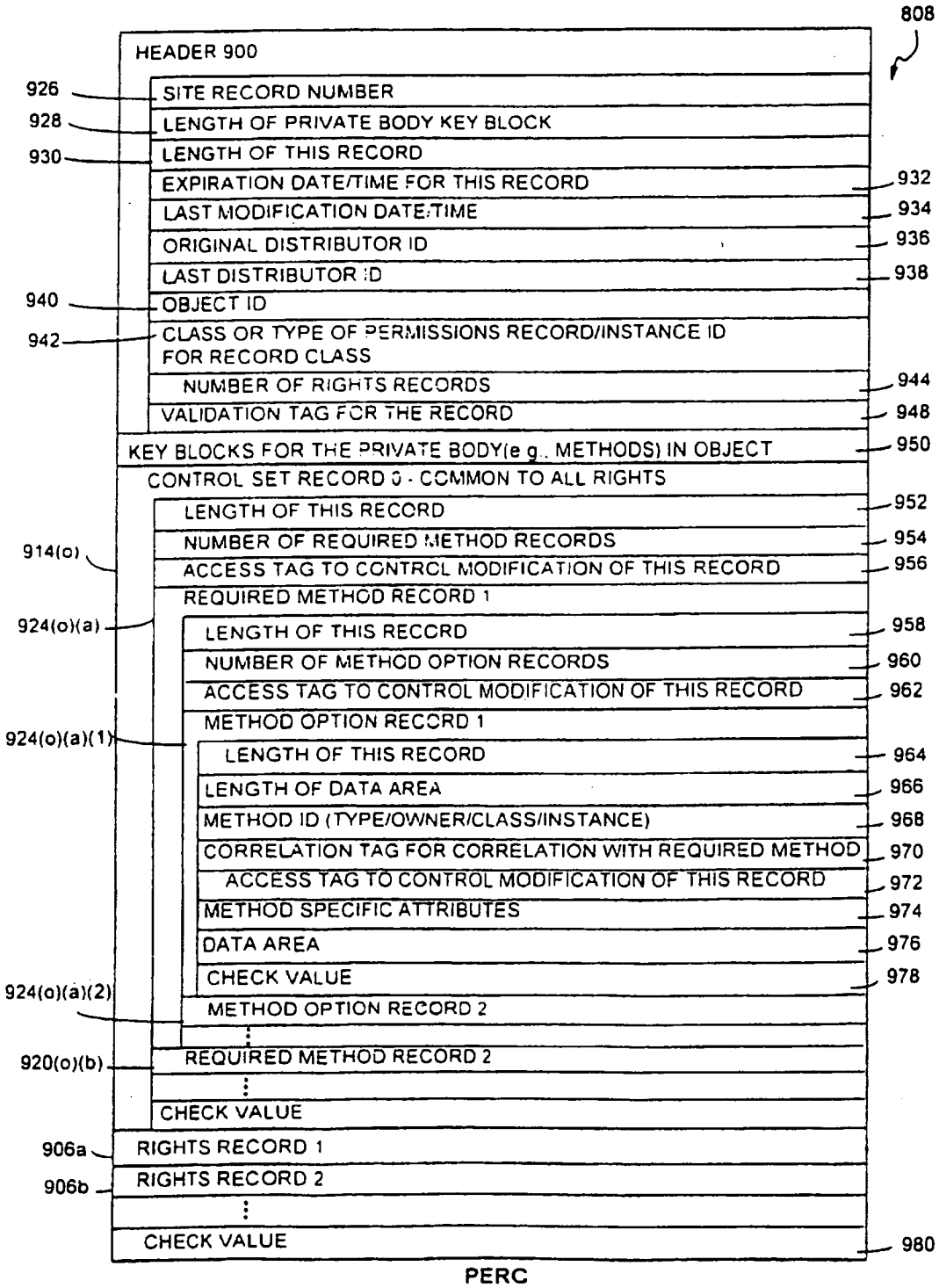
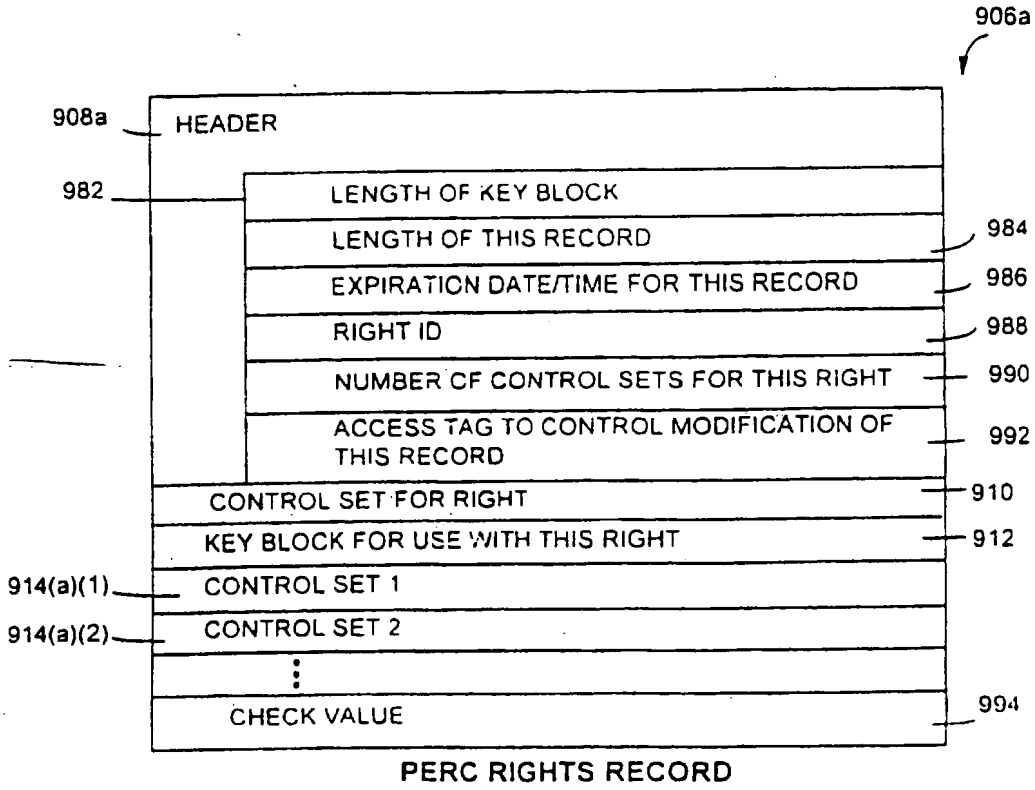


FIG. 26A



PERC

FIG. 26B



PERC RIGHTS RECORD

FIG. 27
SHIPPING TABLE

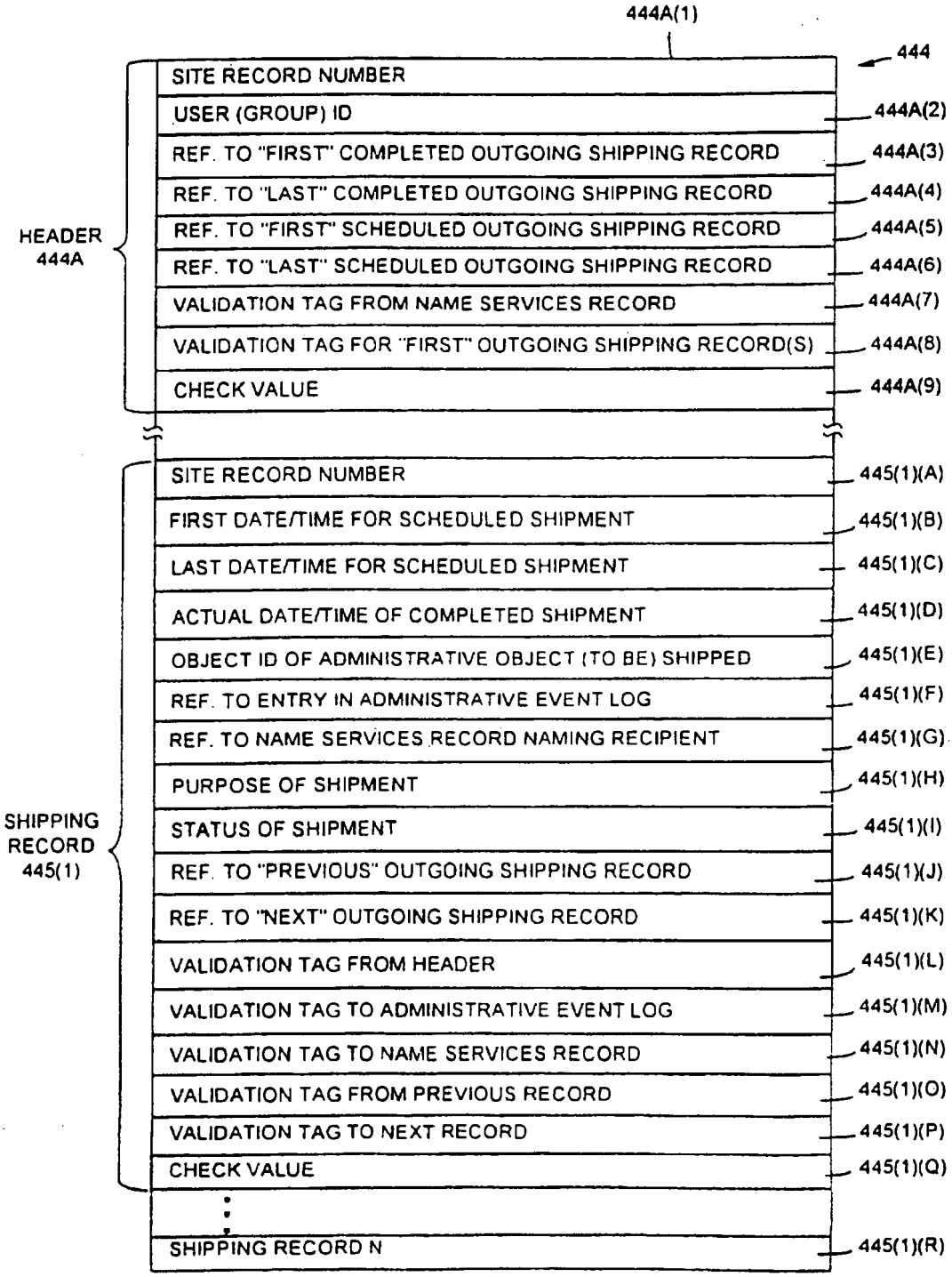


FIG. 28
RECEIVING TABLE

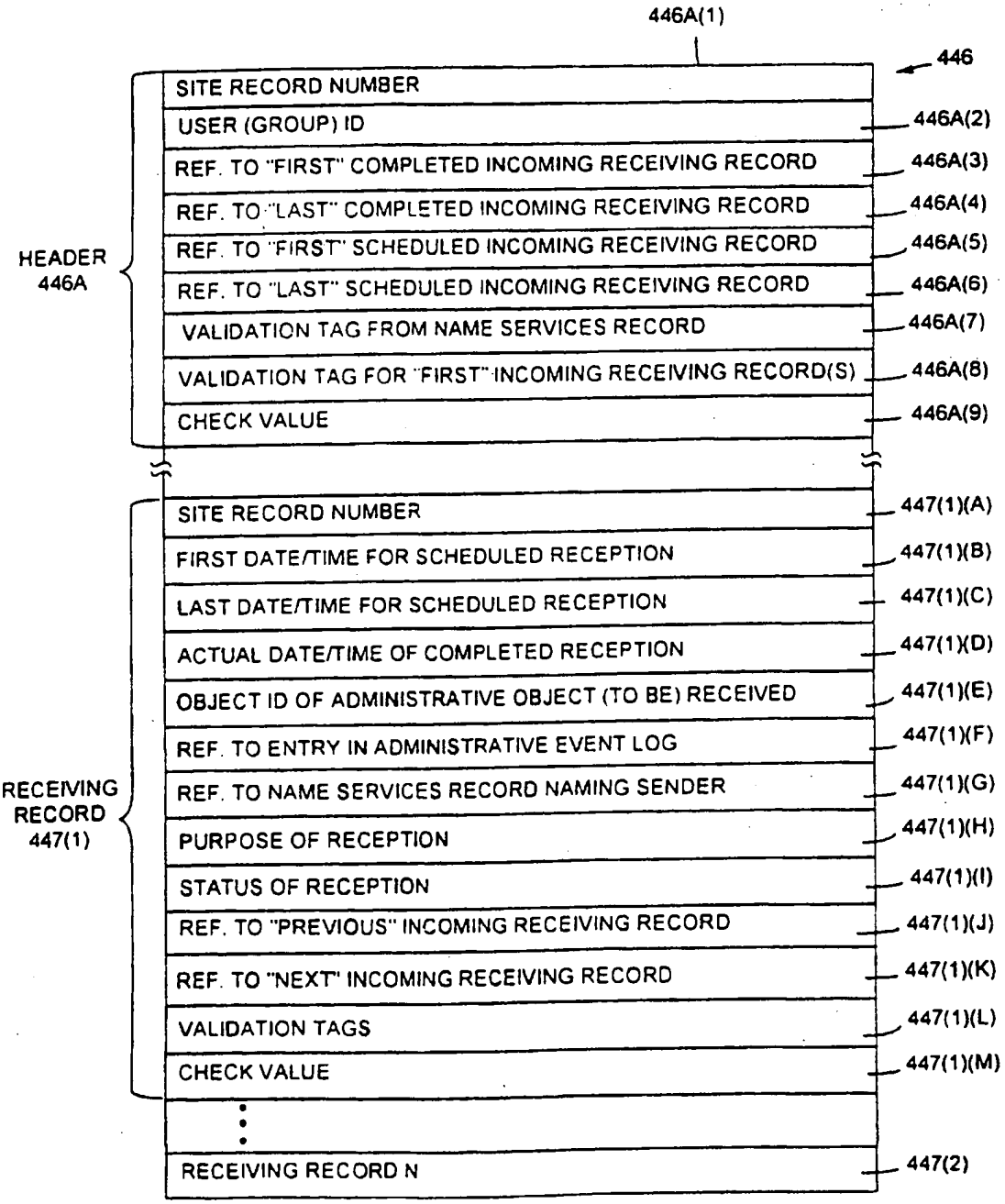
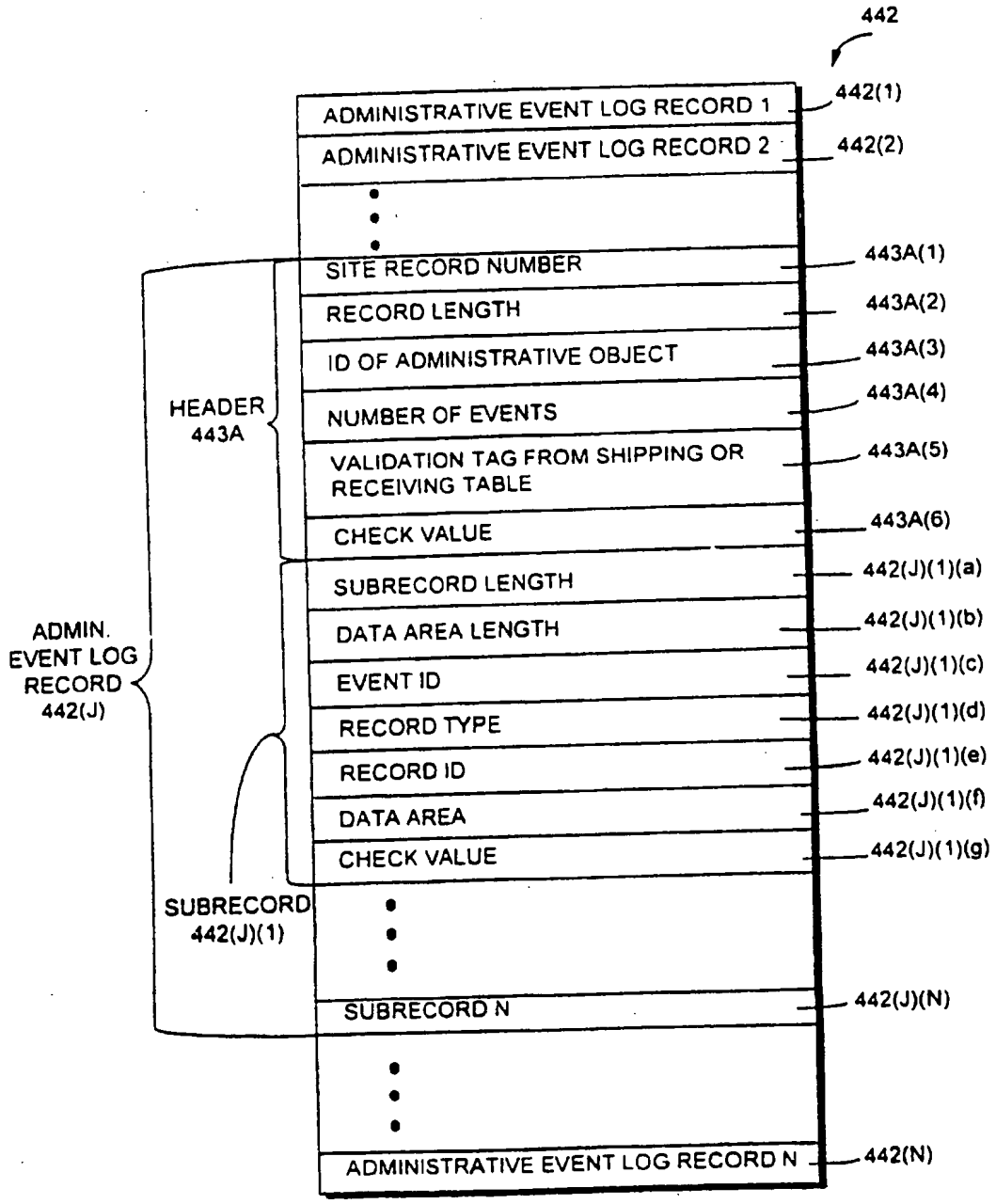


FIG. 29
ADMINISTRATIVE EVENT LOG



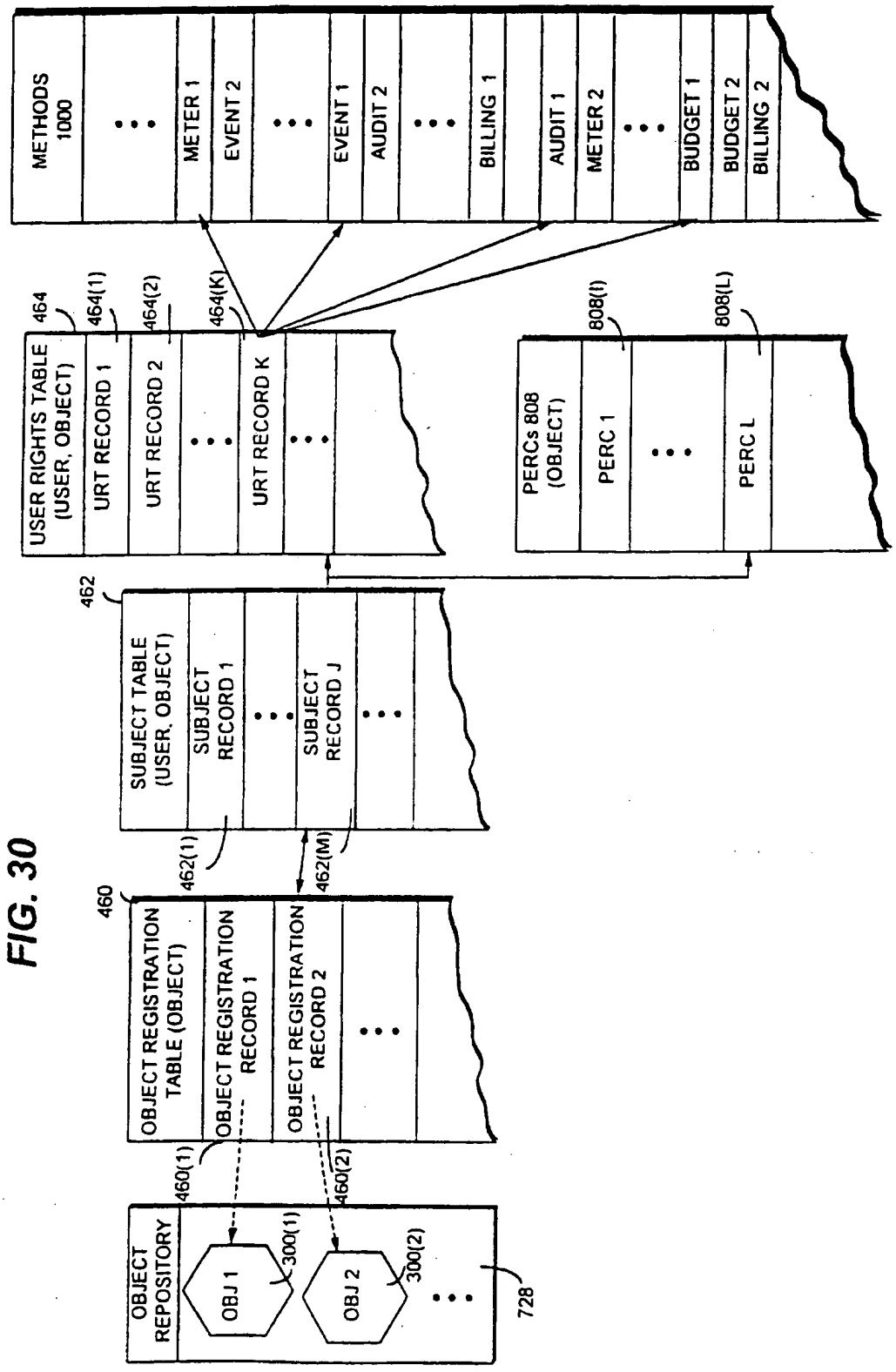


FIG. 30

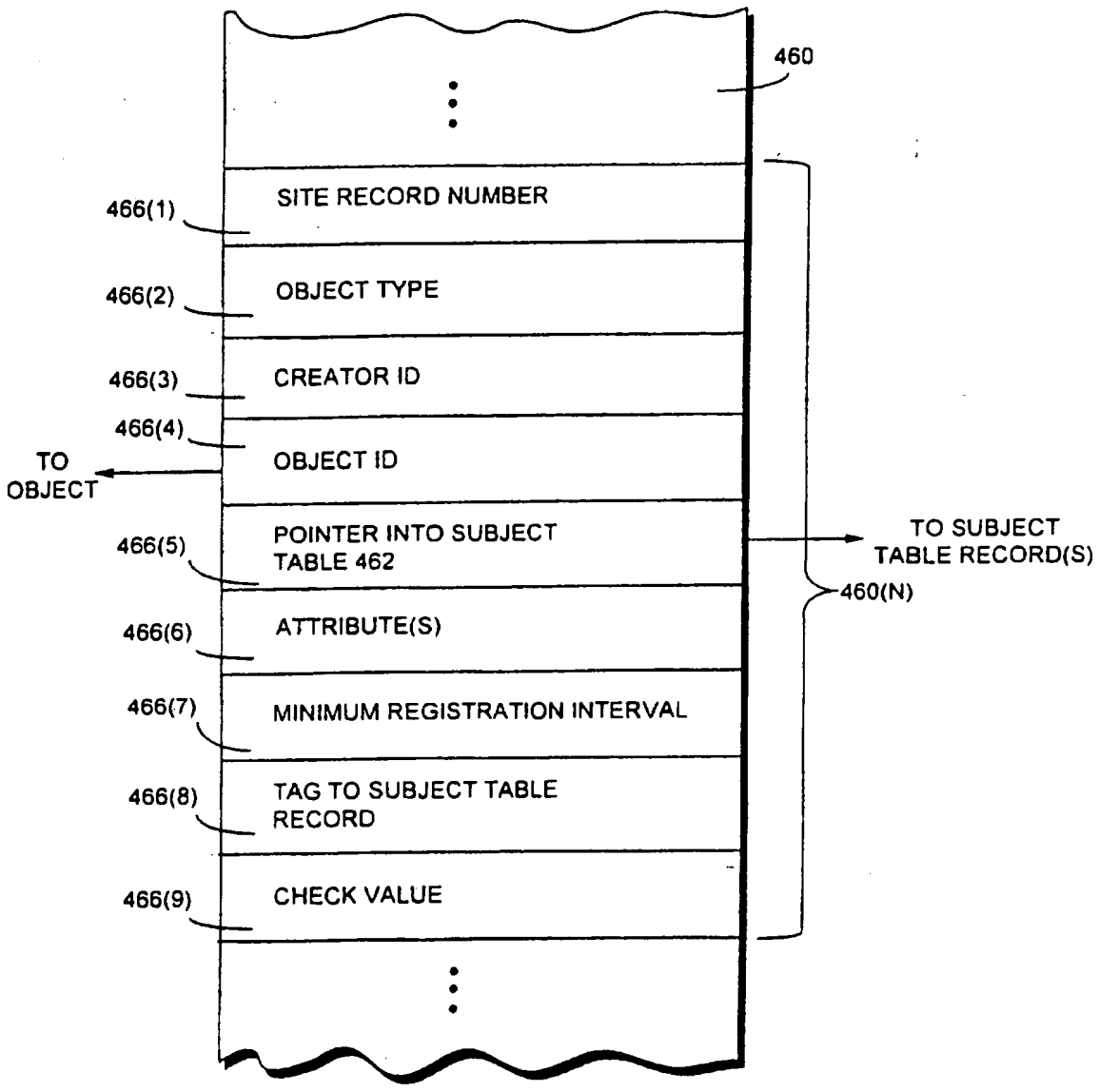


FIG. 31
OBJECT REGISTRATION TABLE

FIG. 32

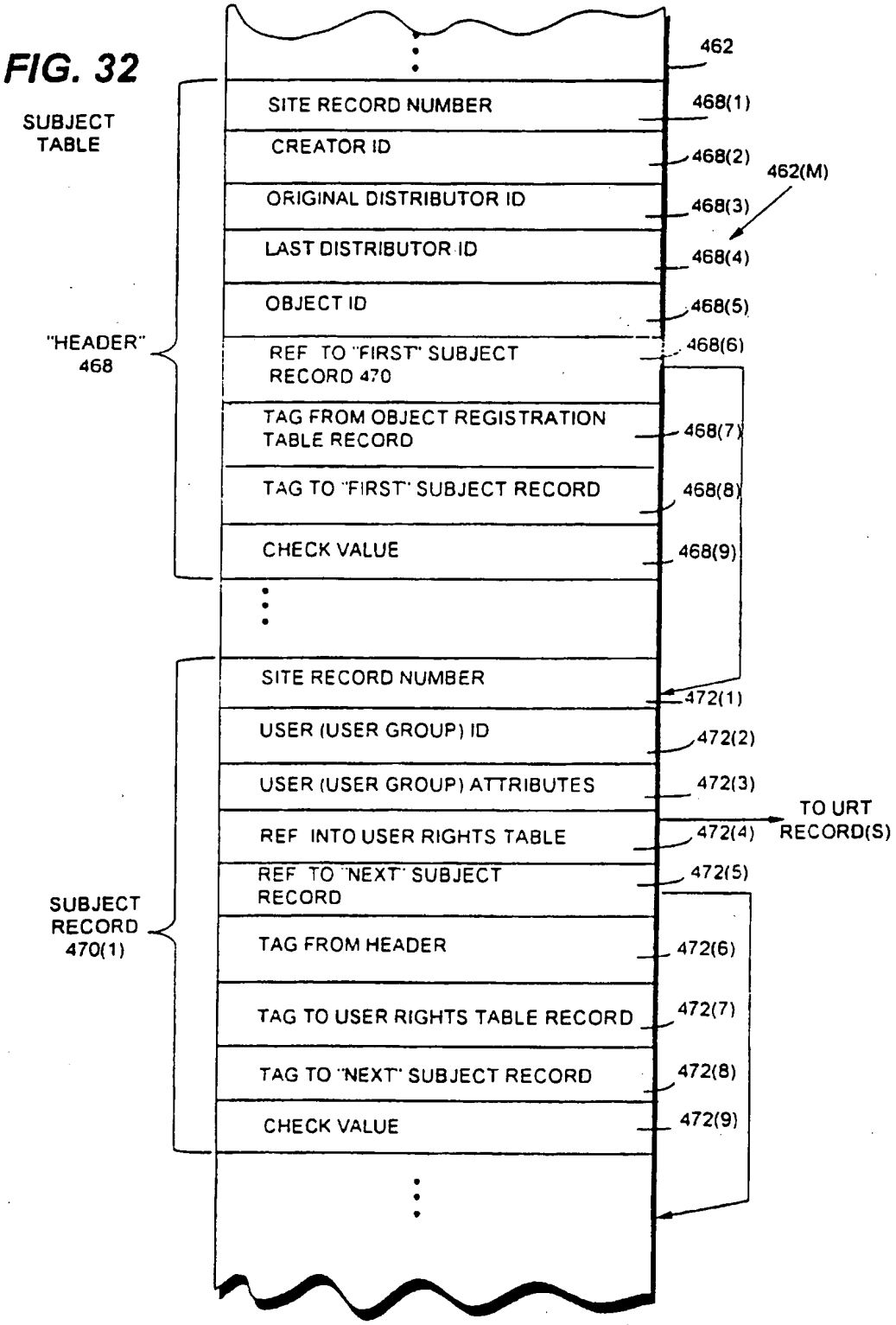


FIG. 33 USER RIGHTS TABLE

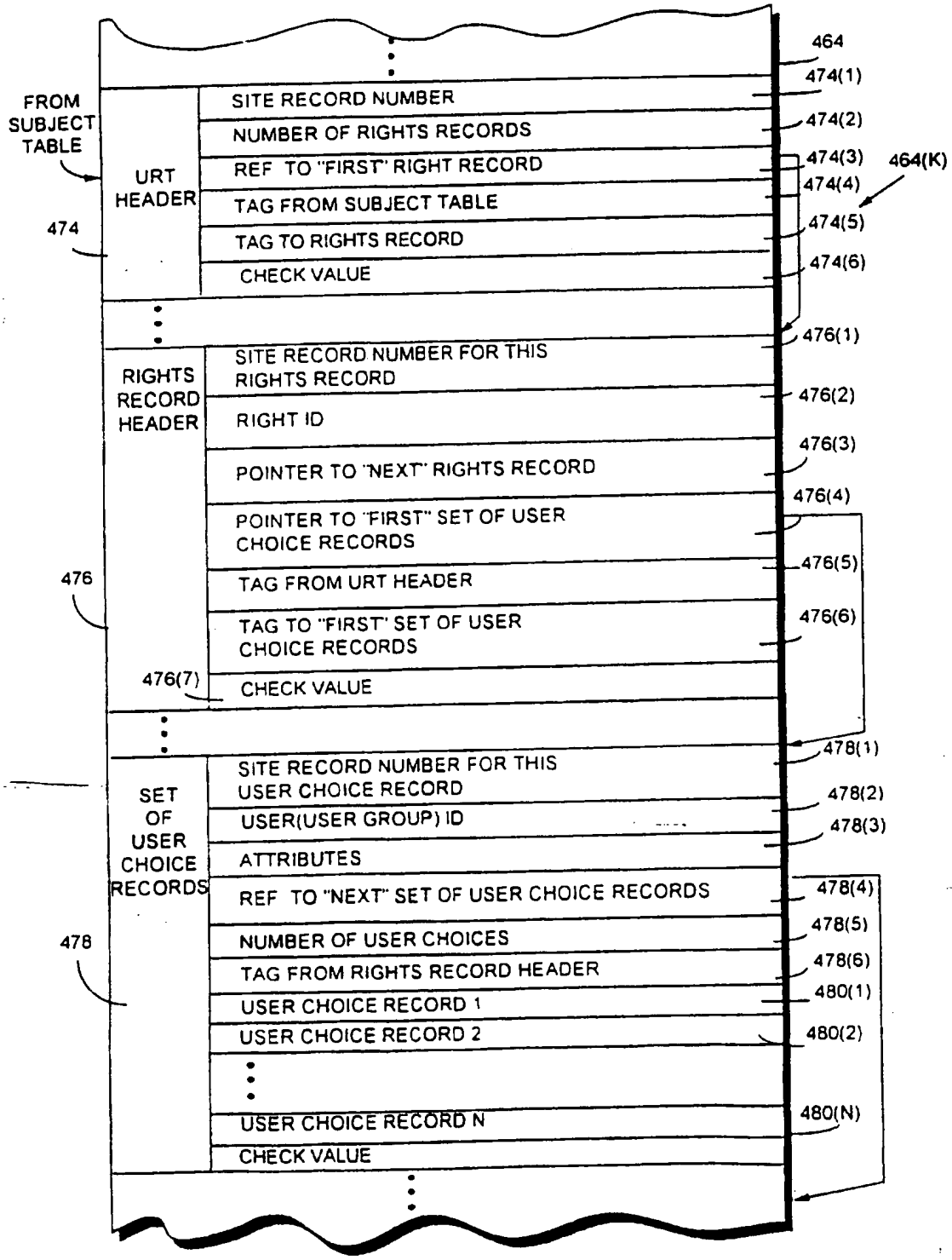


FIG. 34

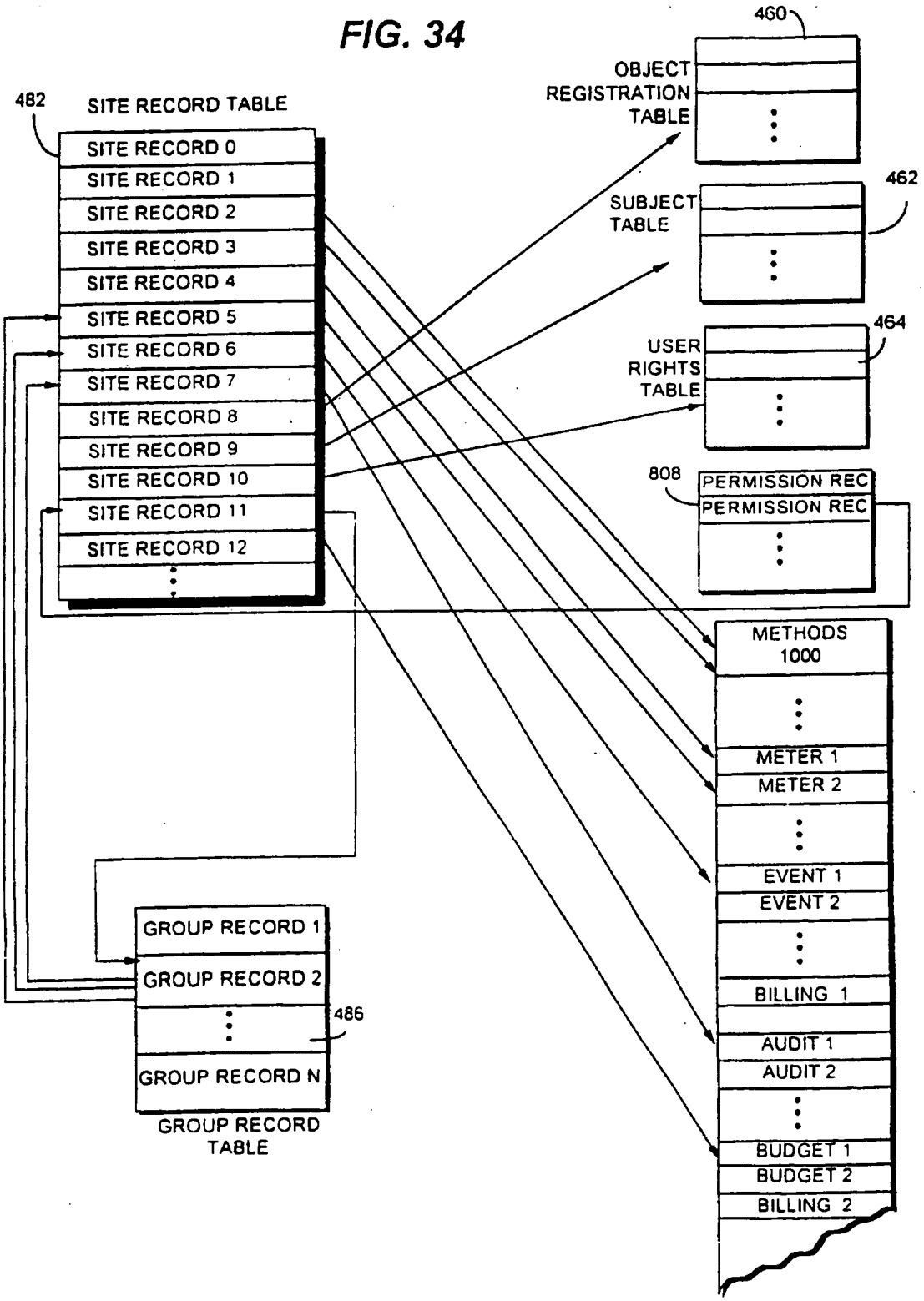


FIG. 34A

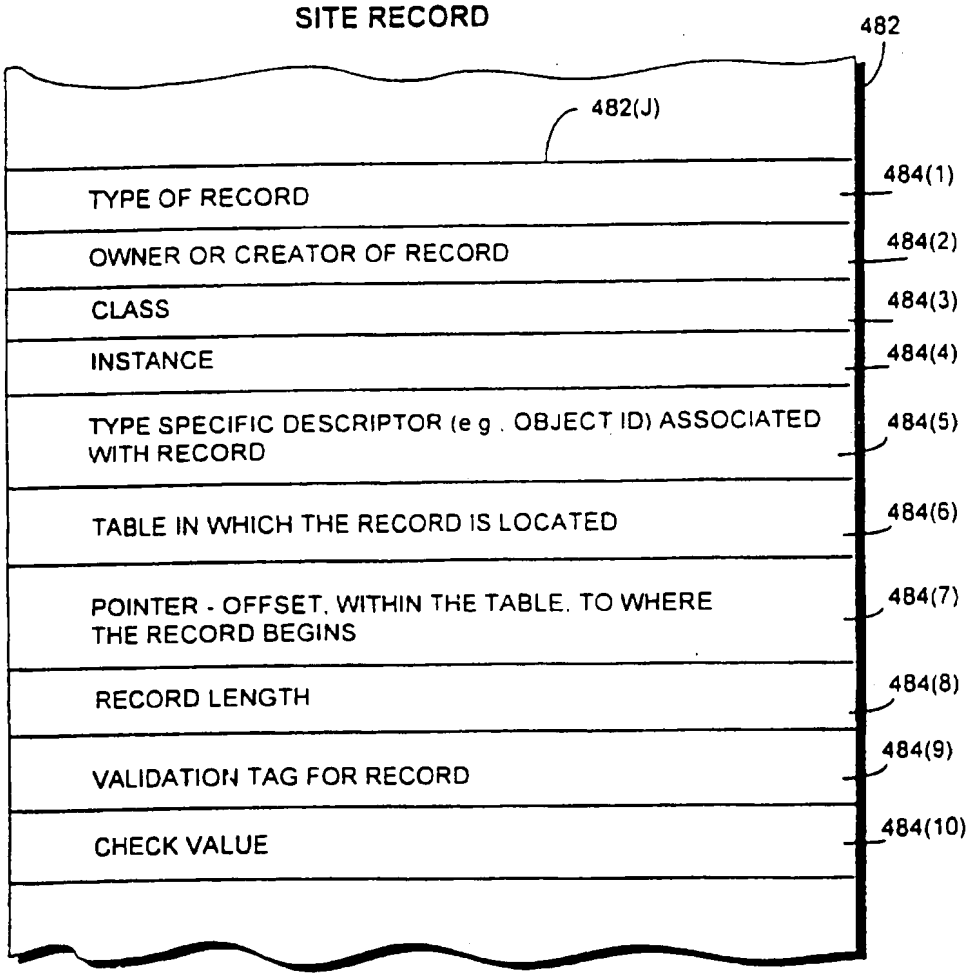


FIG. 34B

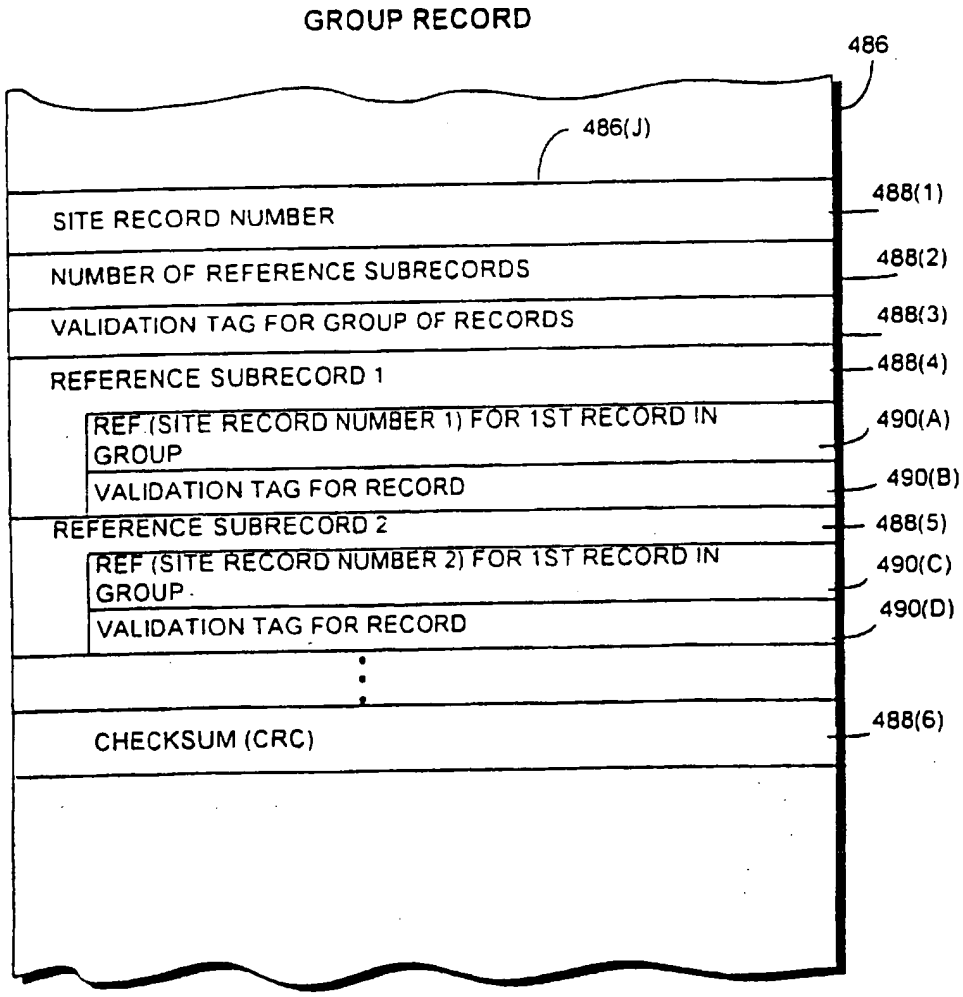


FIG. 35

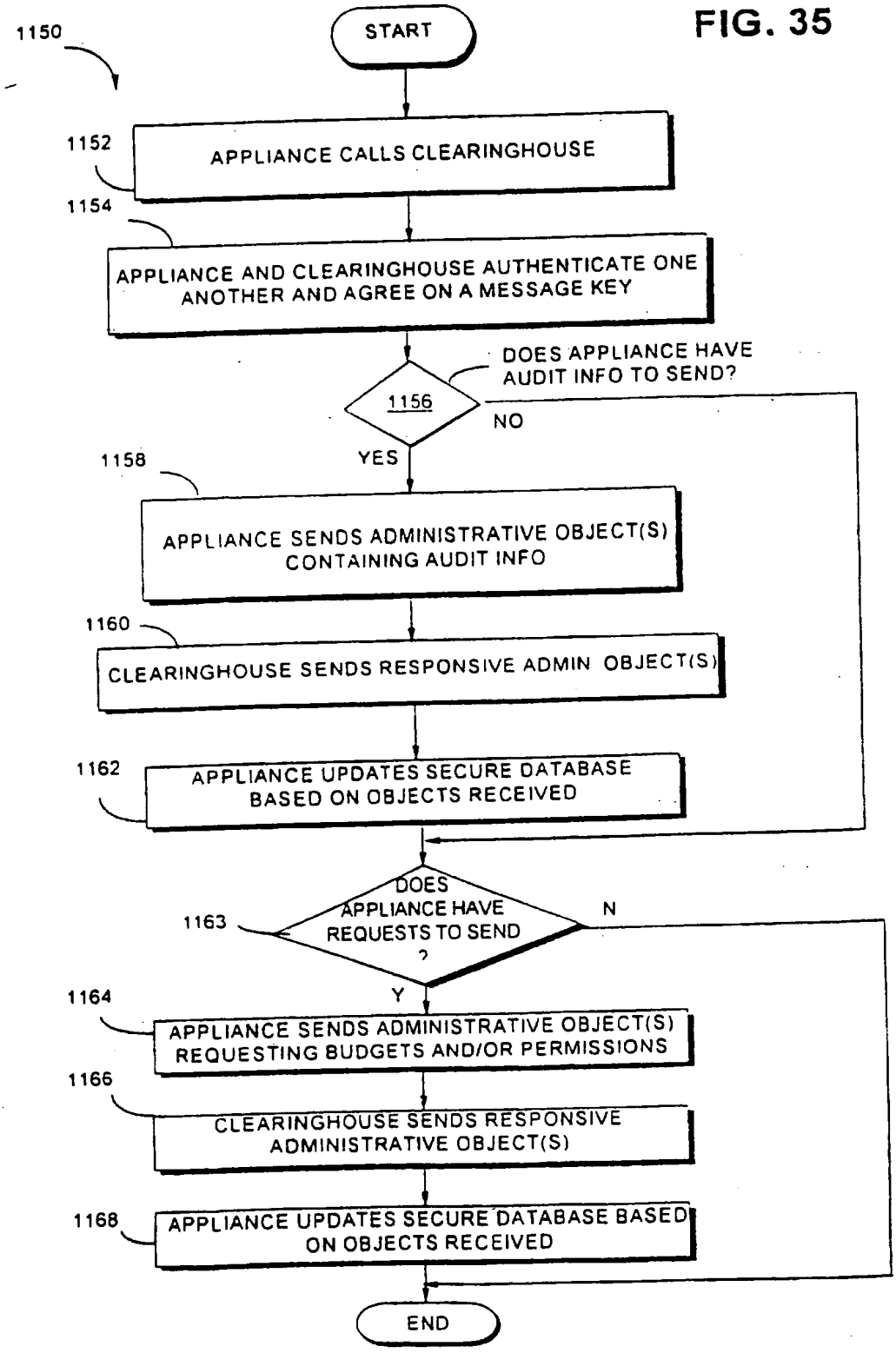
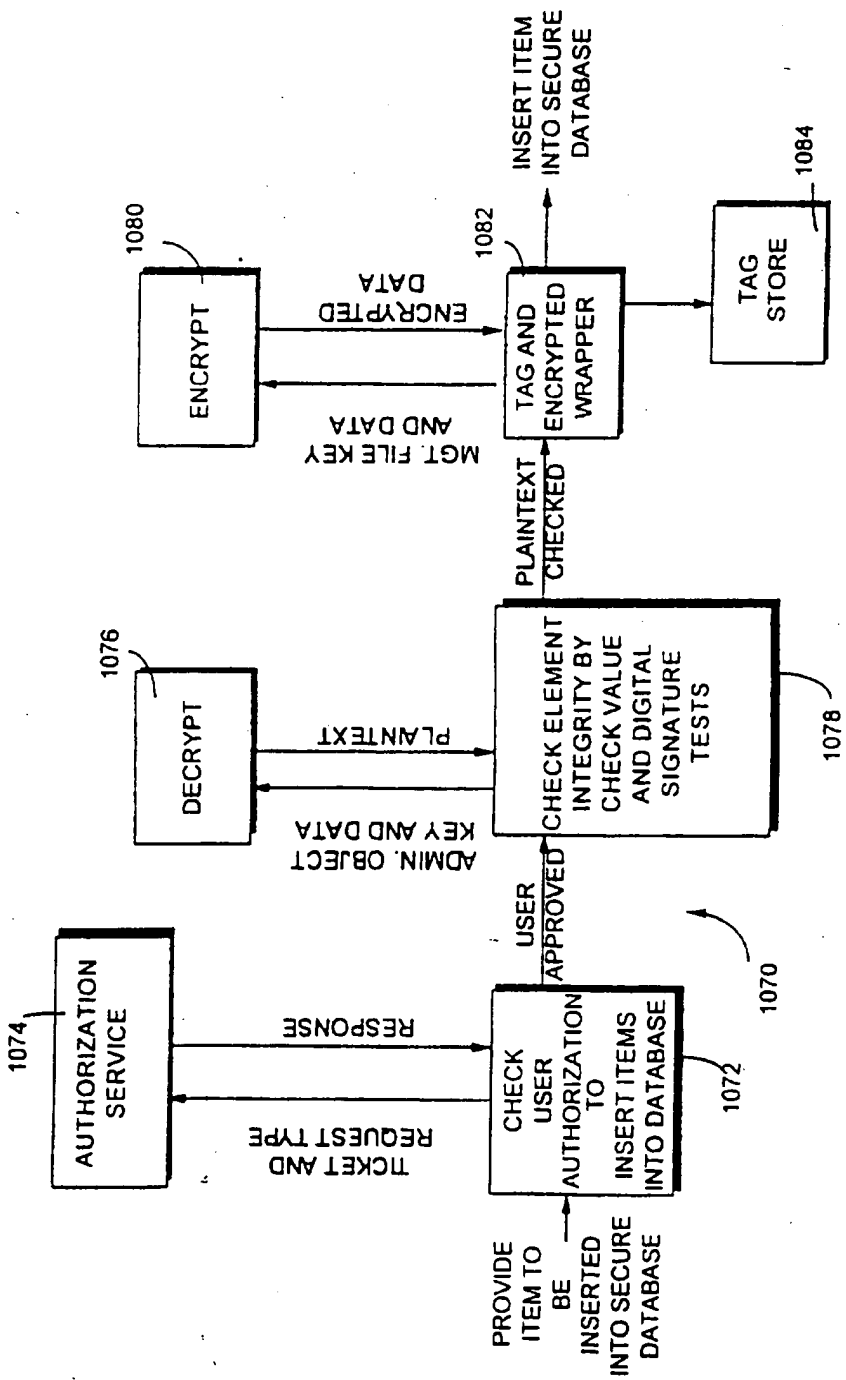


FIG. 36



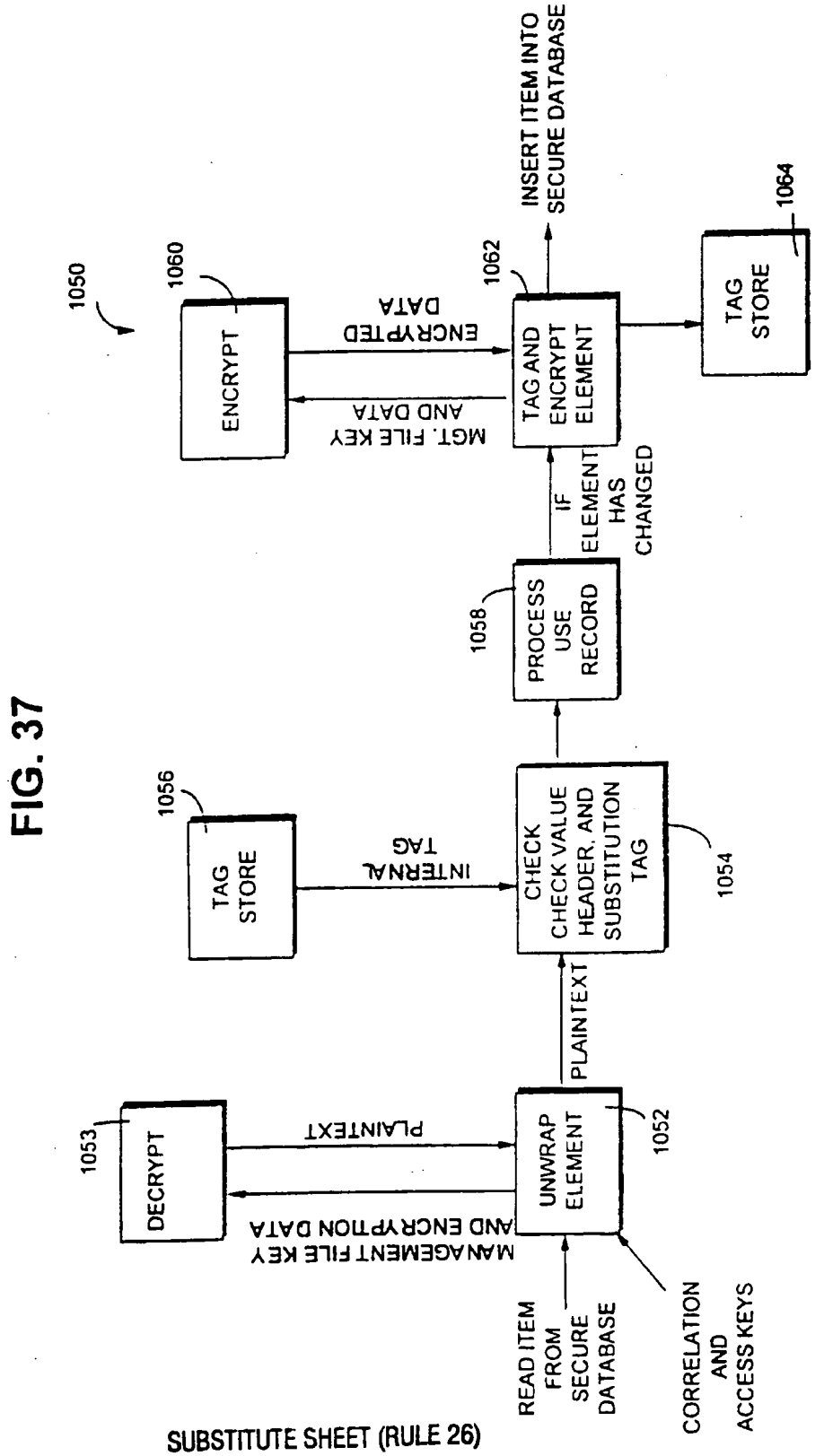


FIG. 37

FIG. 38

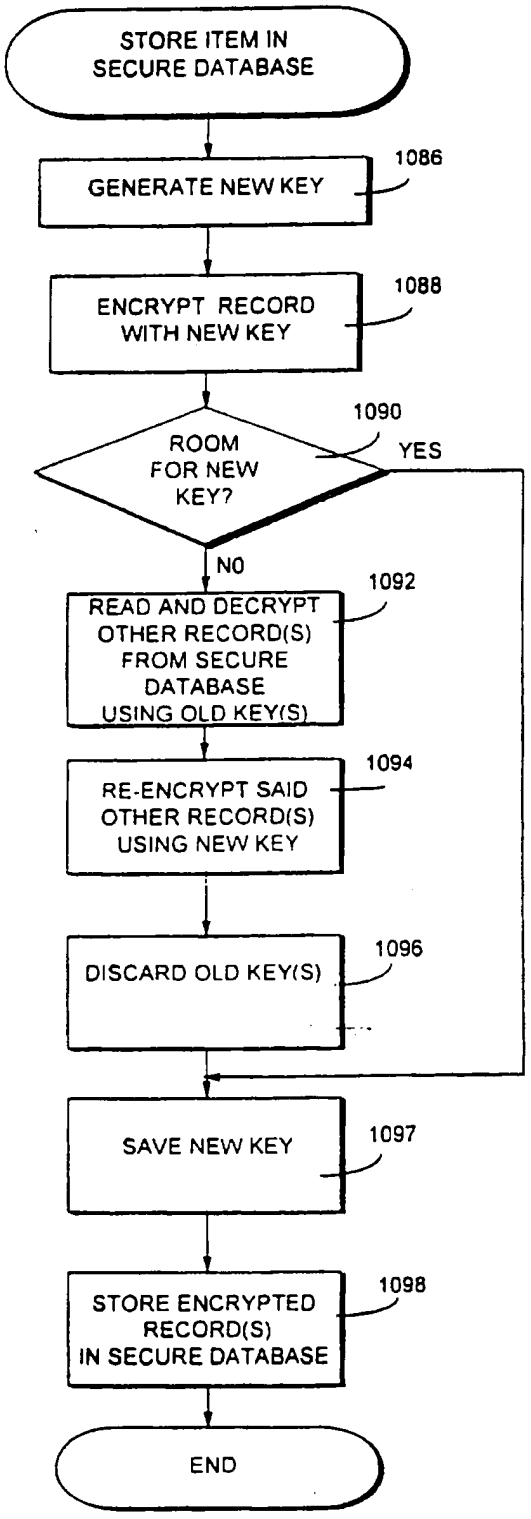


FIG. 39
BACKUP

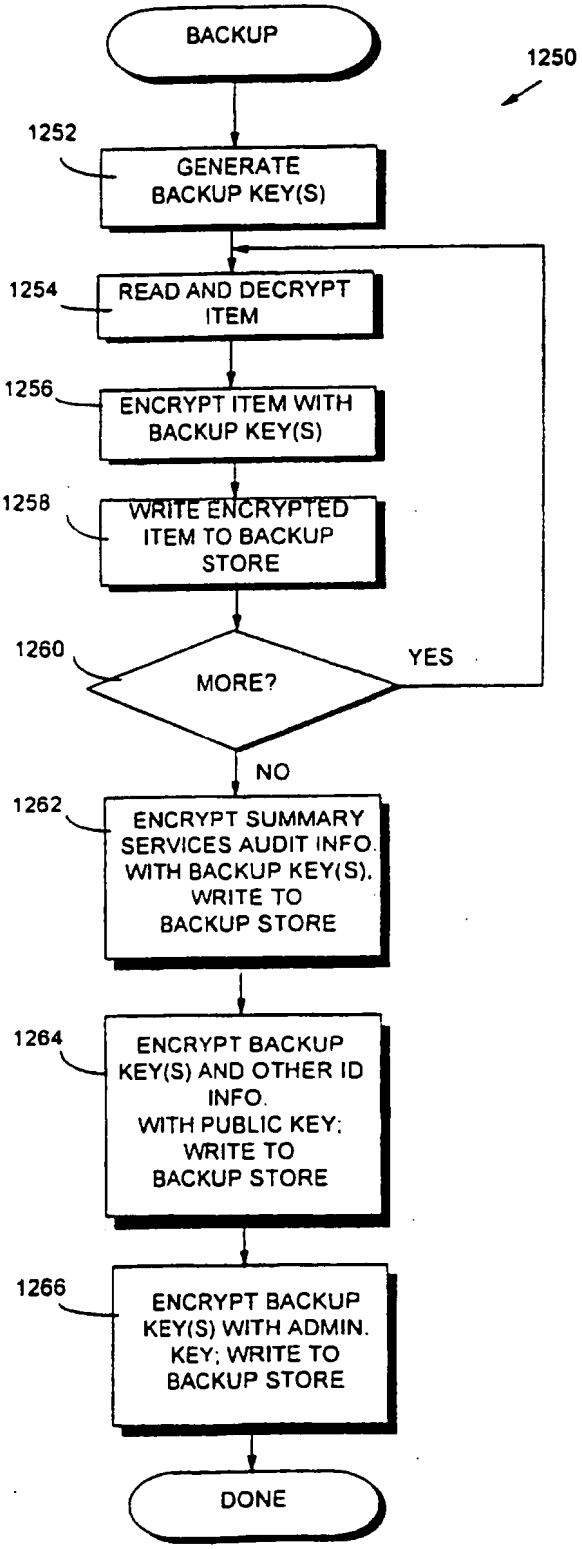
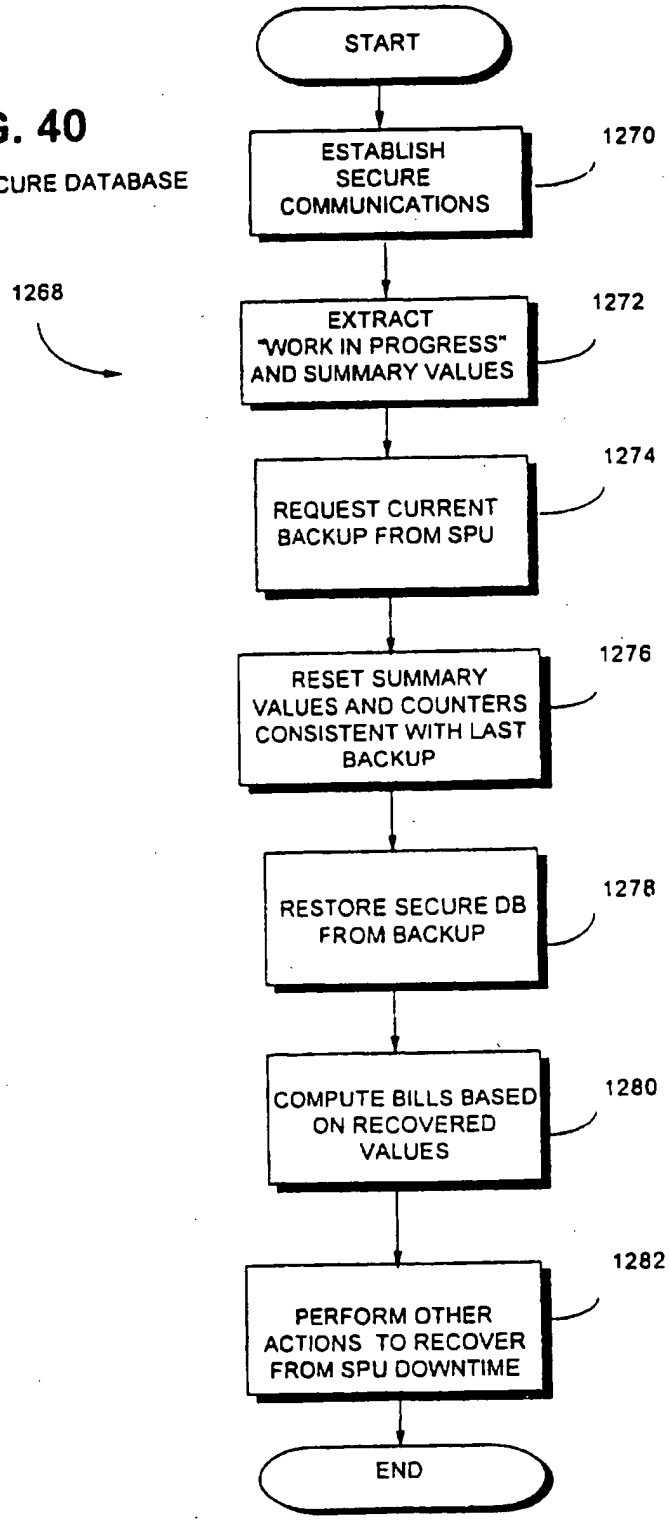


FIG. 40

RECOVER SECURE DATABASE



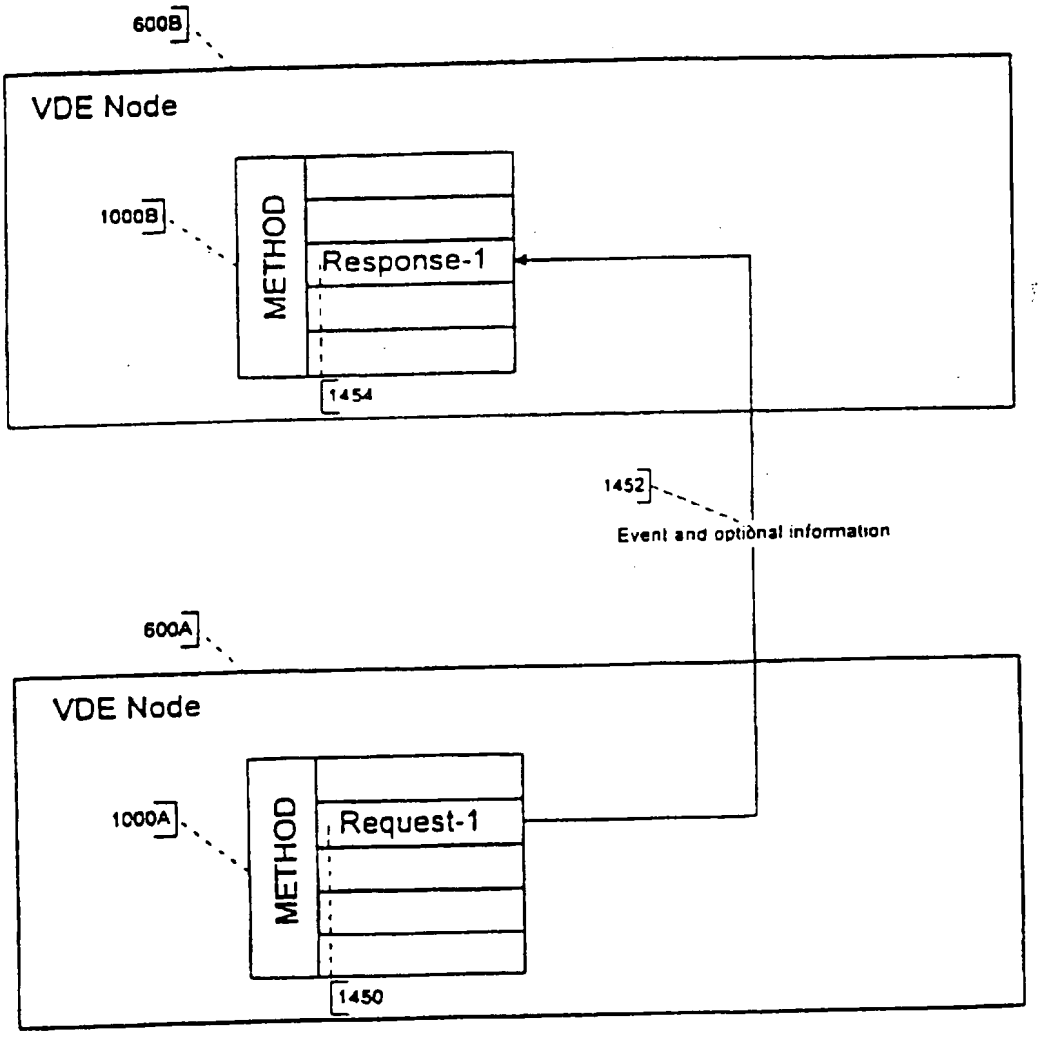


Figure 41a

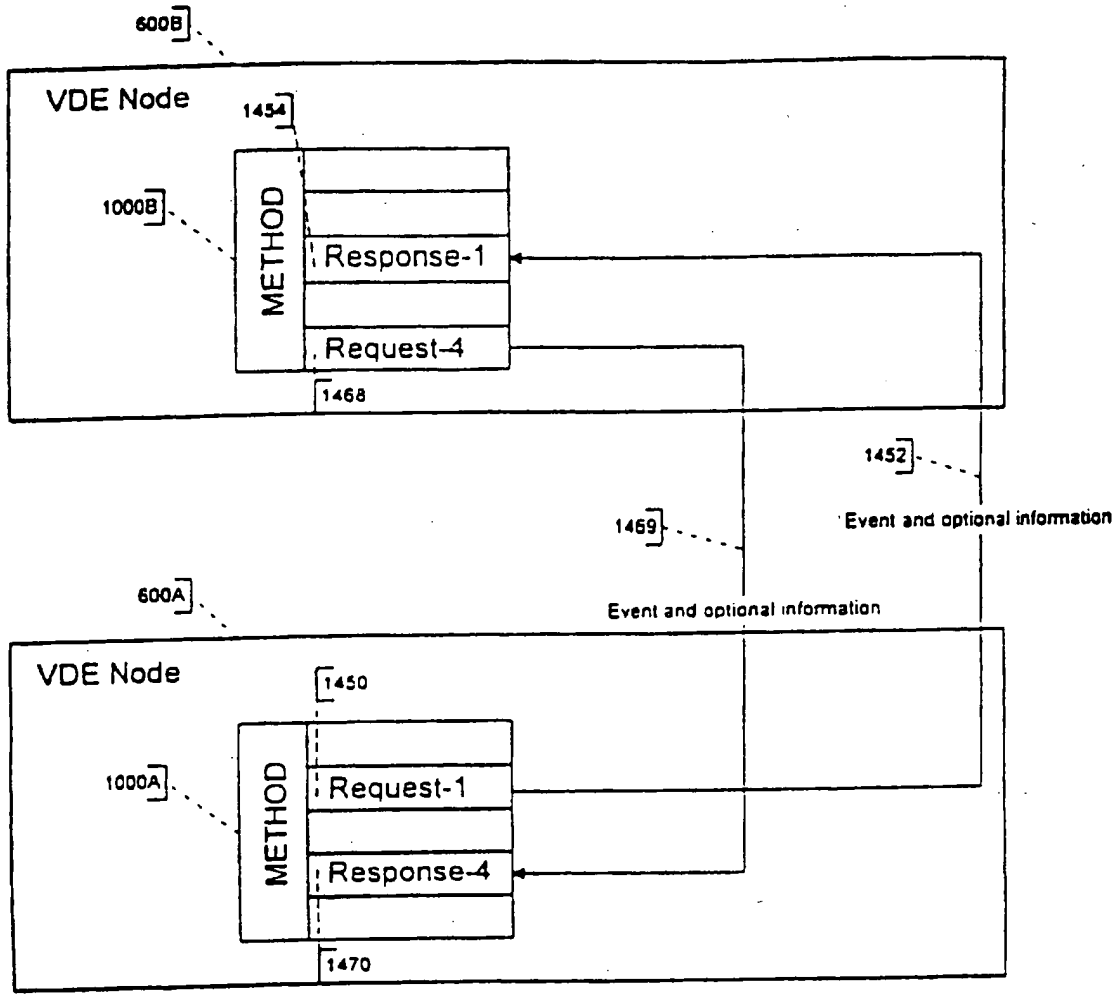


Figure 41b

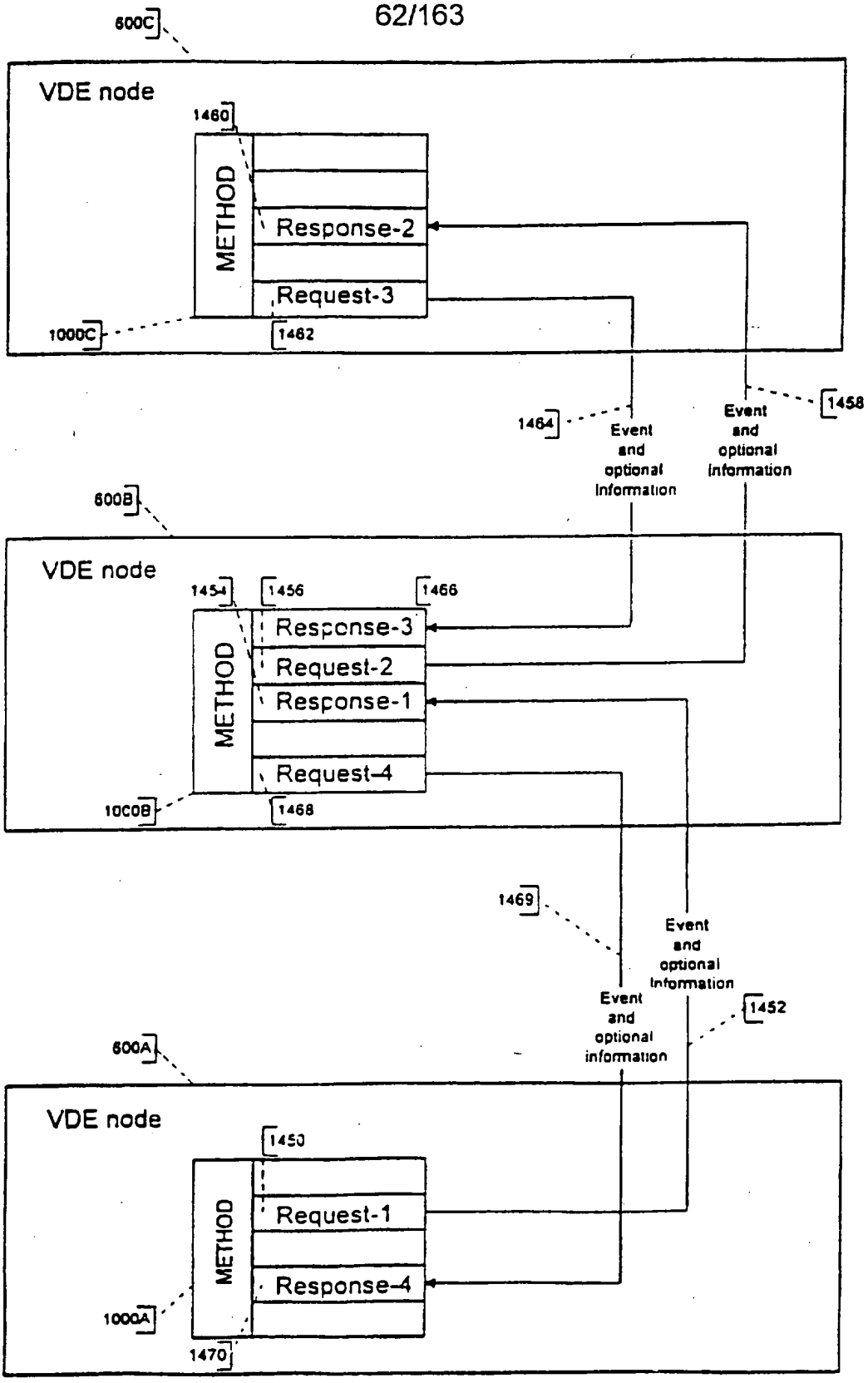


Figure 41c

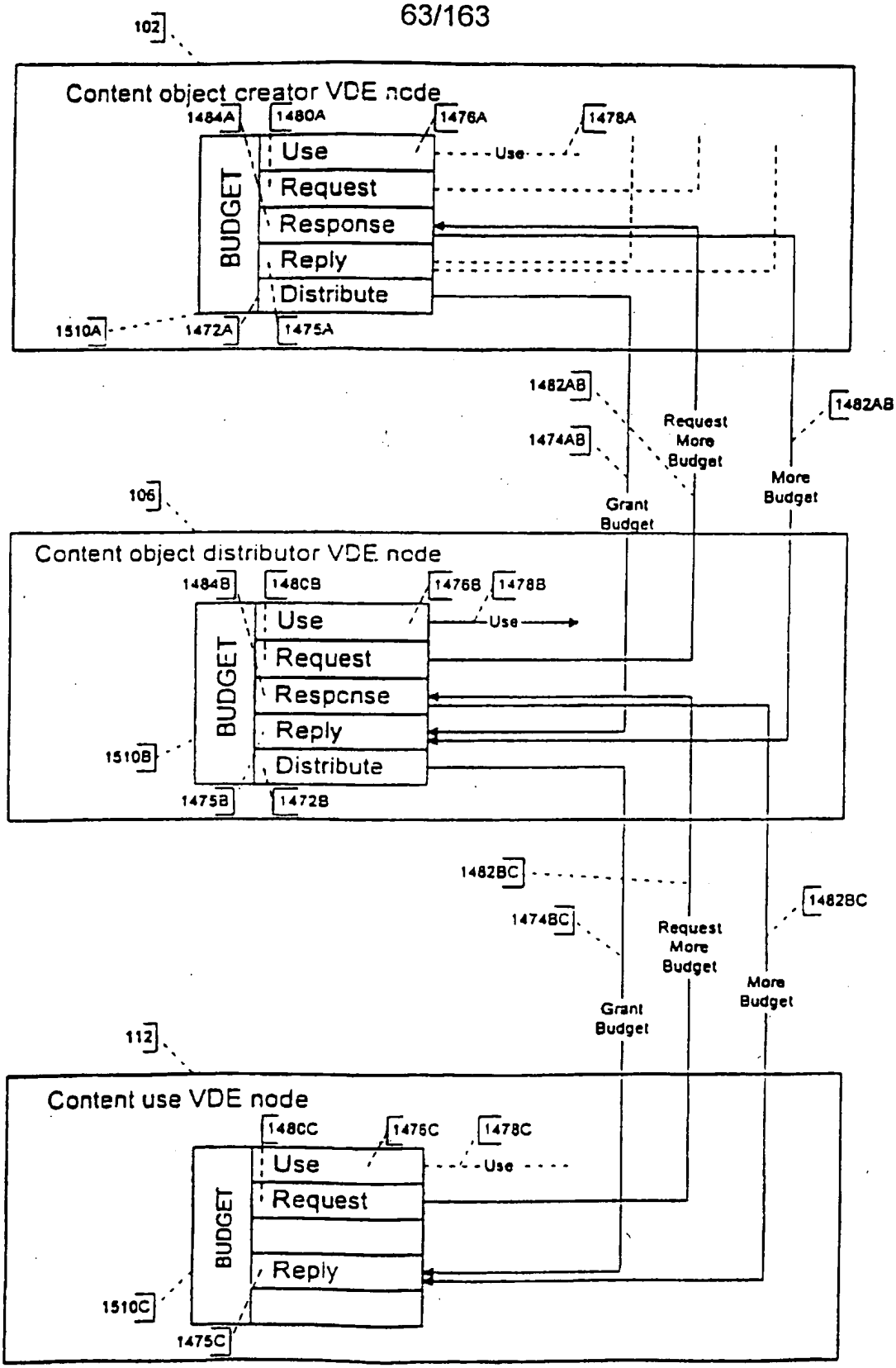


Figure 41d

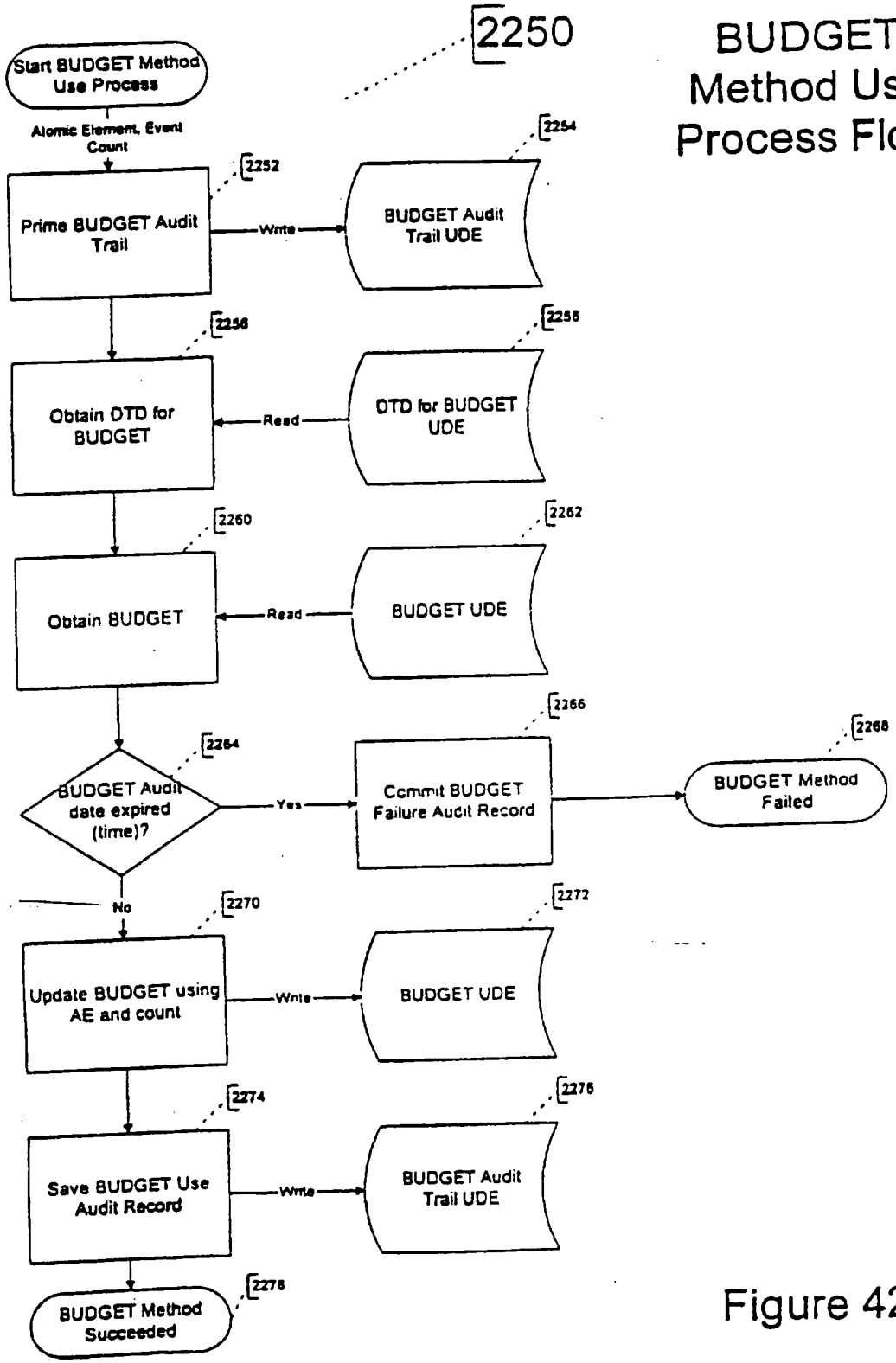


Figure 42a

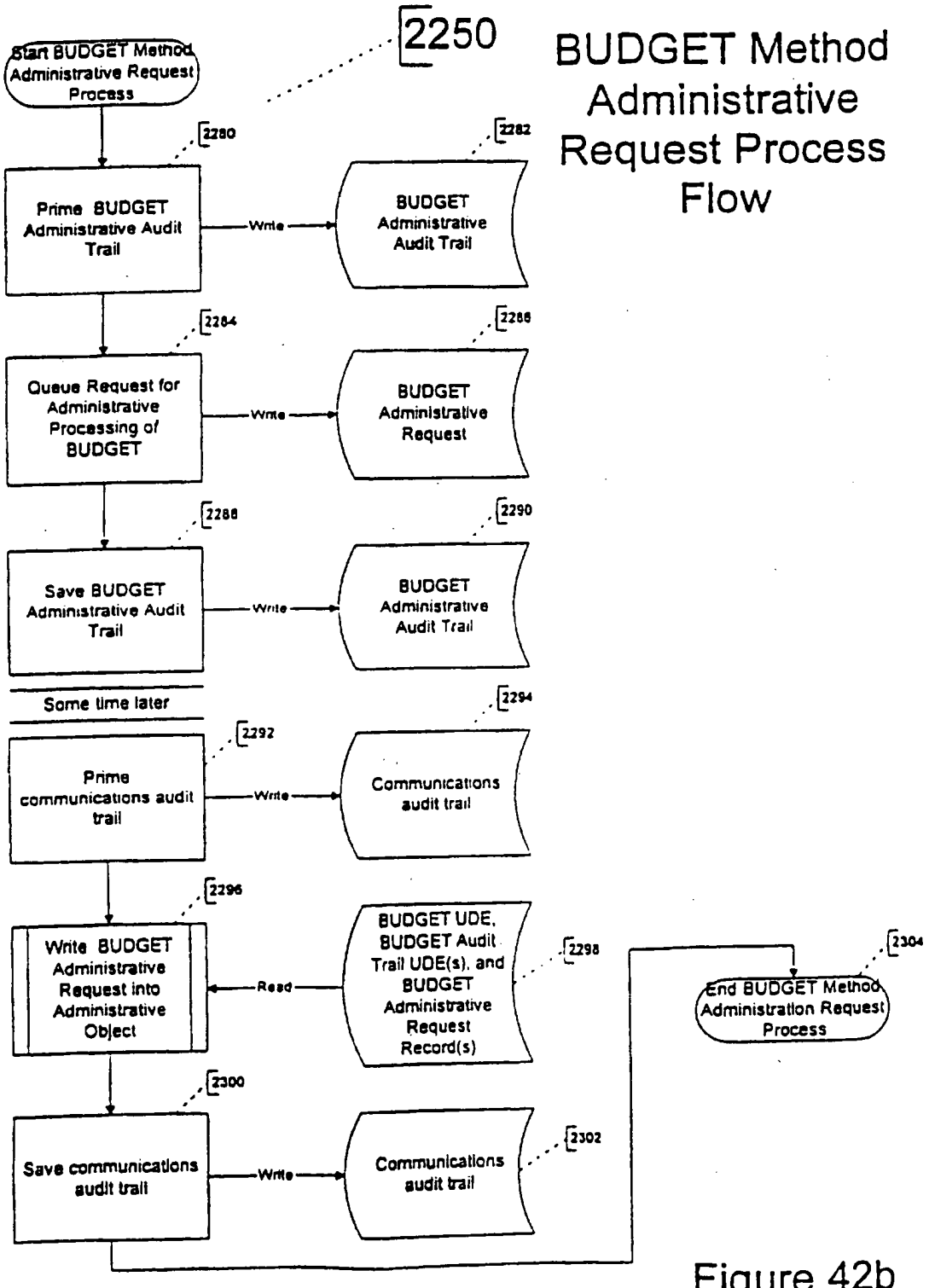
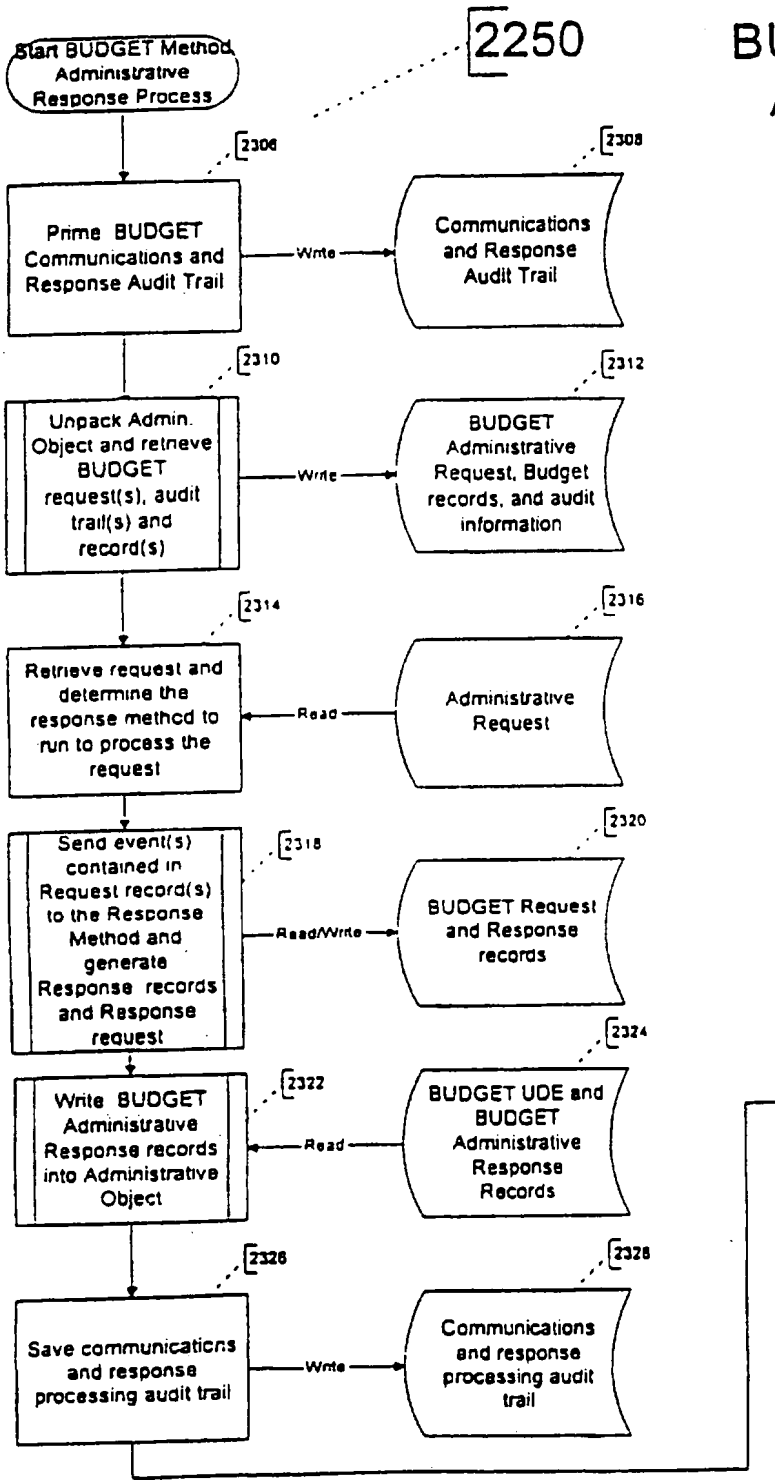


Figure 42b



BUDGET Method Administrative Response Process Flow

Figure 42c

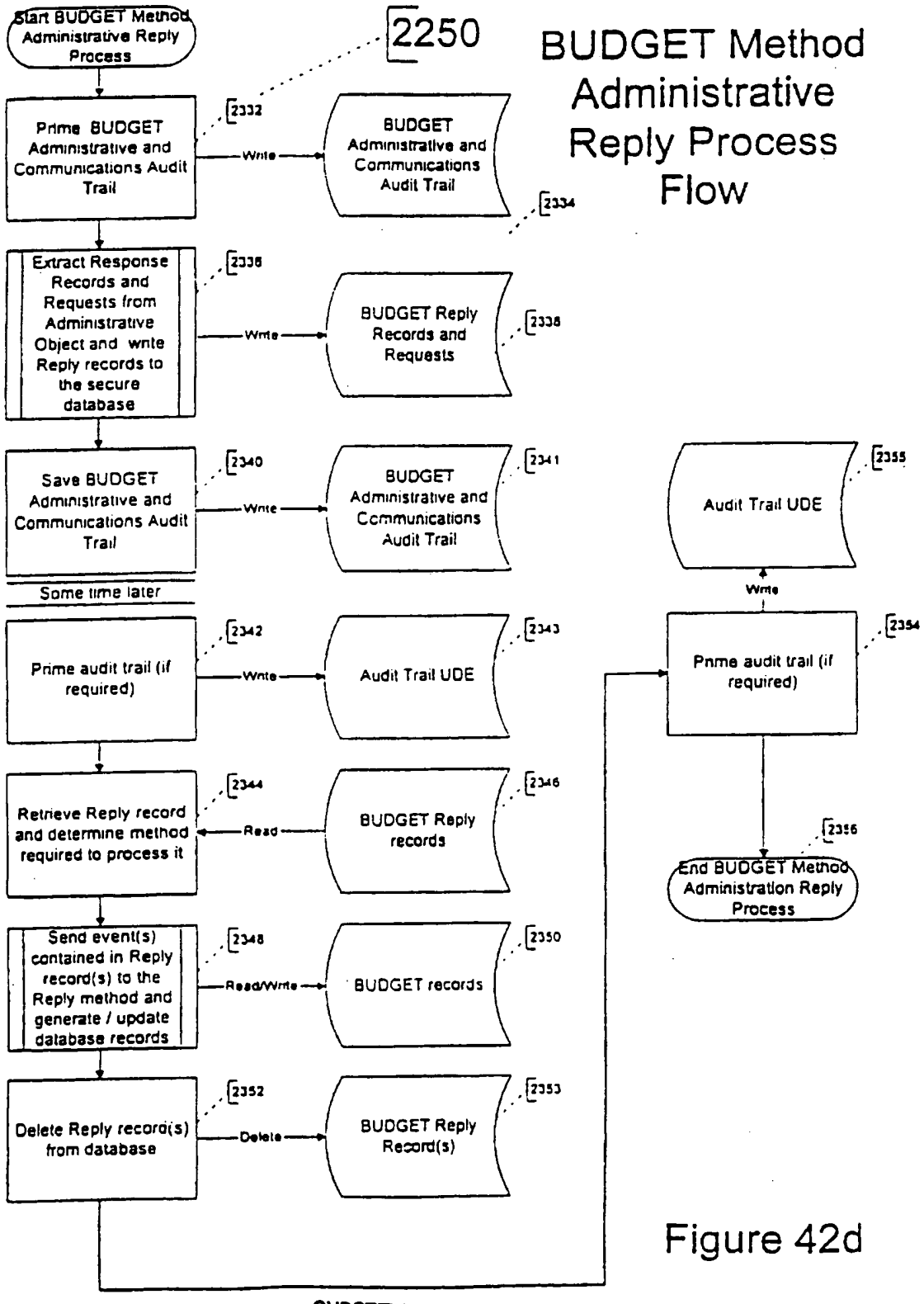


Figure 42d

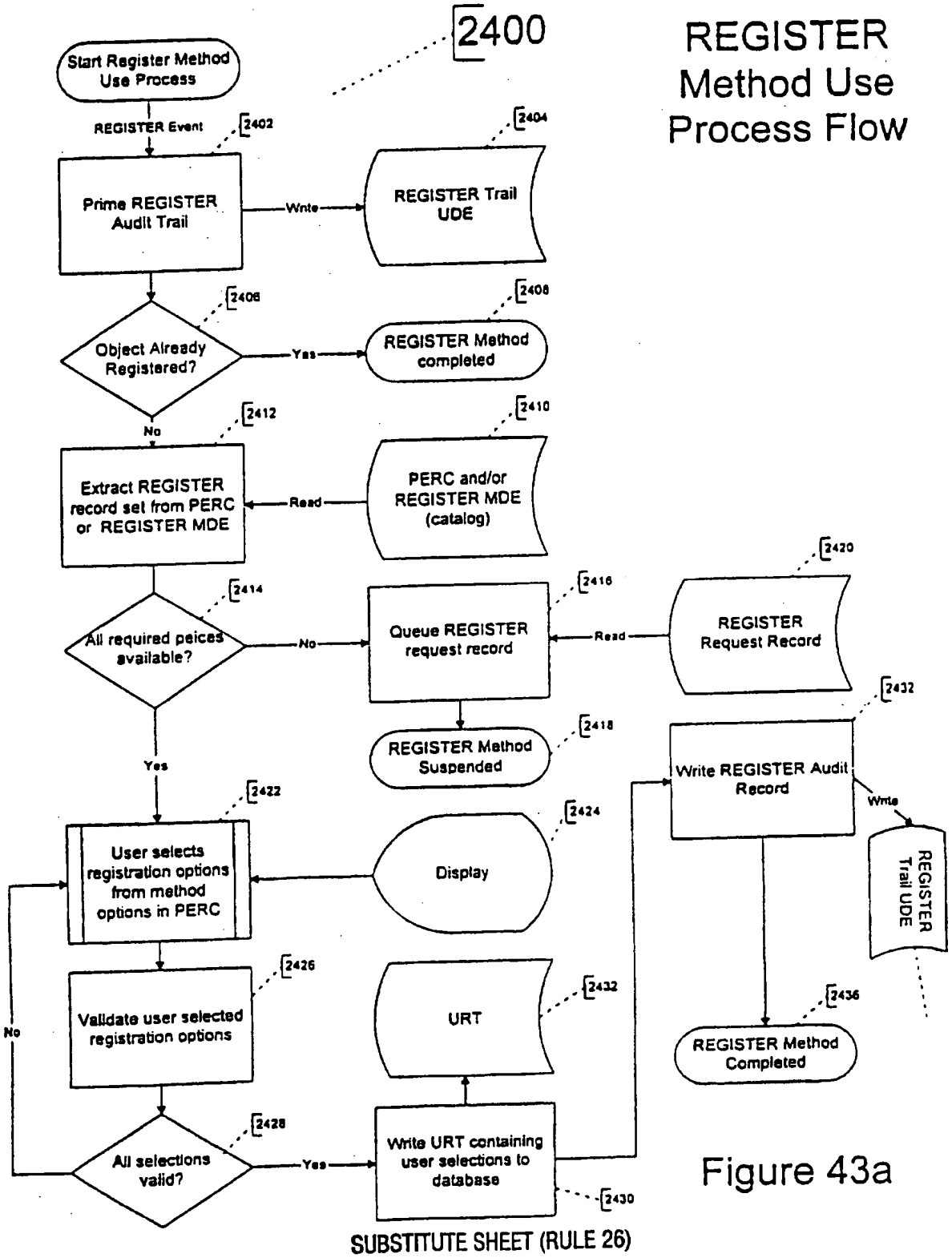
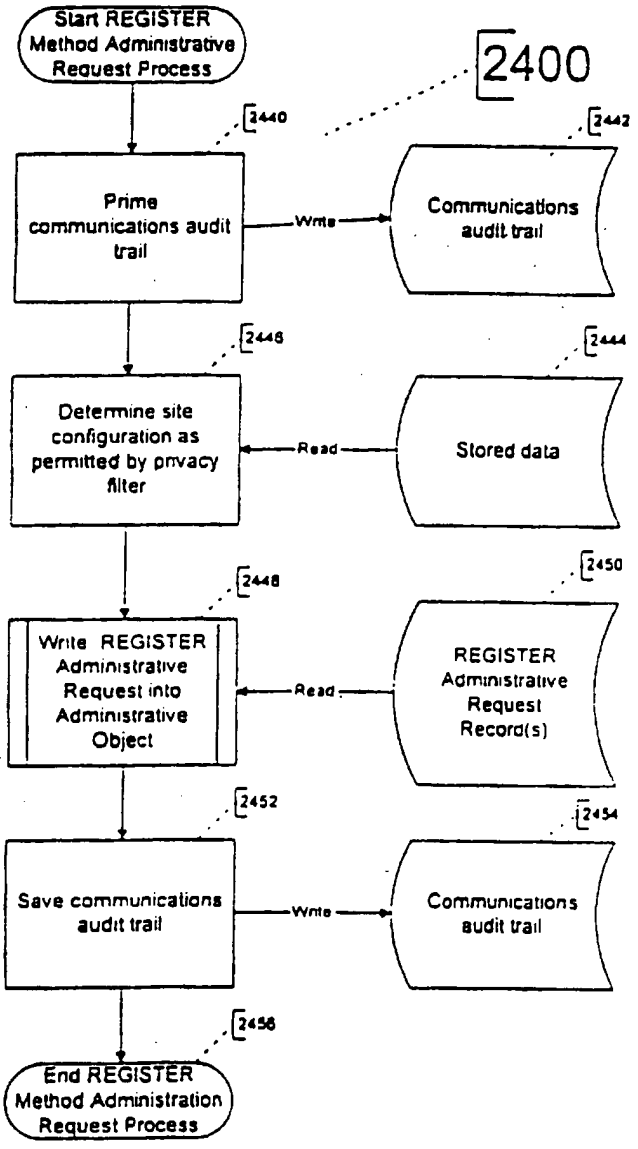
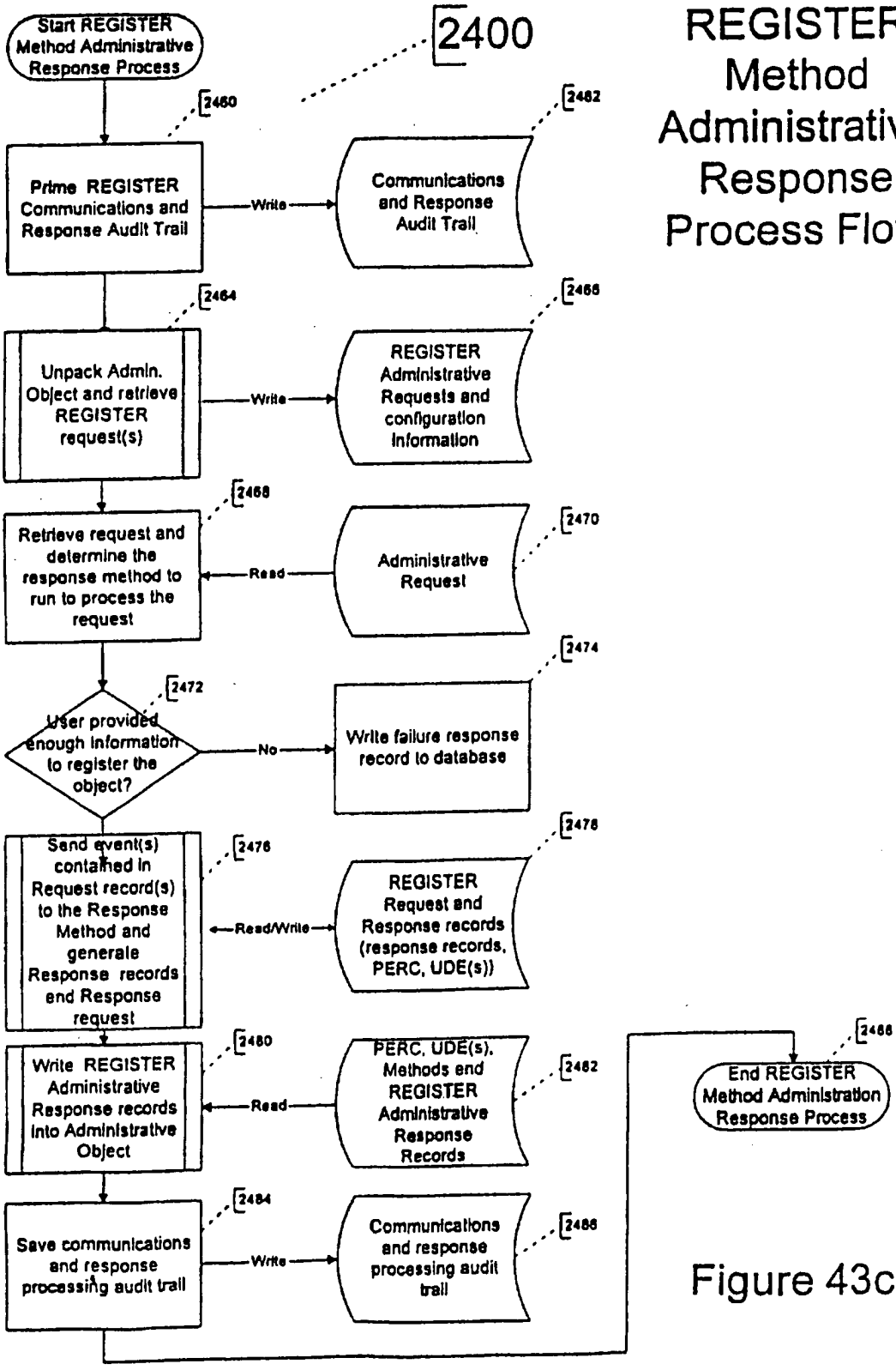


Figure 43a



REGISTER Method Administrative Request Process Flow

Figure 43b



REGISTER Method Administrative Response Process Flow

Figure 43c

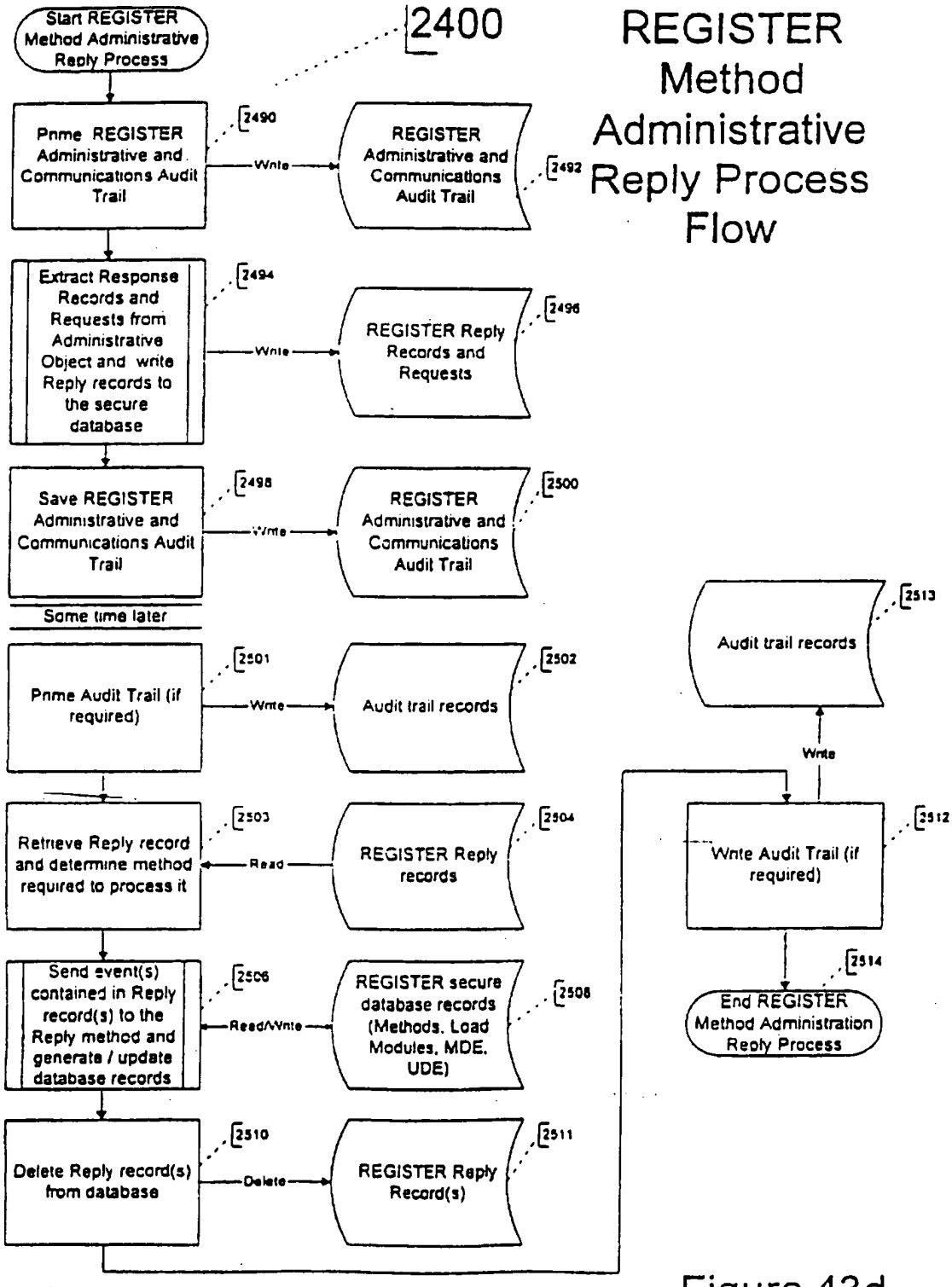


Figure 43d

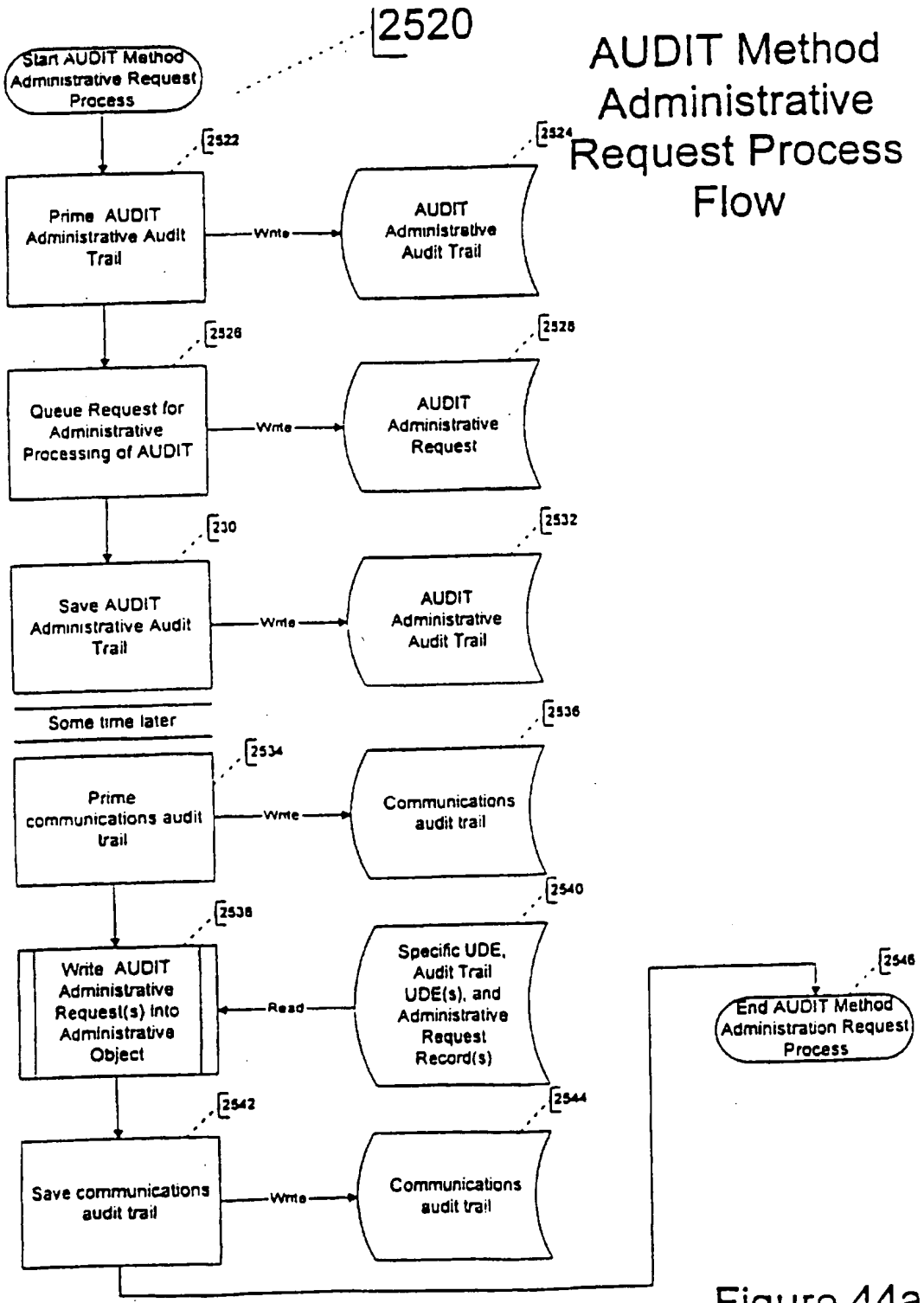


Figure 44a

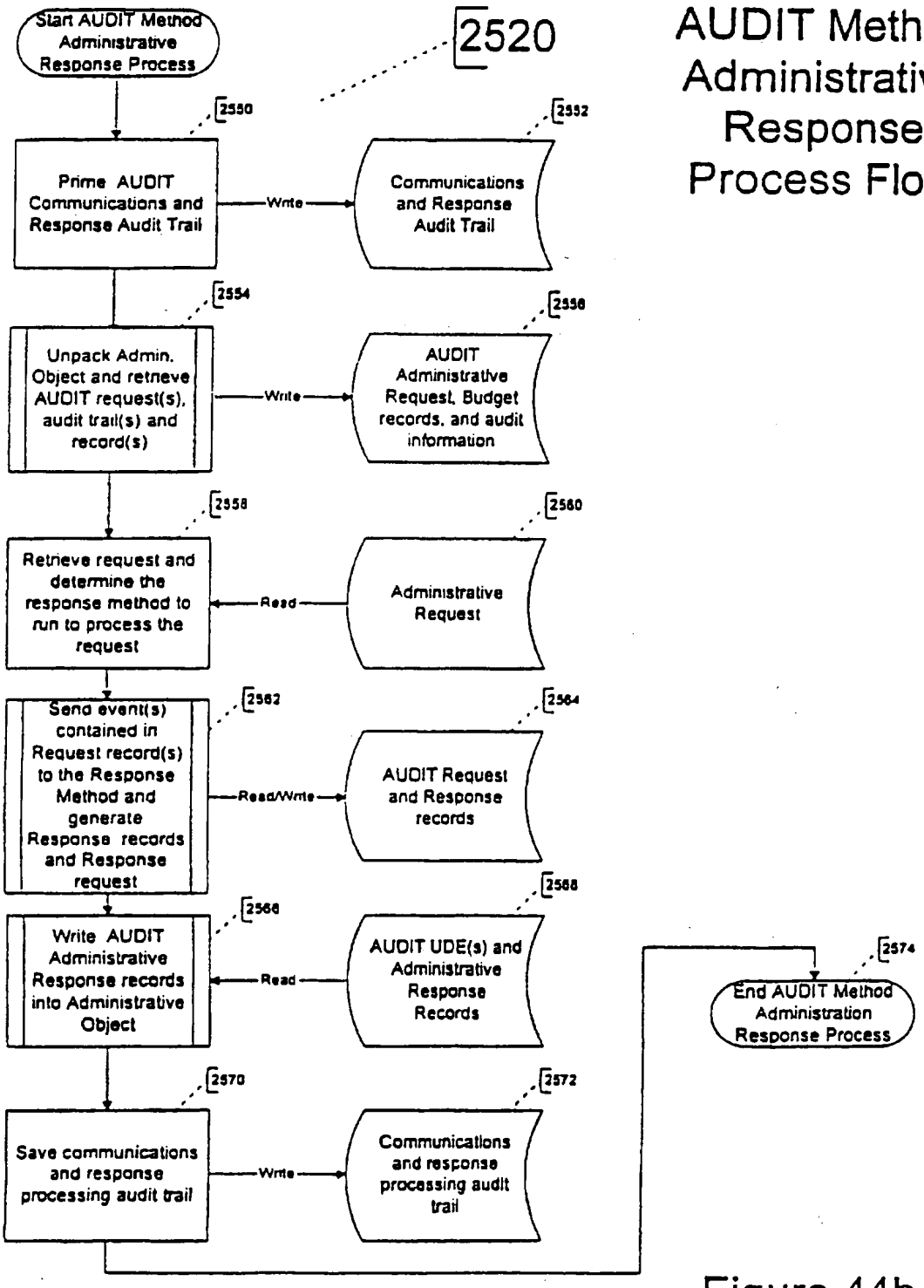


Figure 44b

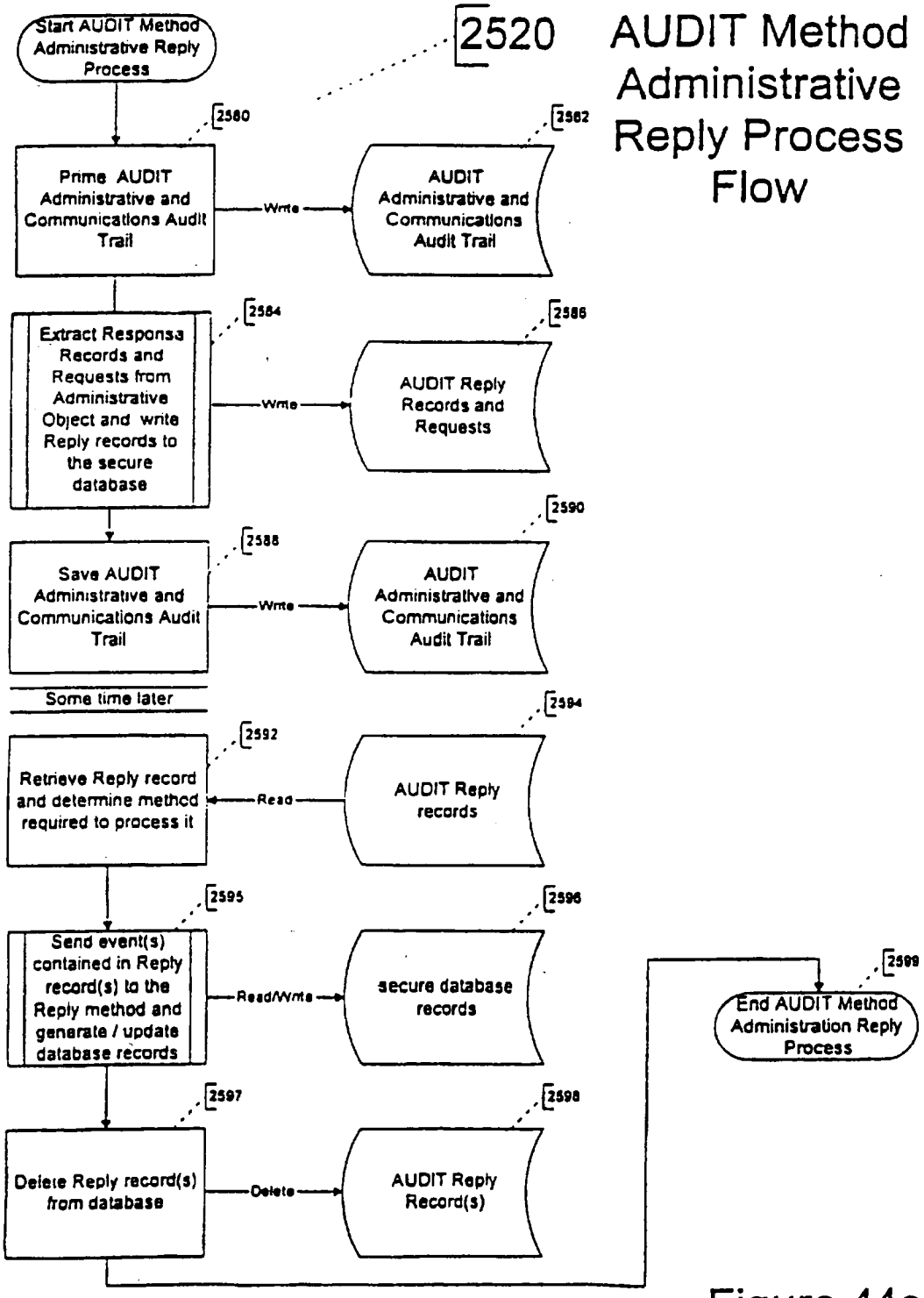


Figure 44c

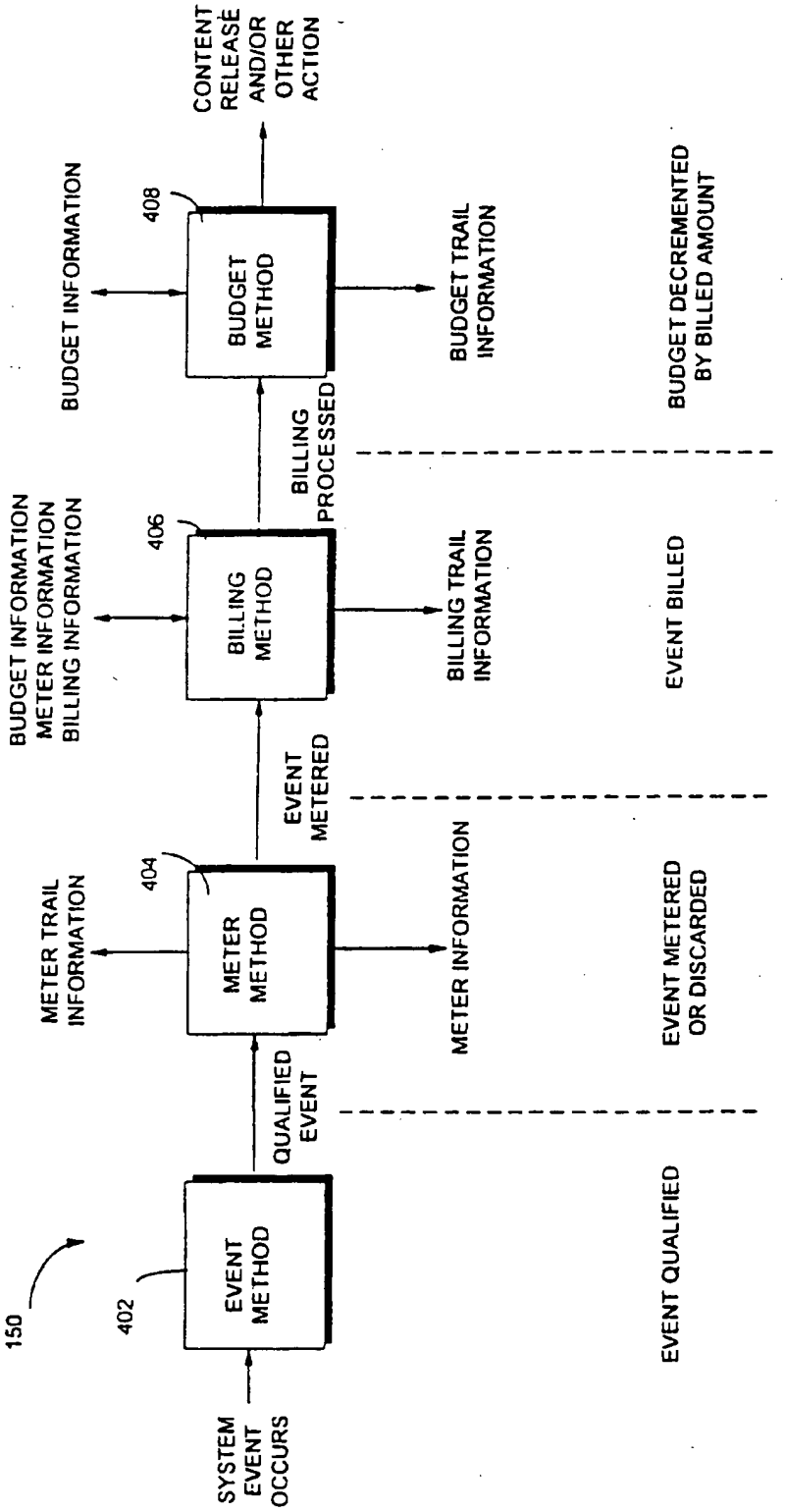
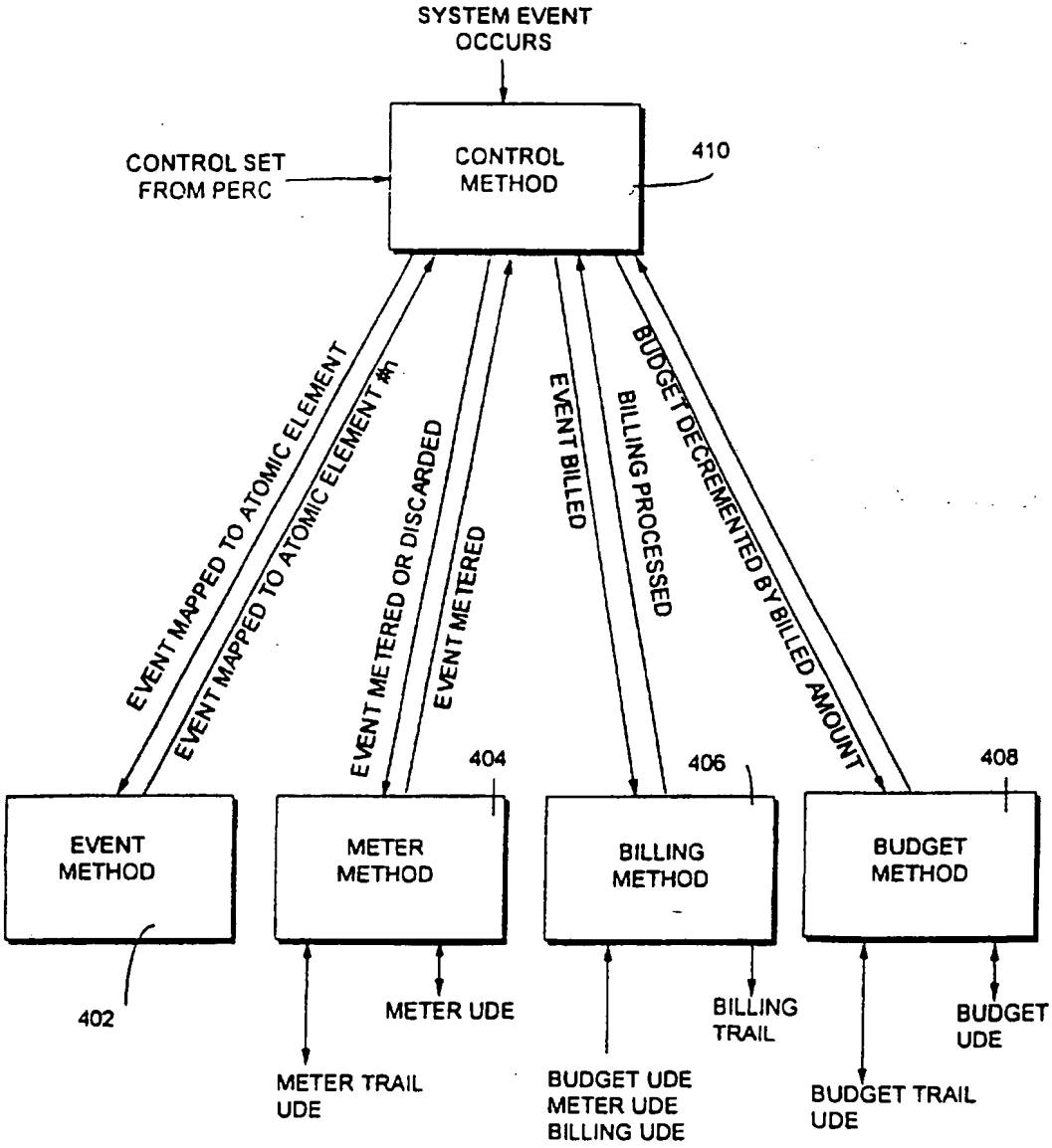


FIG. 45

FIG. 46



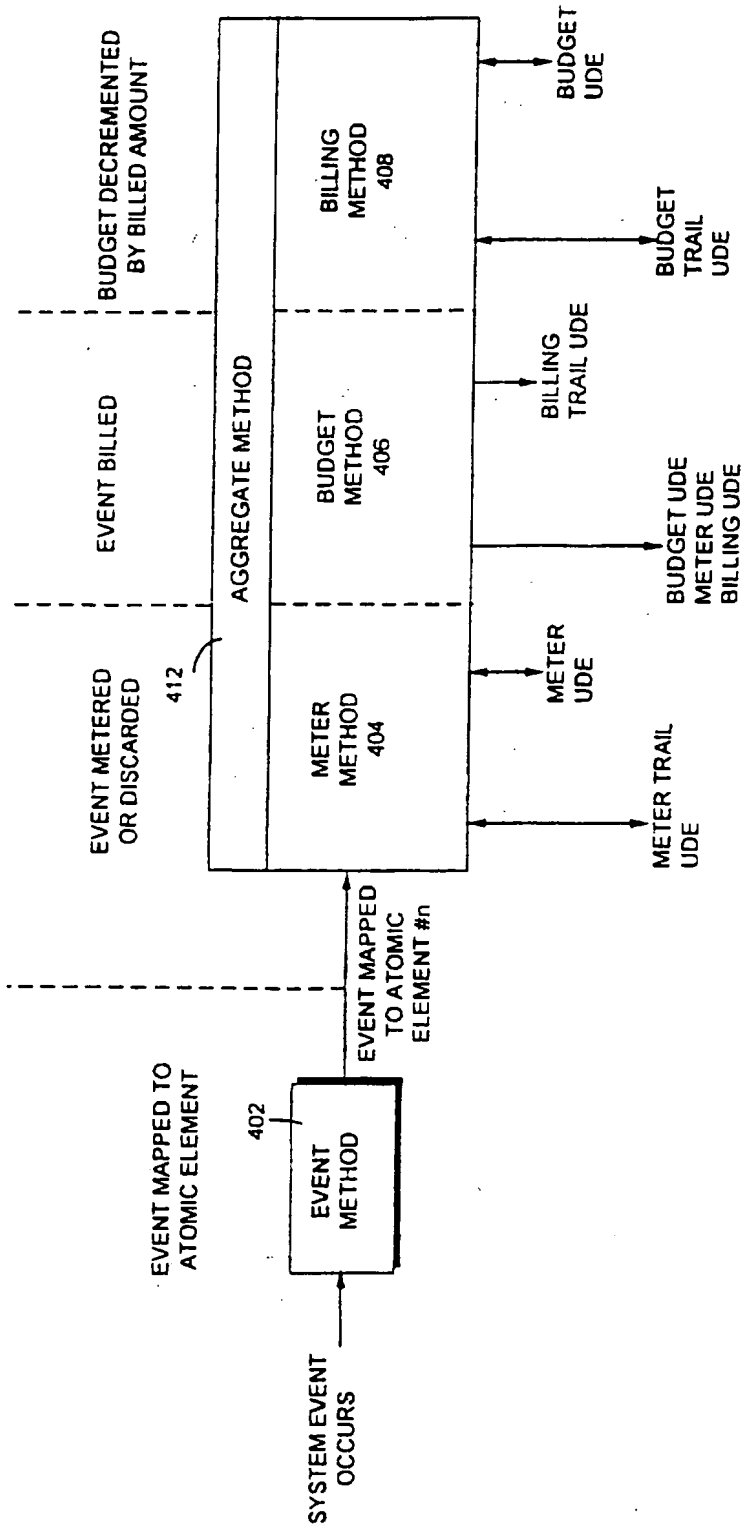


FIG. 47

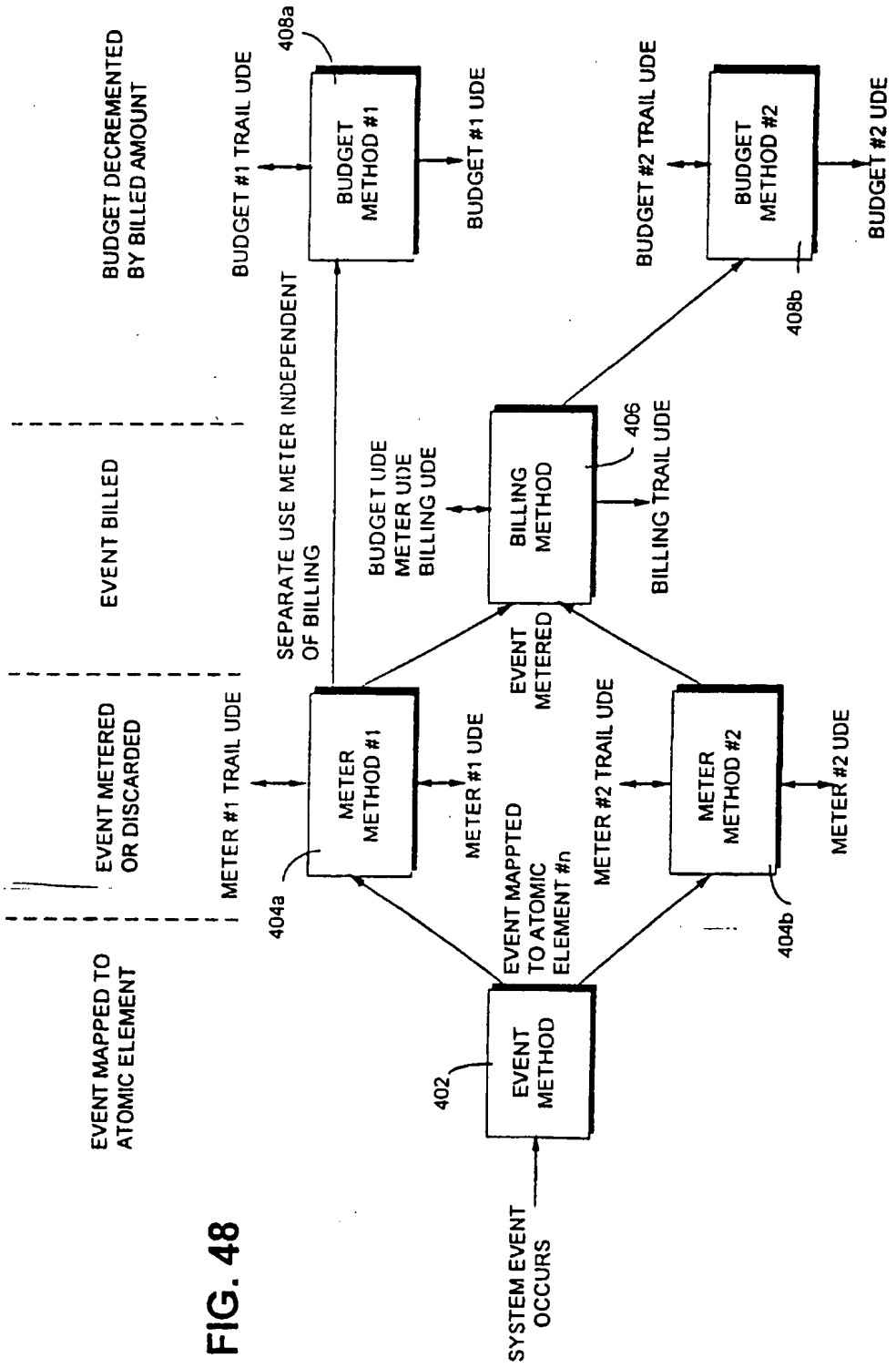


FIG. 48

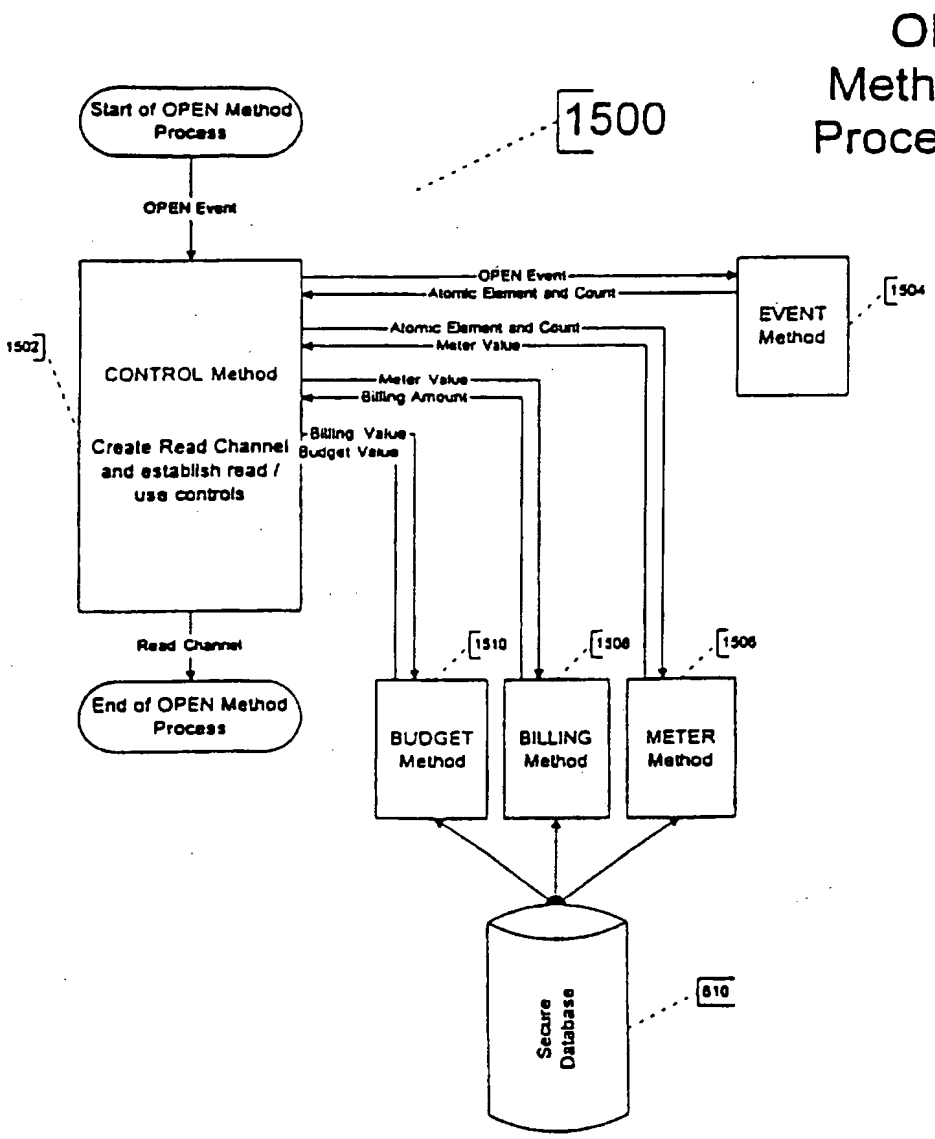


Figure 49

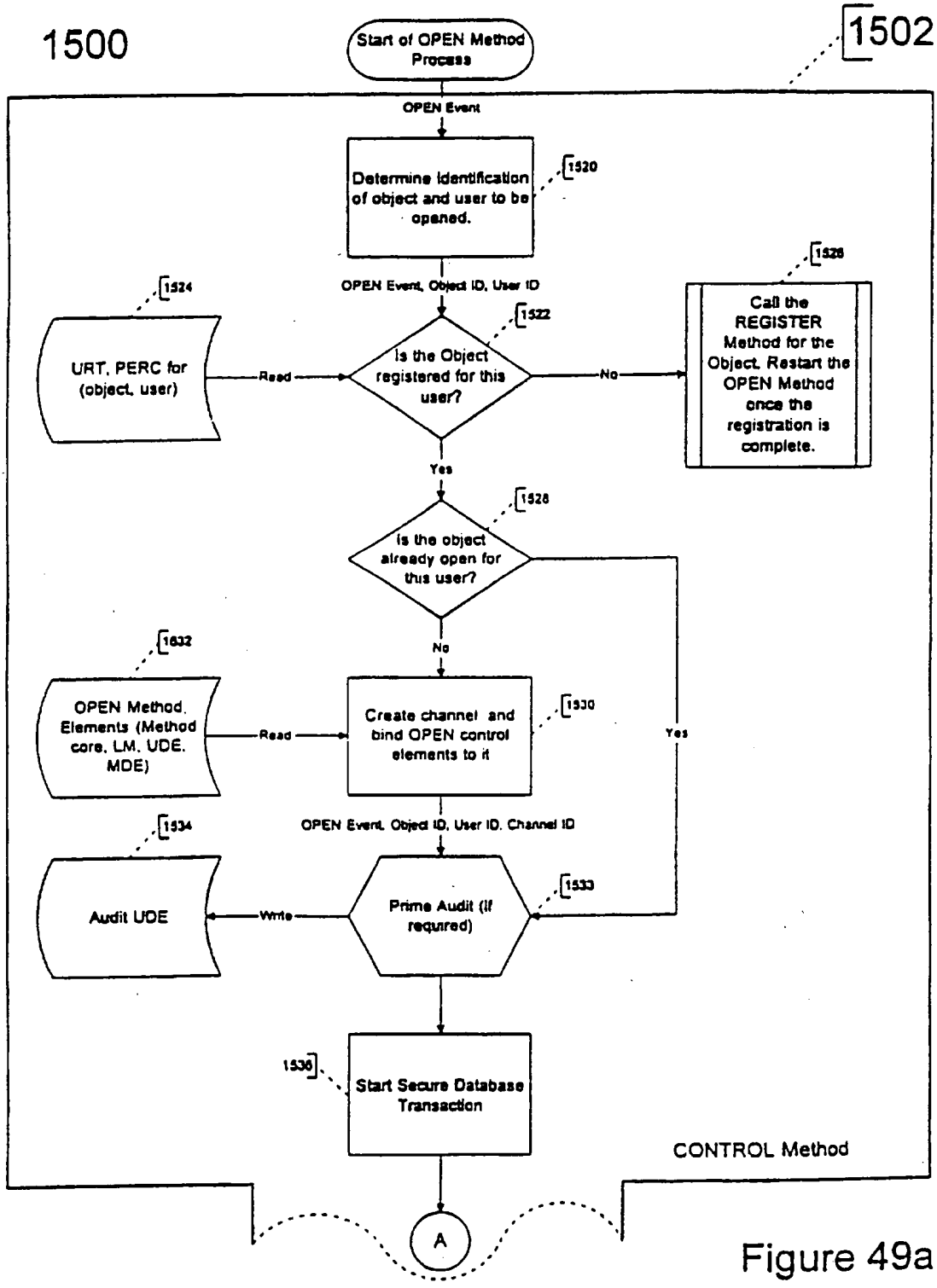


Figure 49a

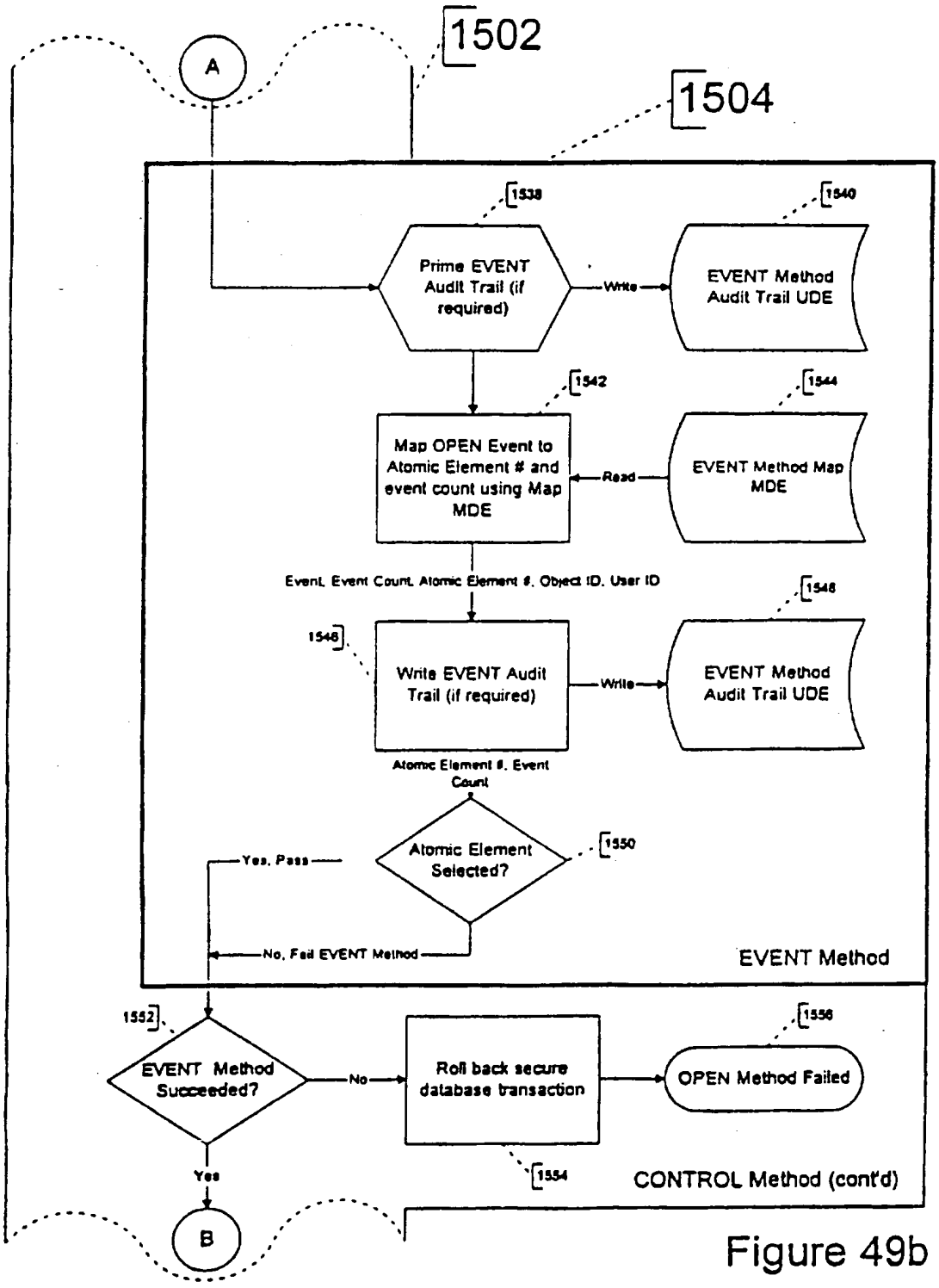


Figure 49b

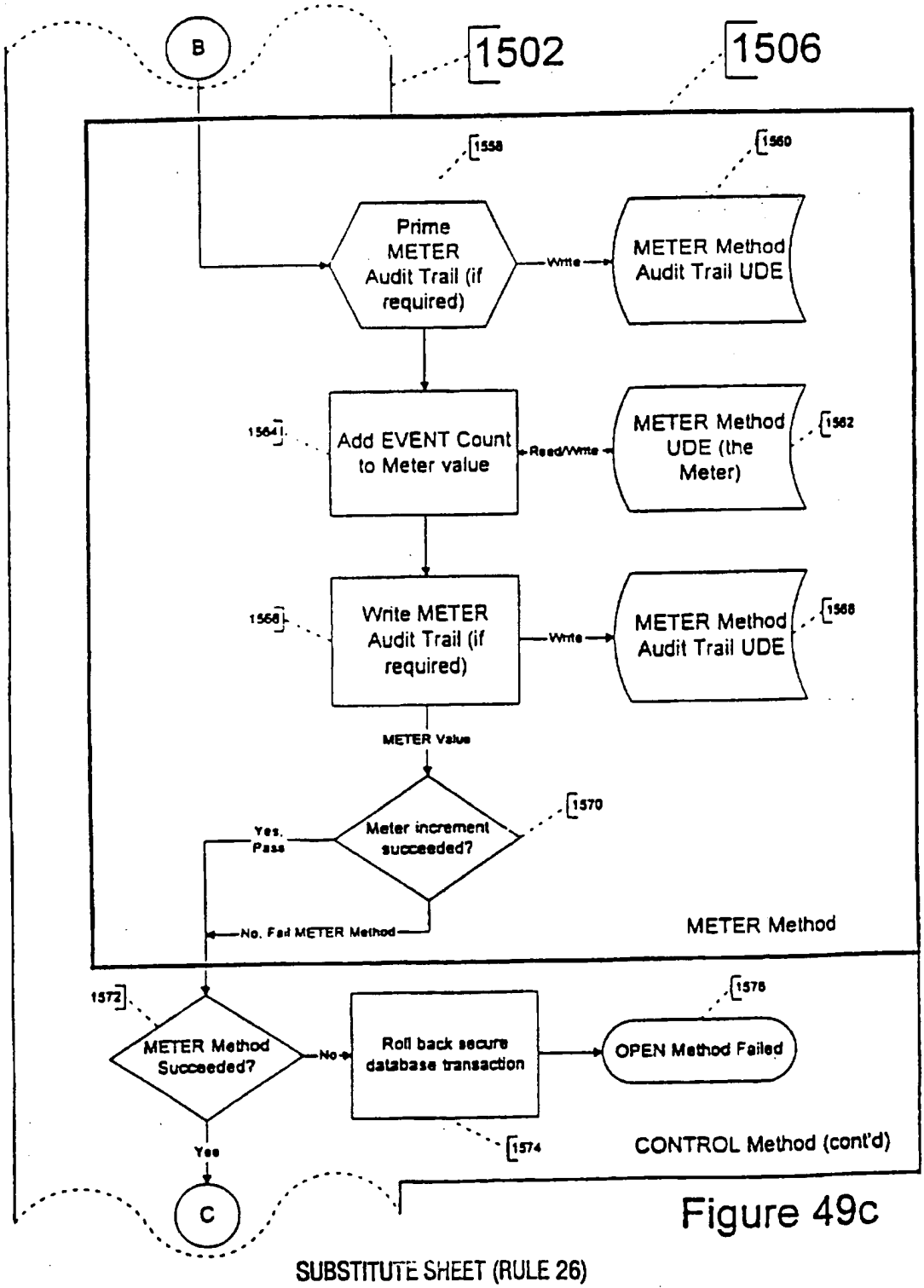


Figure 49c

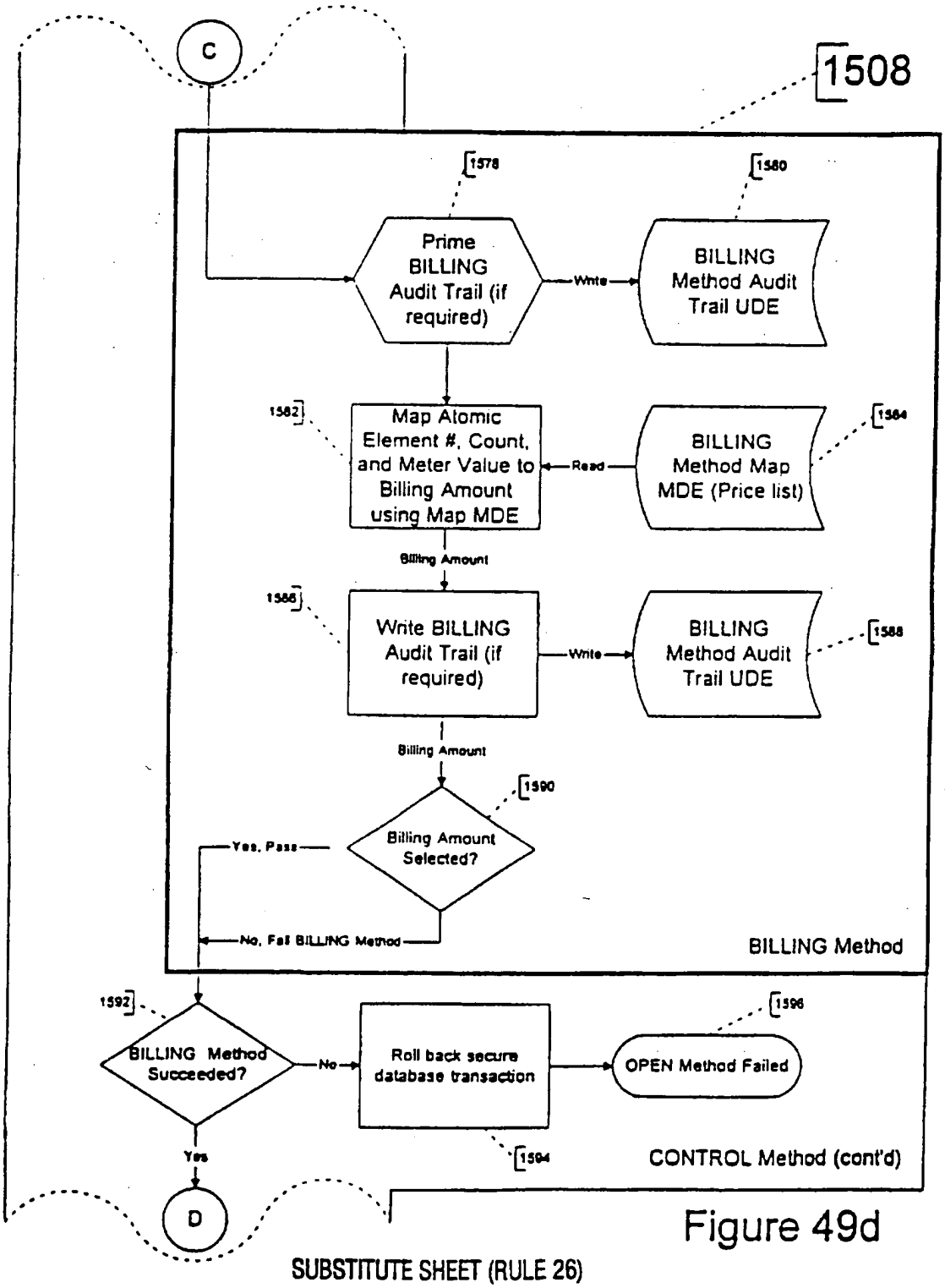


Figure 49d

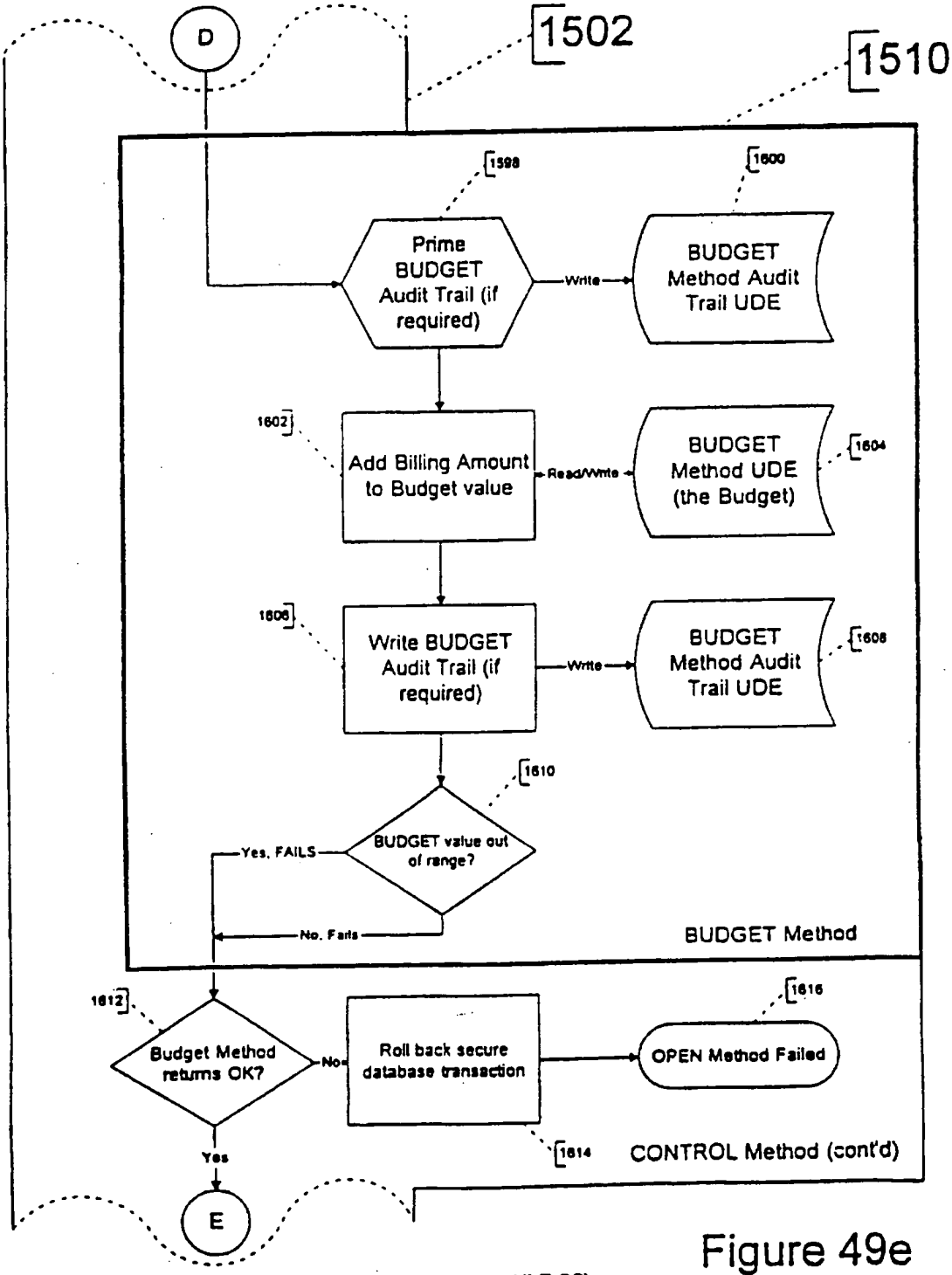


Figure 49e

SUBSTITUTE SHEET (RULE 26)

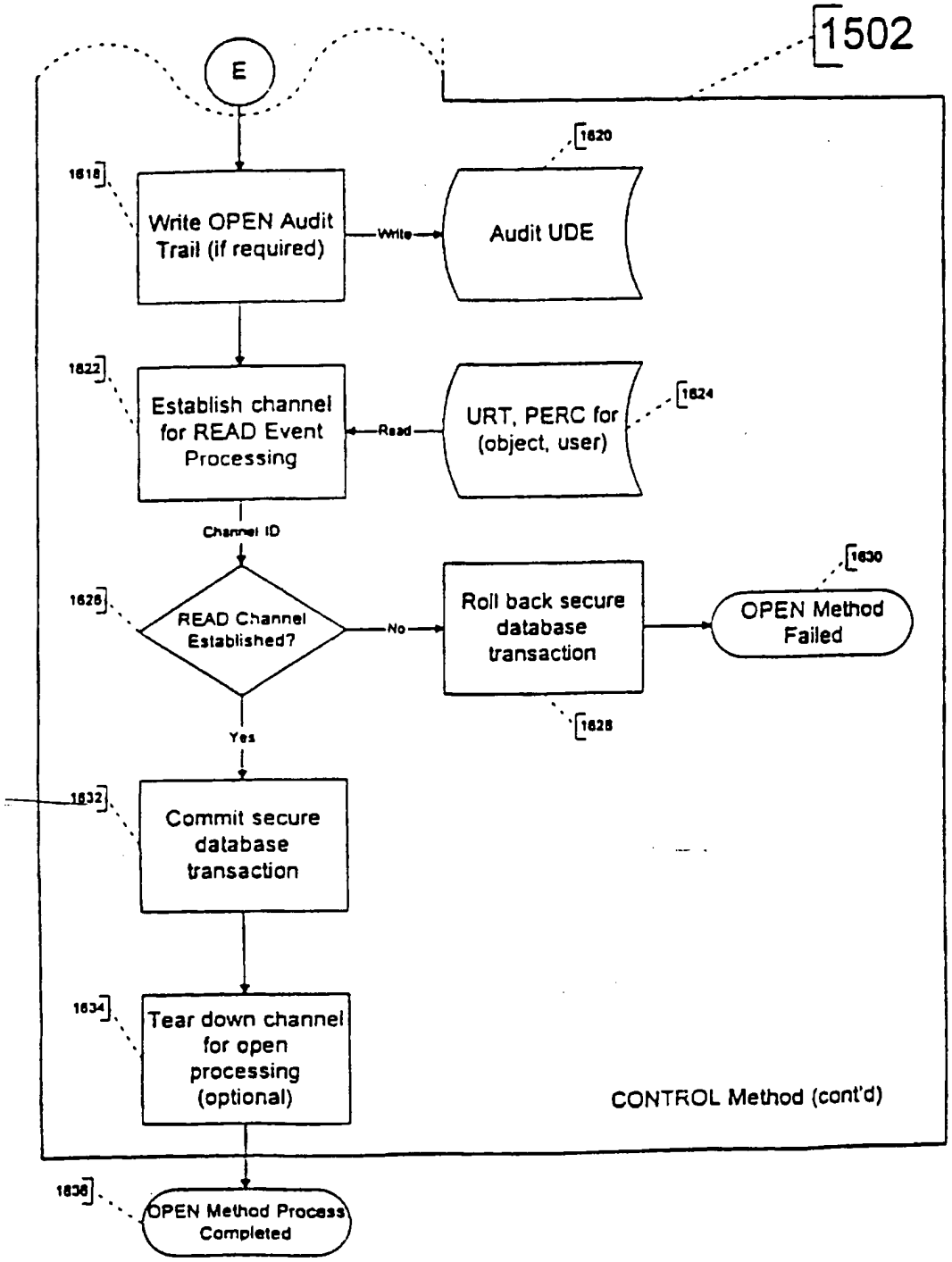


Figure 49f

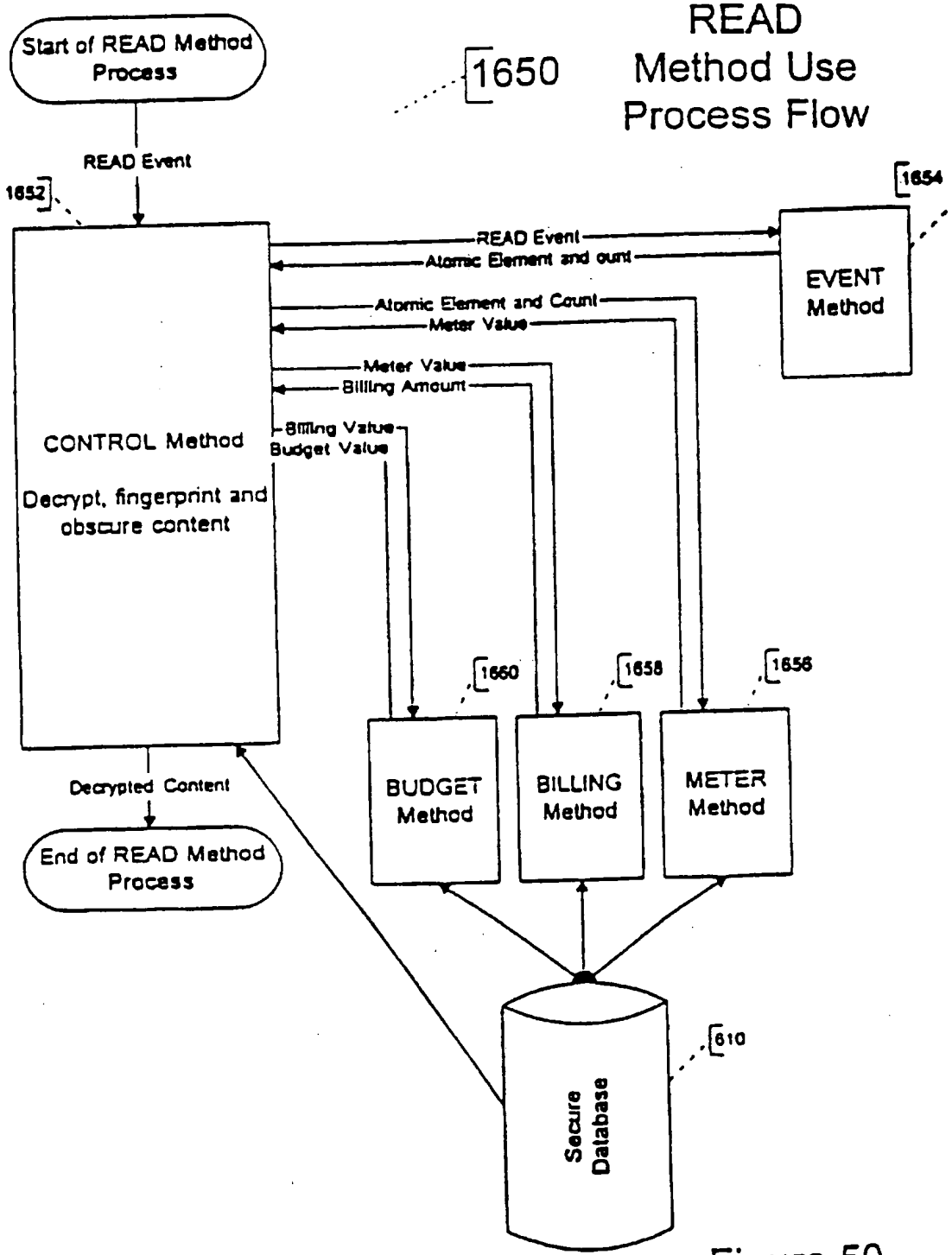


Figure 50

1650

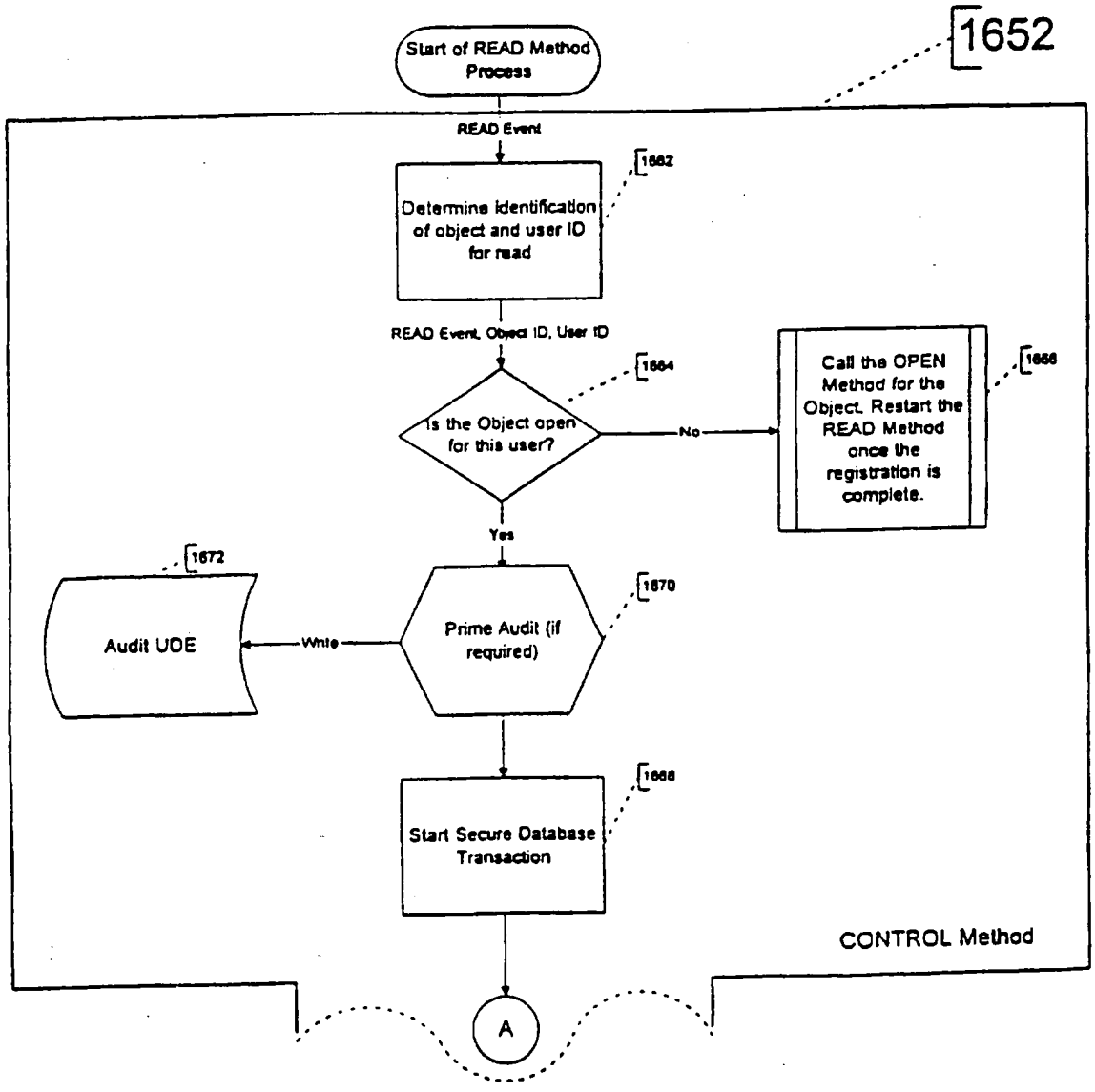


Figure 50a

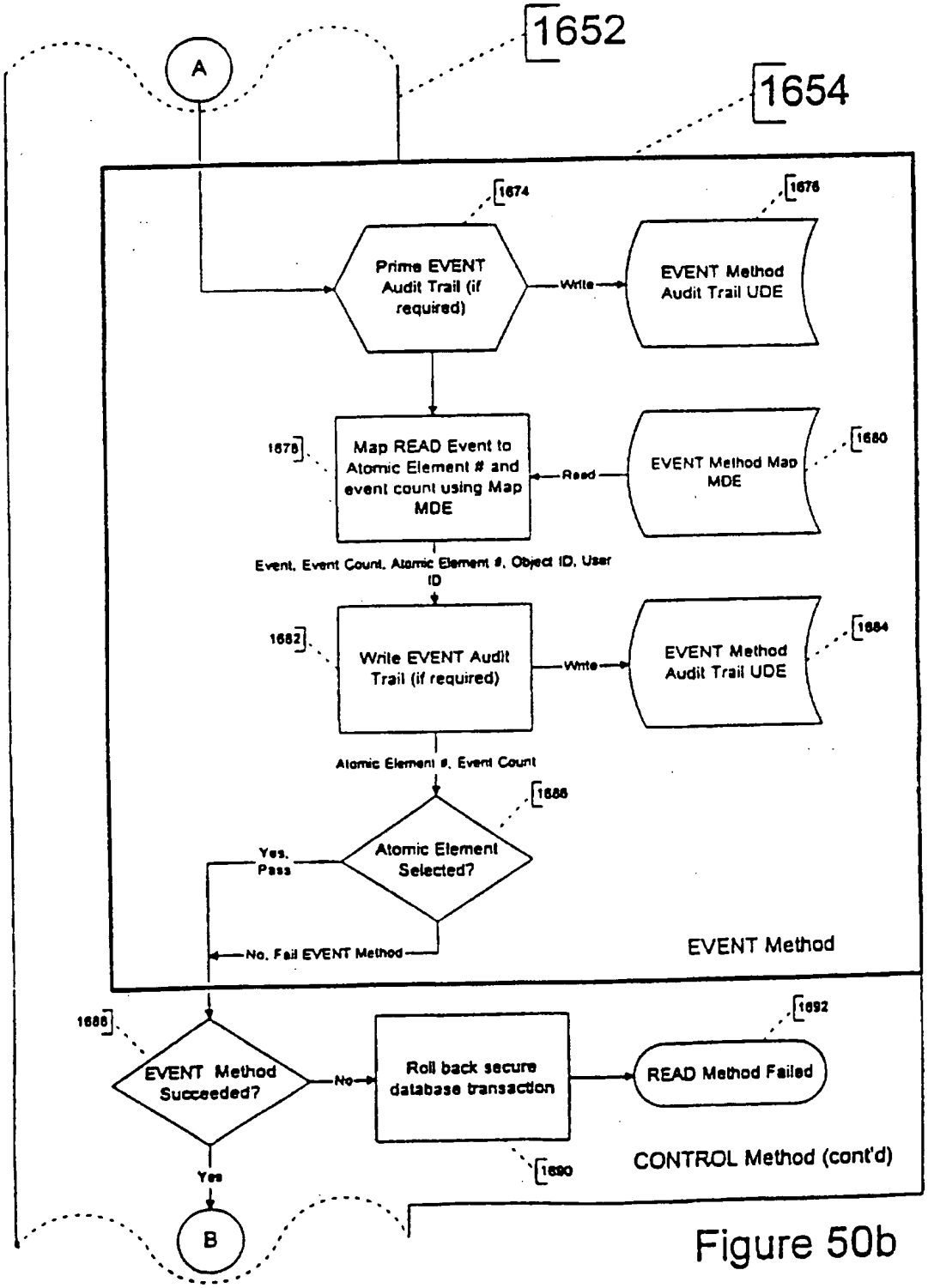


Figure 50b

SUBSTITUTE SHEET (RULE 26)

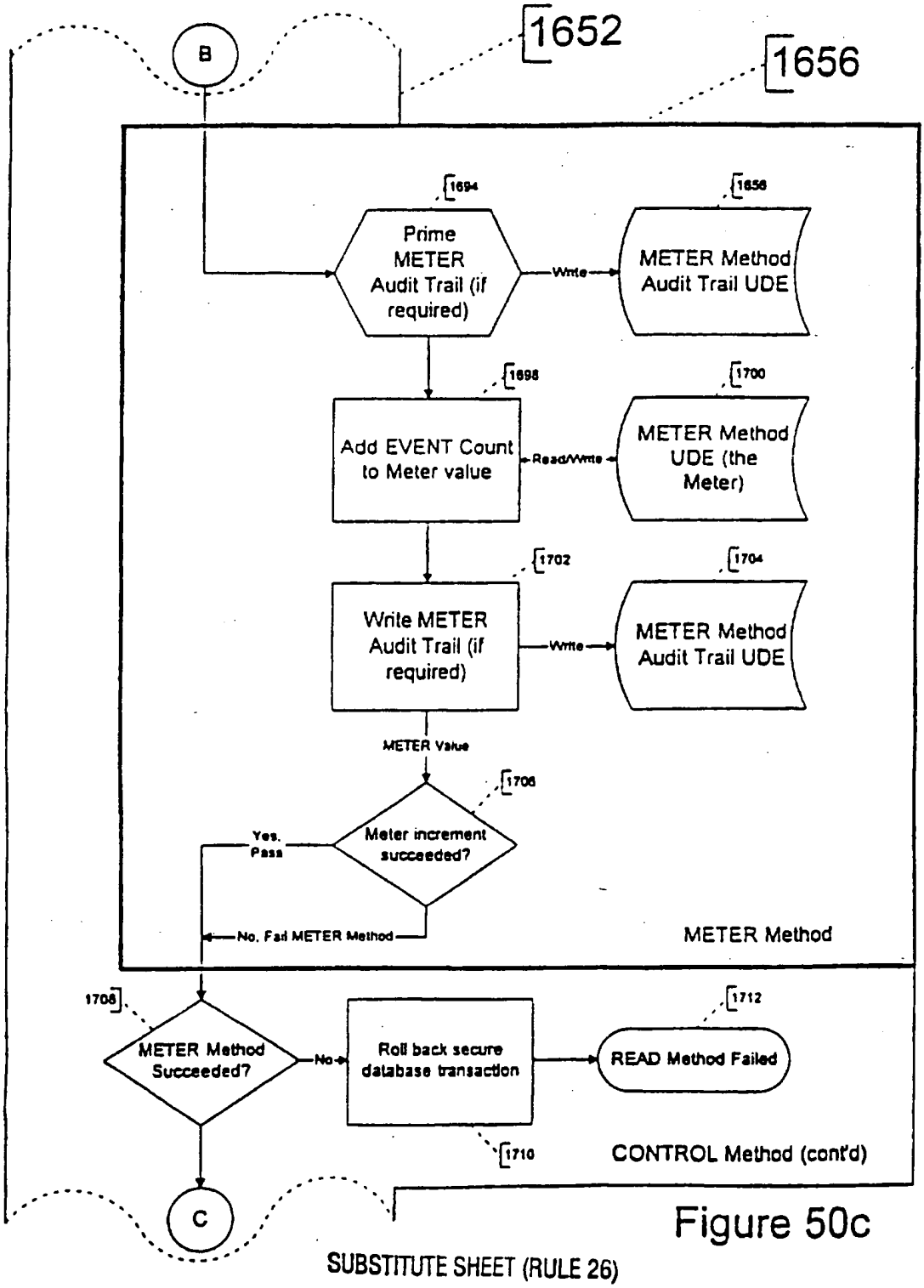


Figure 50c

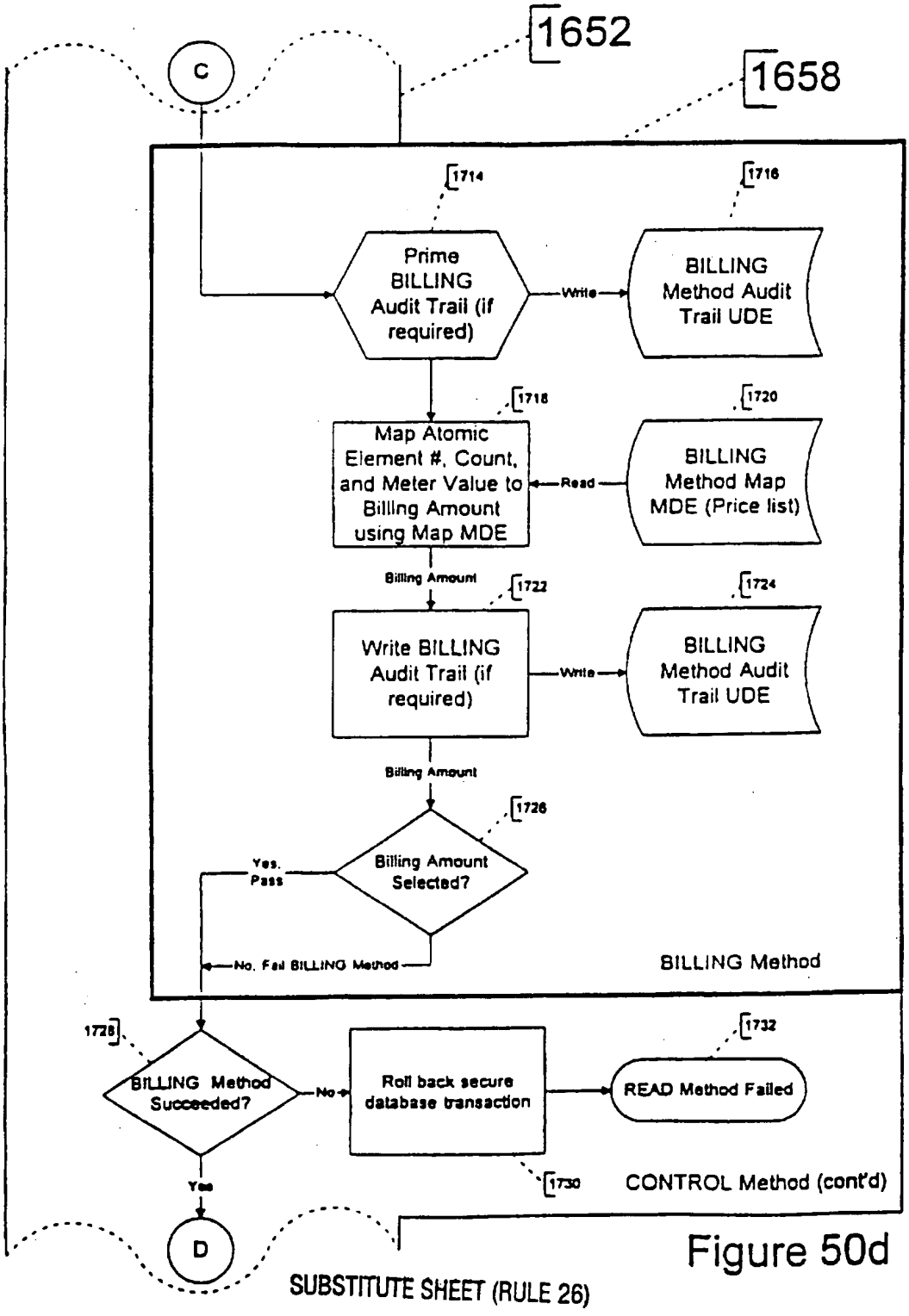


Figure 50d

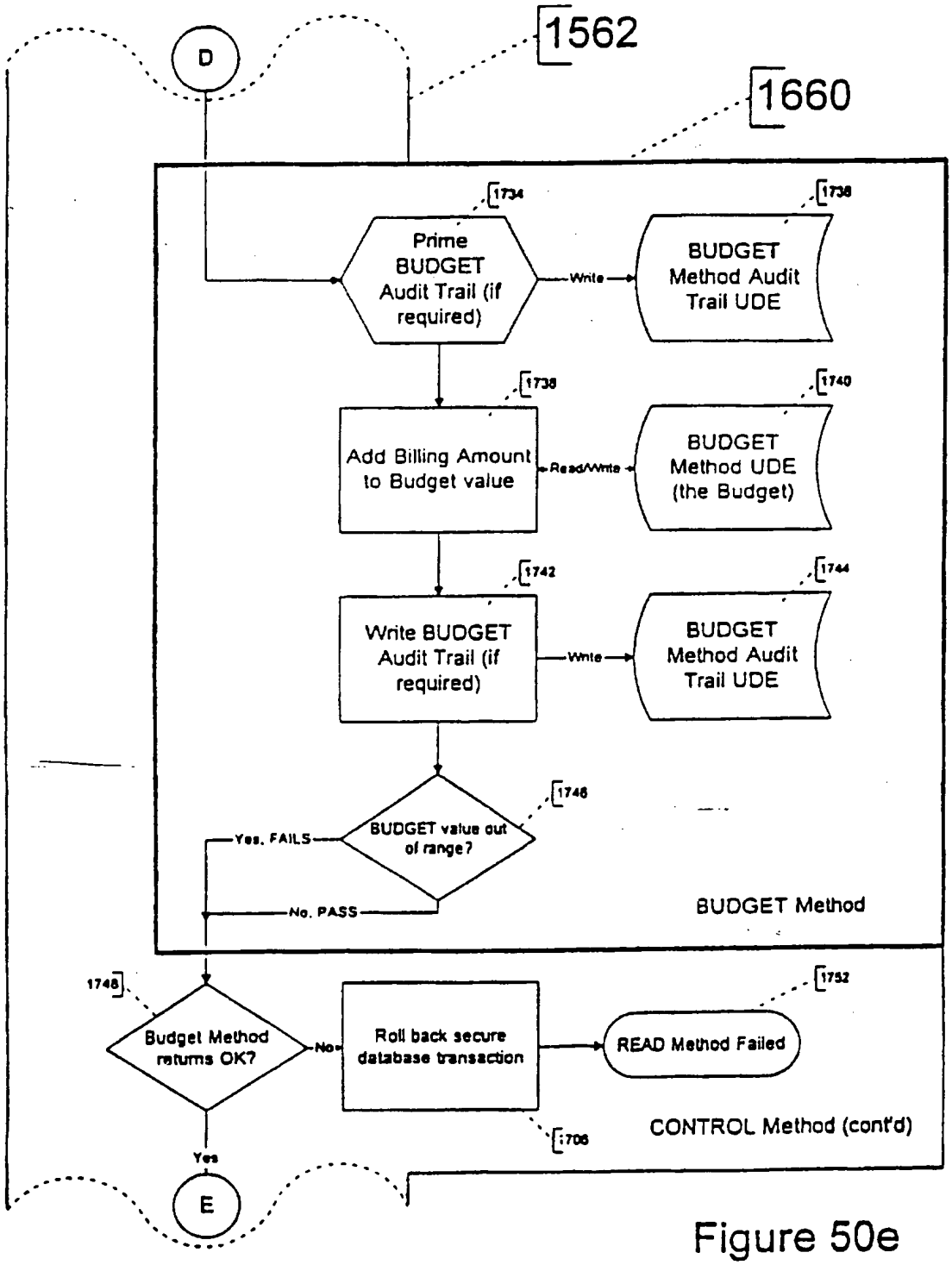


Figure 50e

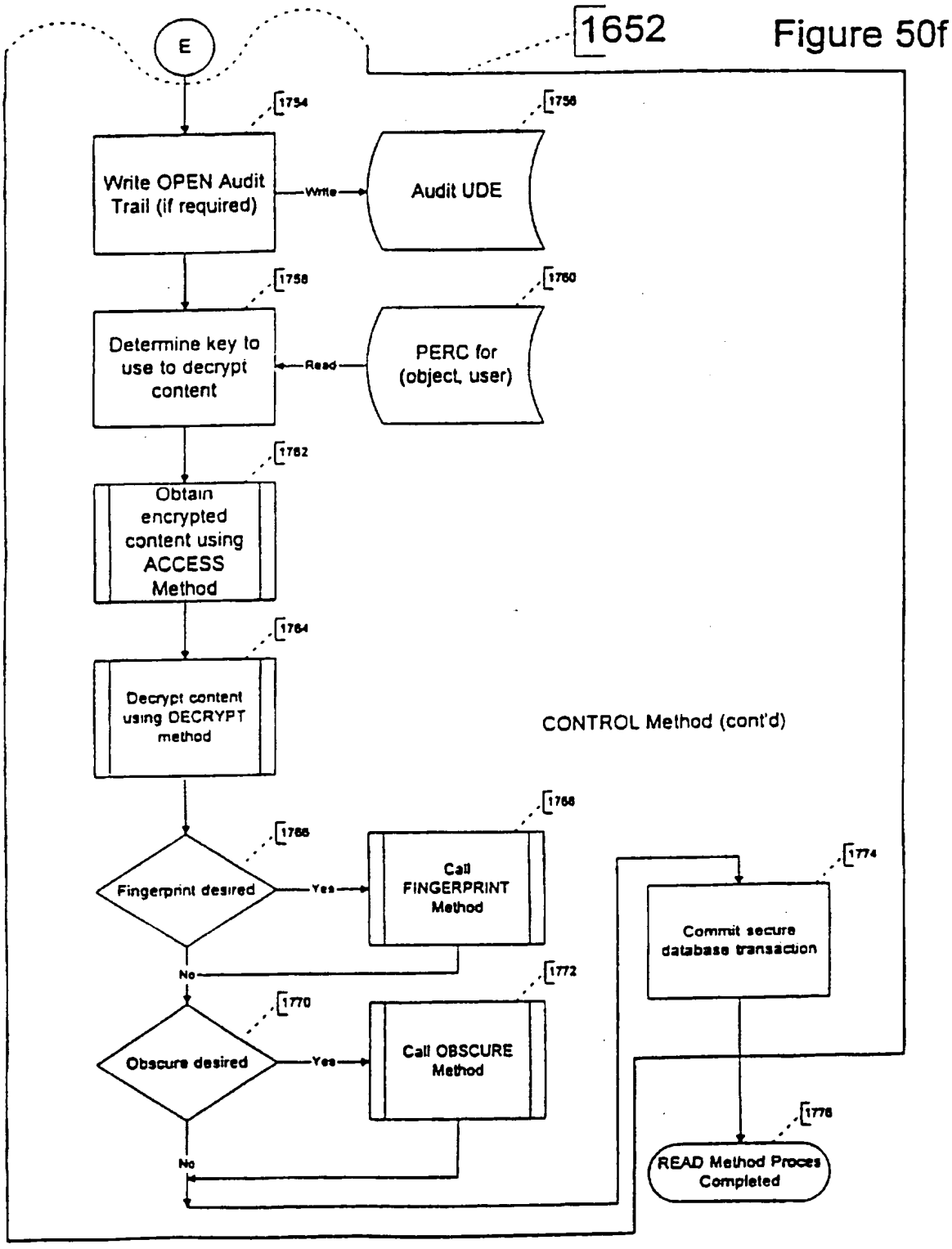
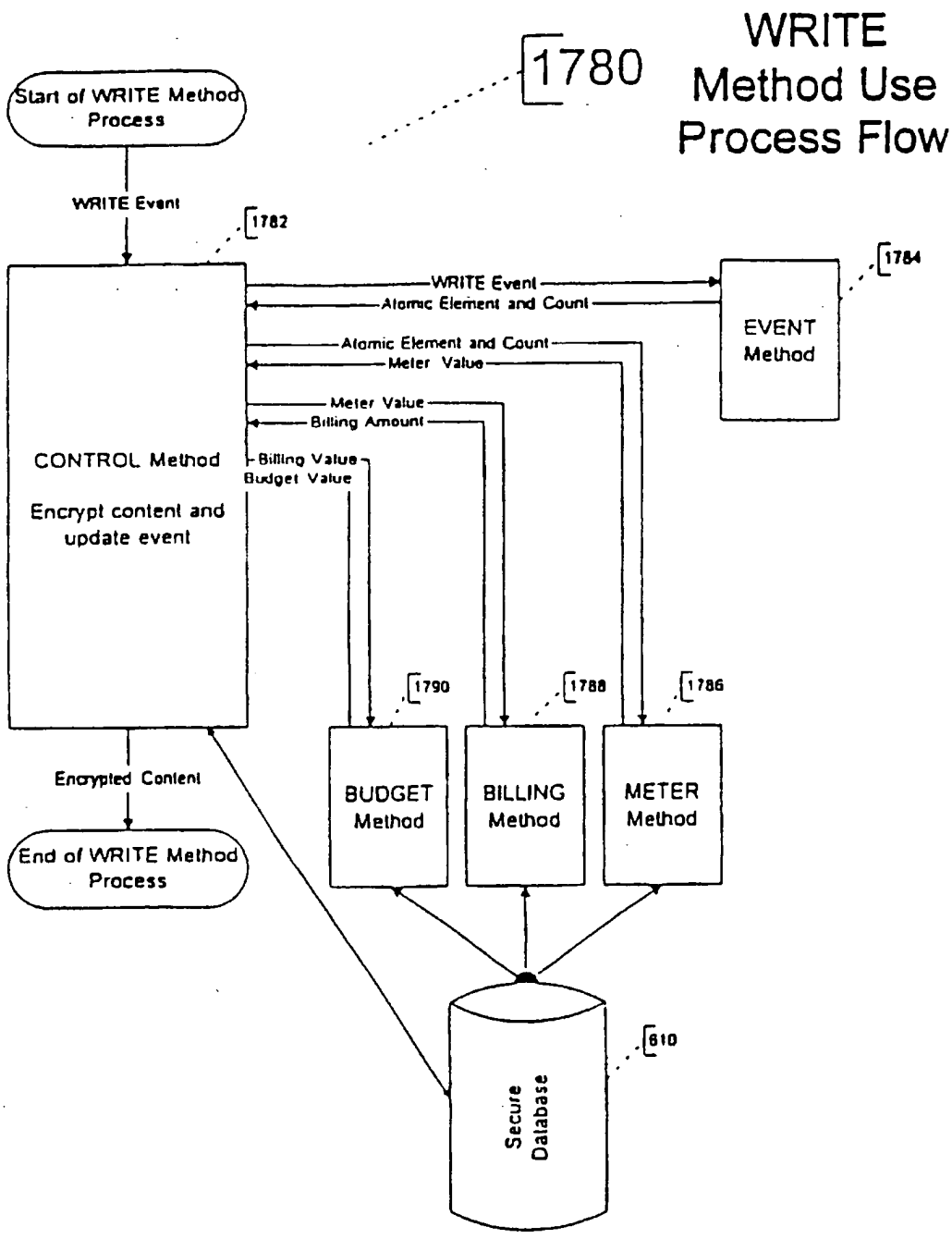


Figure 50f



SUBSTITUTE SHEET (RULE 26)

Figure 51

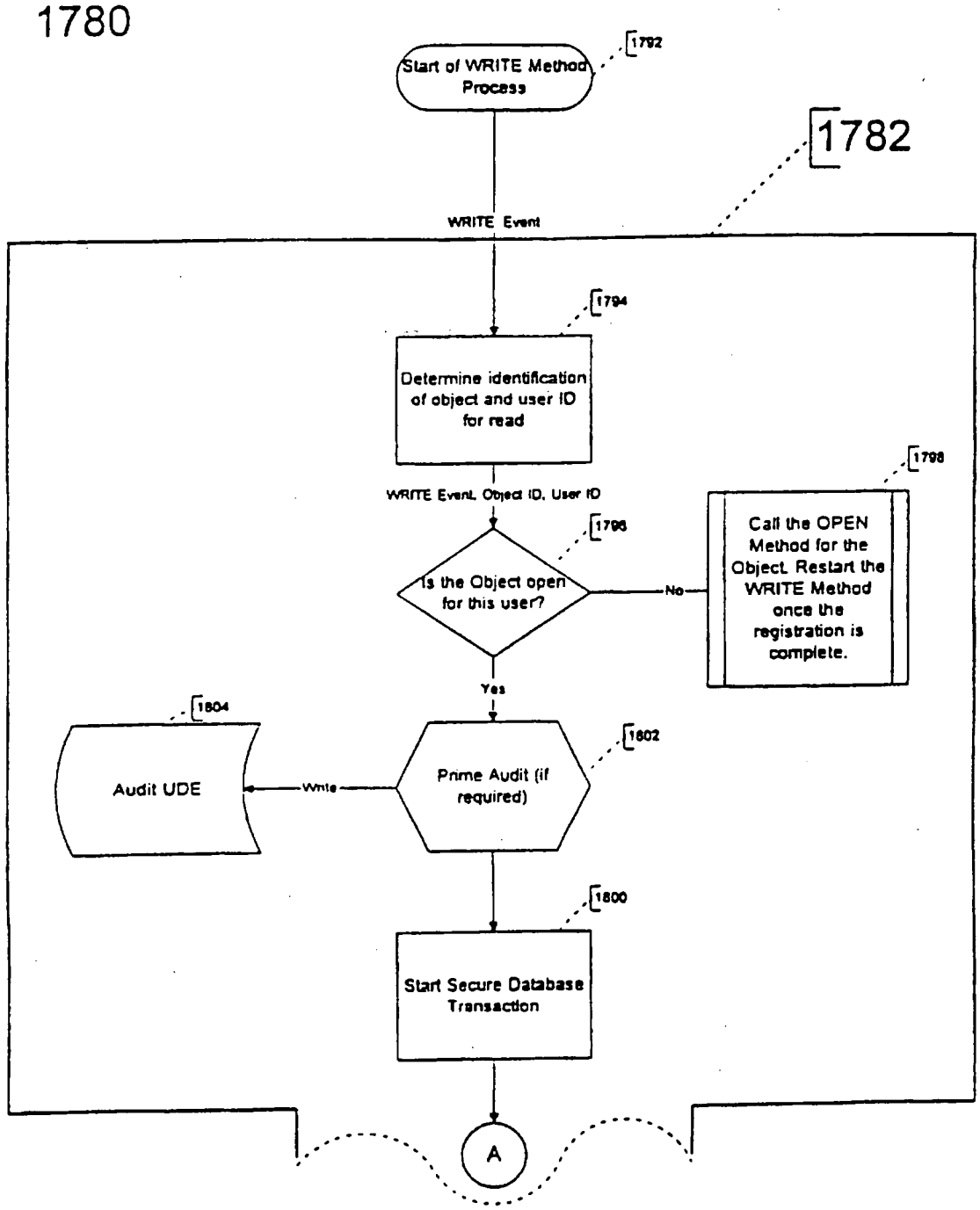

1780
1792
Start of WRITE Method Process
1782
WRITE Event
1794
Determine identification of object and user ID for read
WRITE Event, Object ID, User ID
1798
1796
Is the Object open for this user?
No
Call the OPEN Method for the Object. Restart the WRITE Method once the registration is complete.
Yes
1804
1802
Audit UDE
Write
Prime Audit (if required)
1800
Start Secure Database Transaction
A

Figure 51a

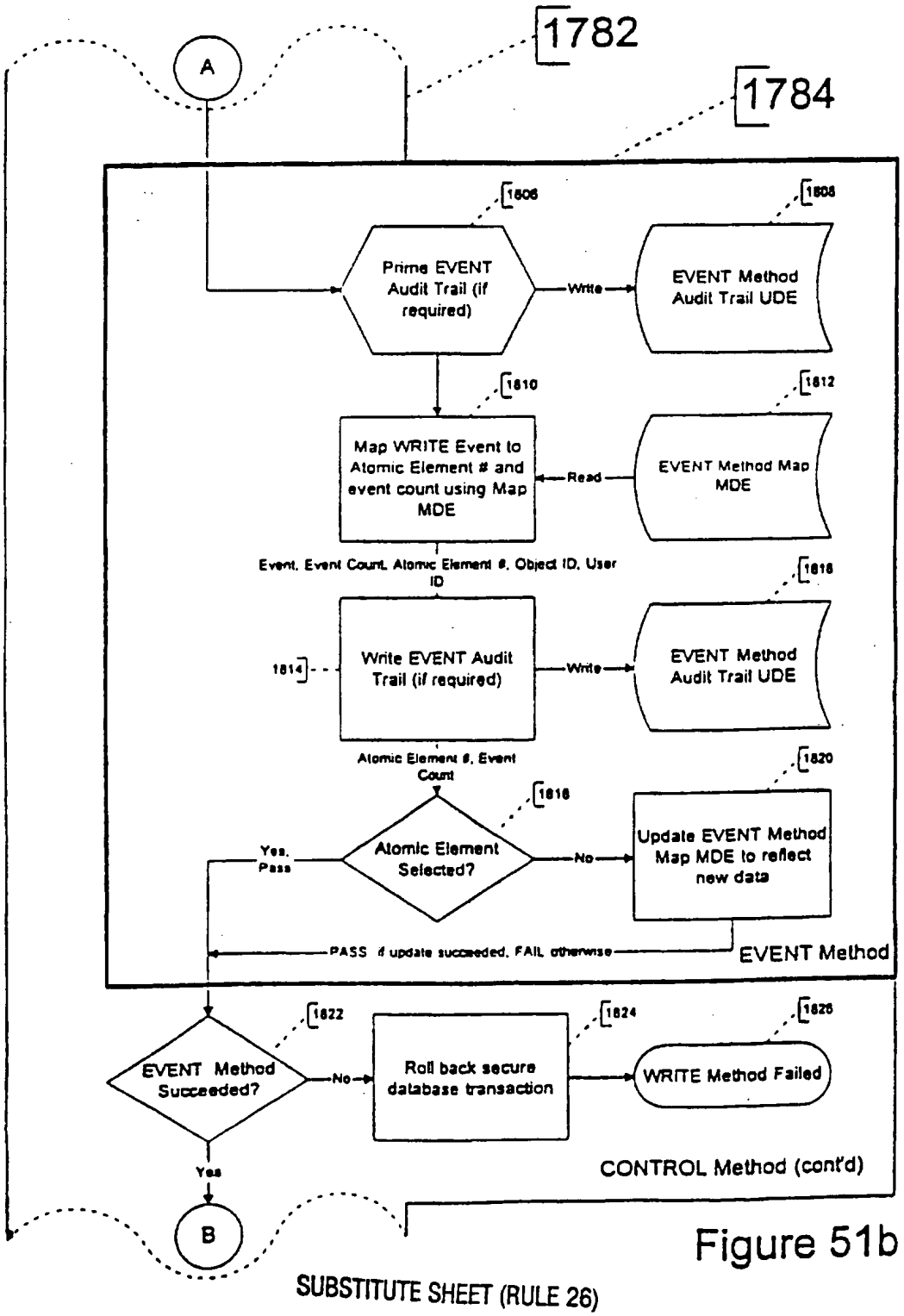


Figure 51b

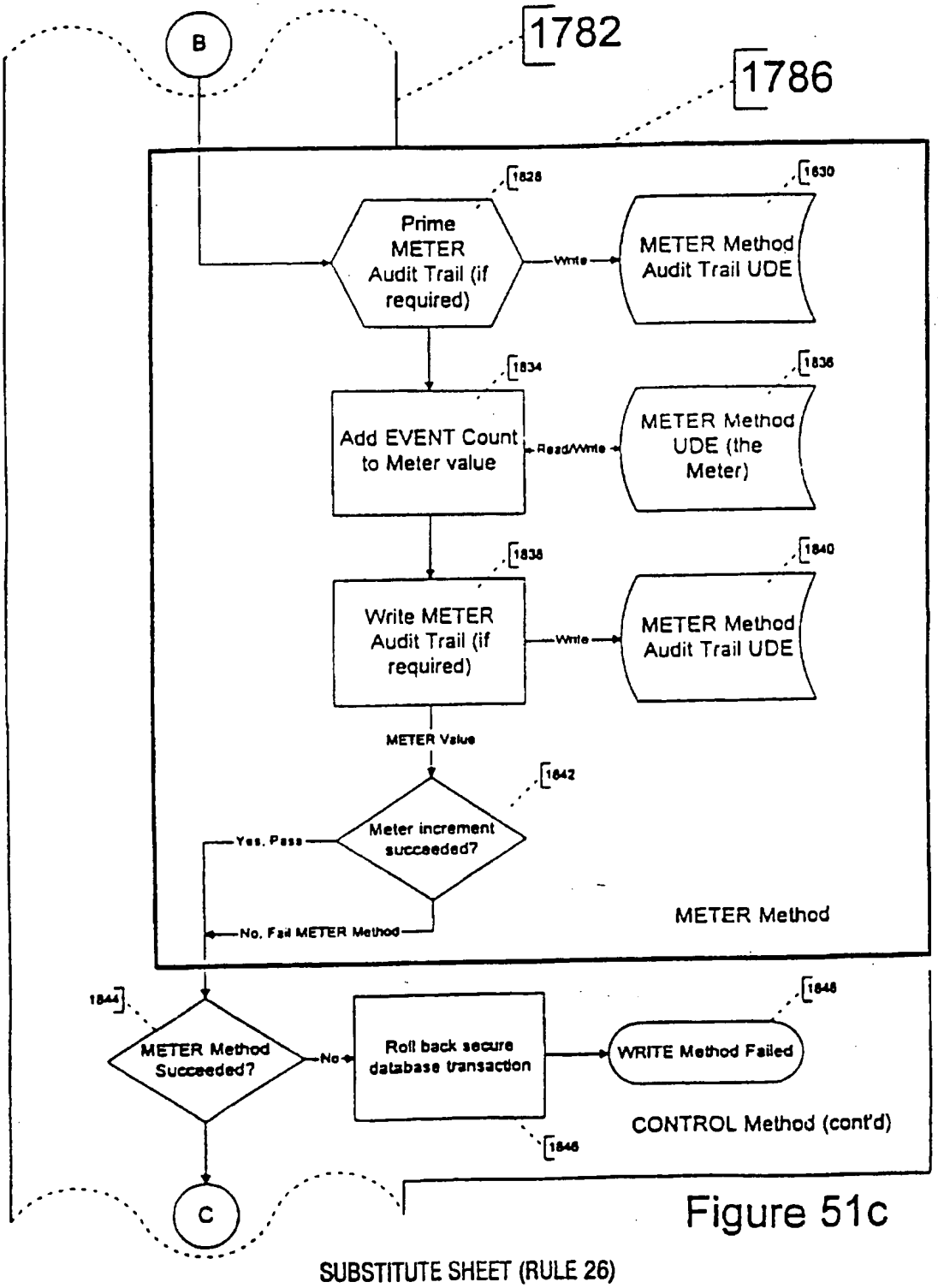


Figure 51c

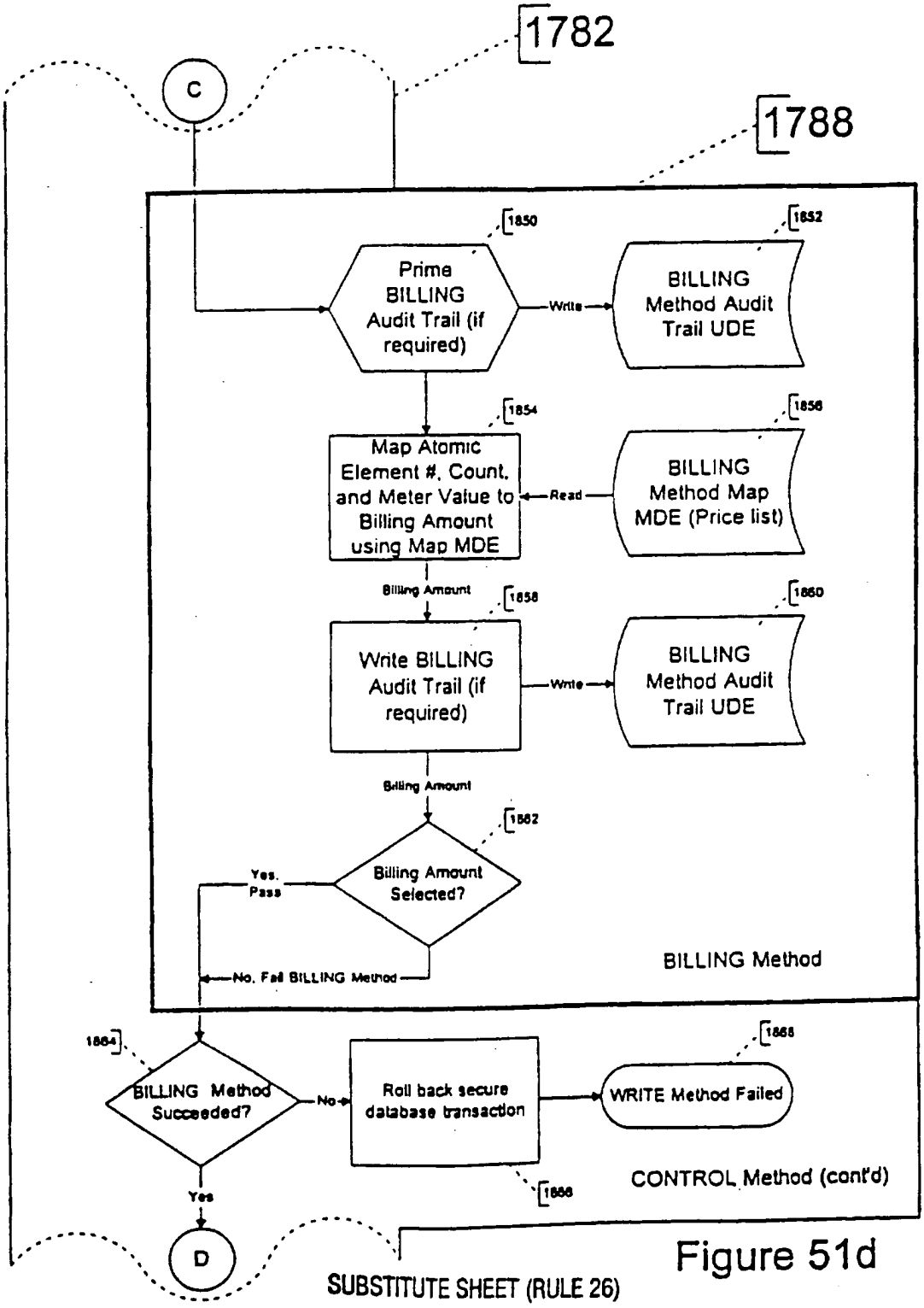


Figure 51d

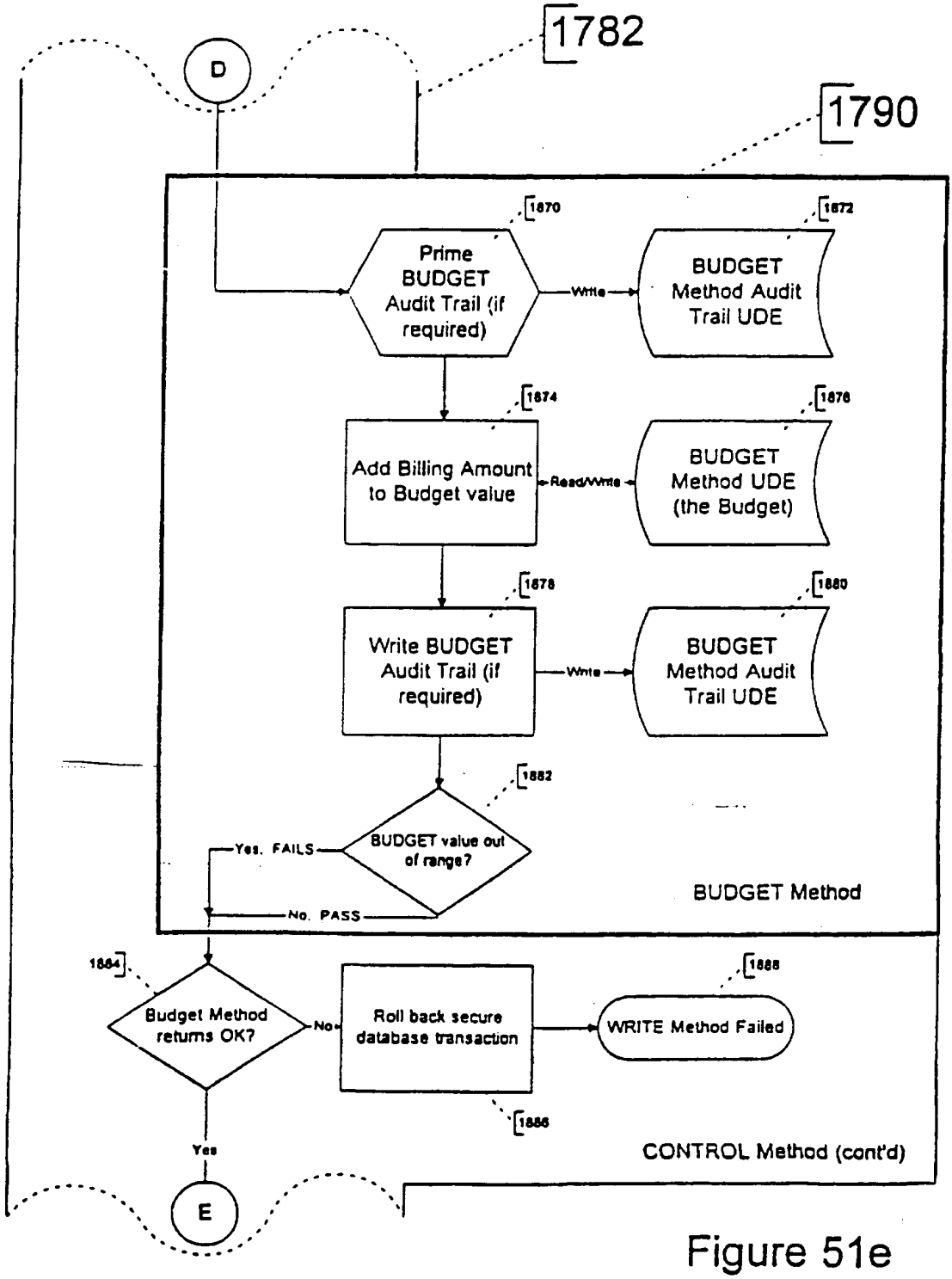


Figure 51e

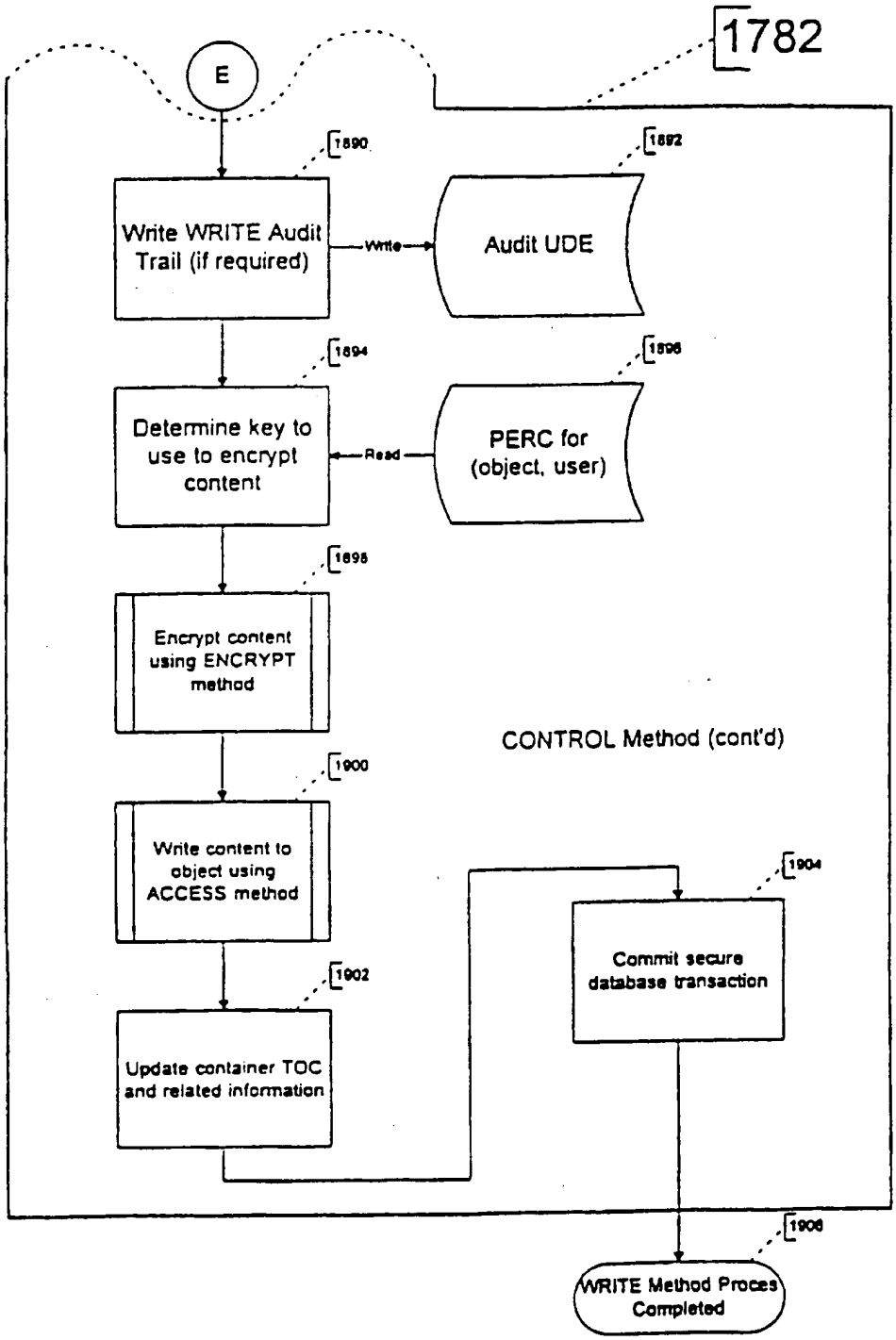
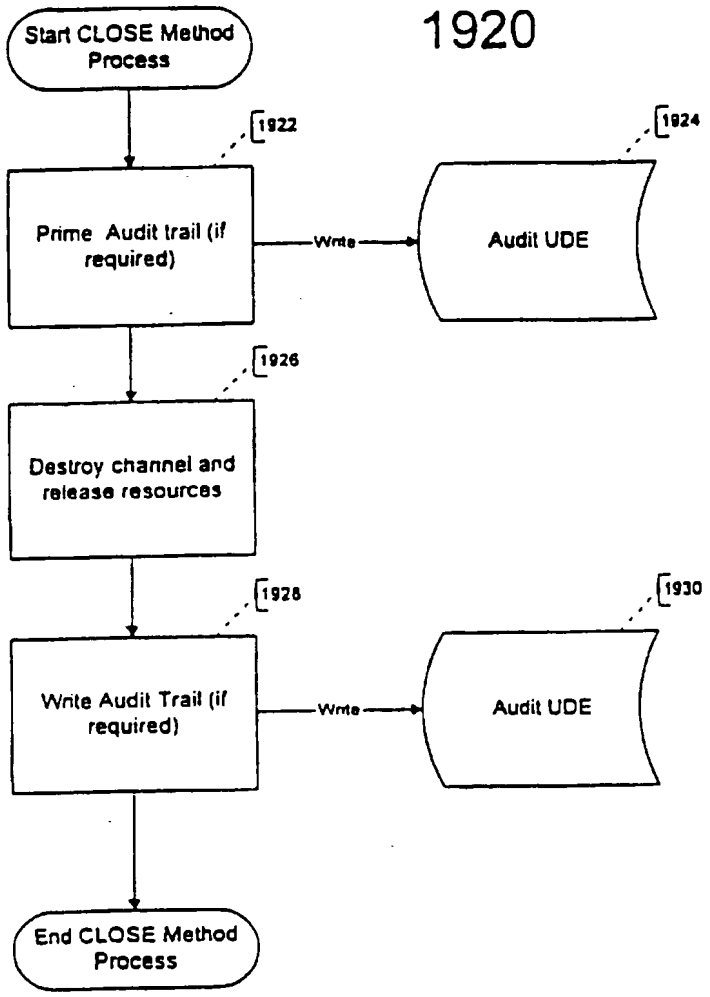


Figure 51f



CLOSE Method Process Flow

Figure 52

EVENT Method Process Flows

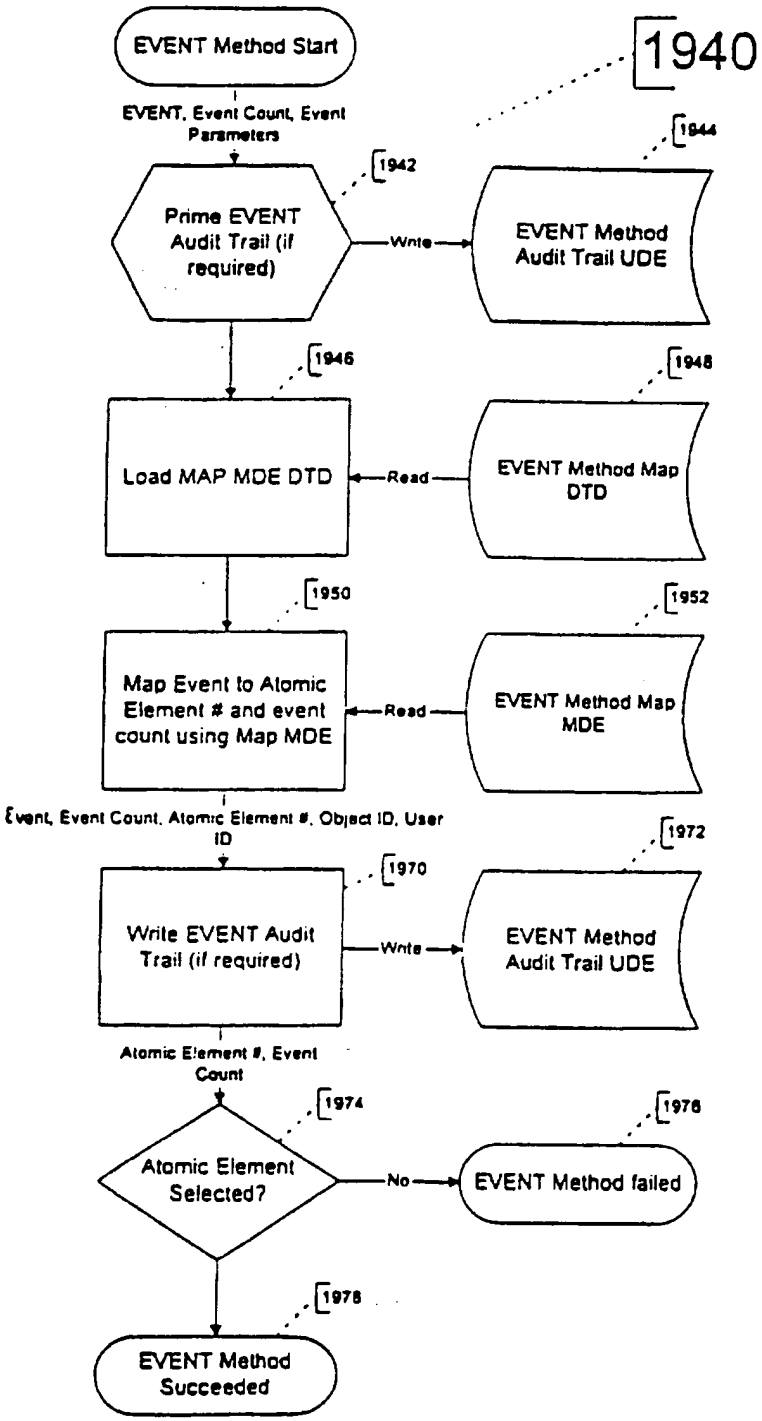


Figure 53a

Sample EVENT Method Mapping Process

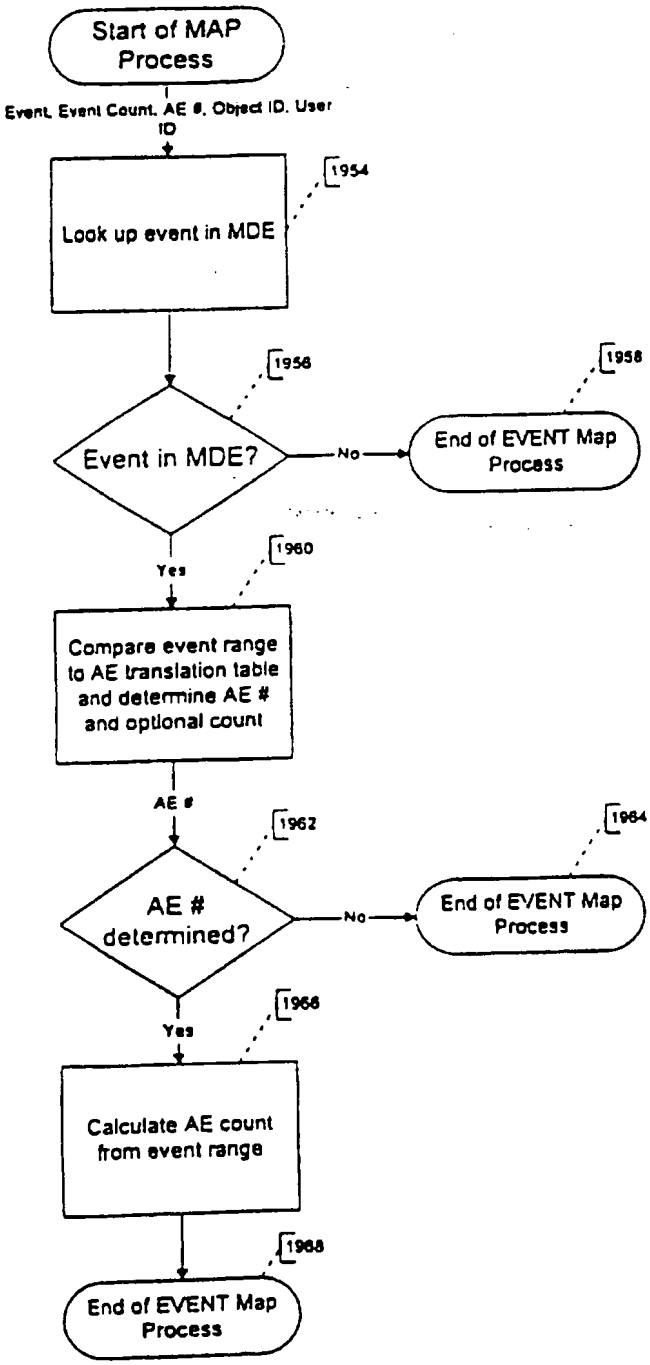
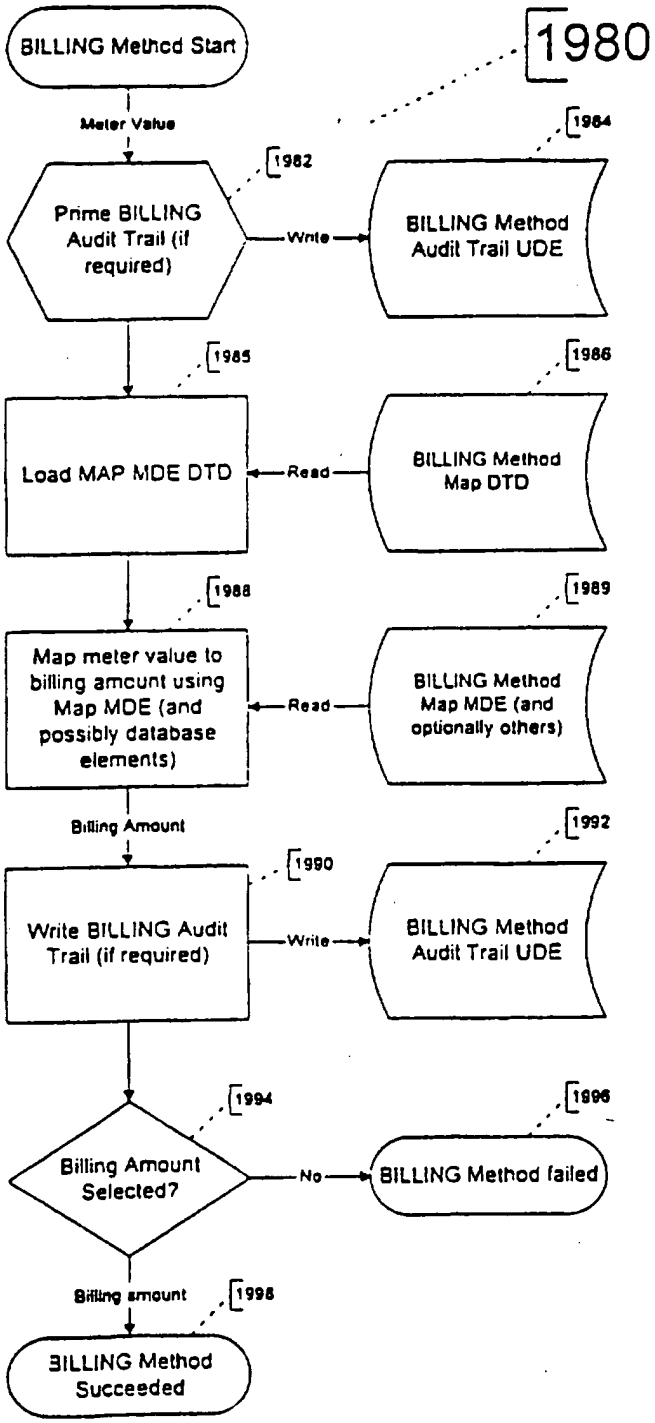


Figure 53b



BILLING Method Process Flows

Figure 53c

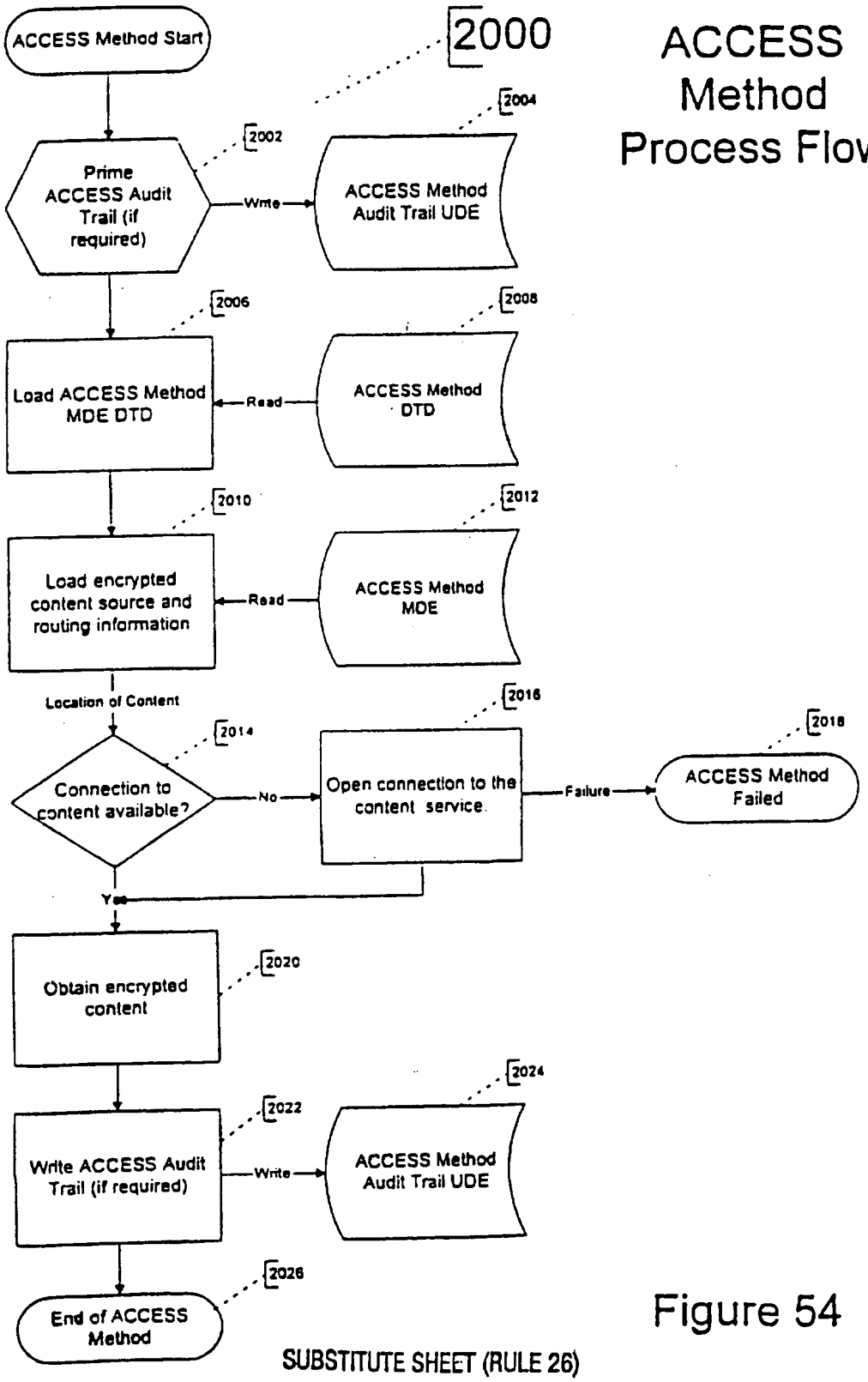


Figure 54

DECRYPT Method Process Flow

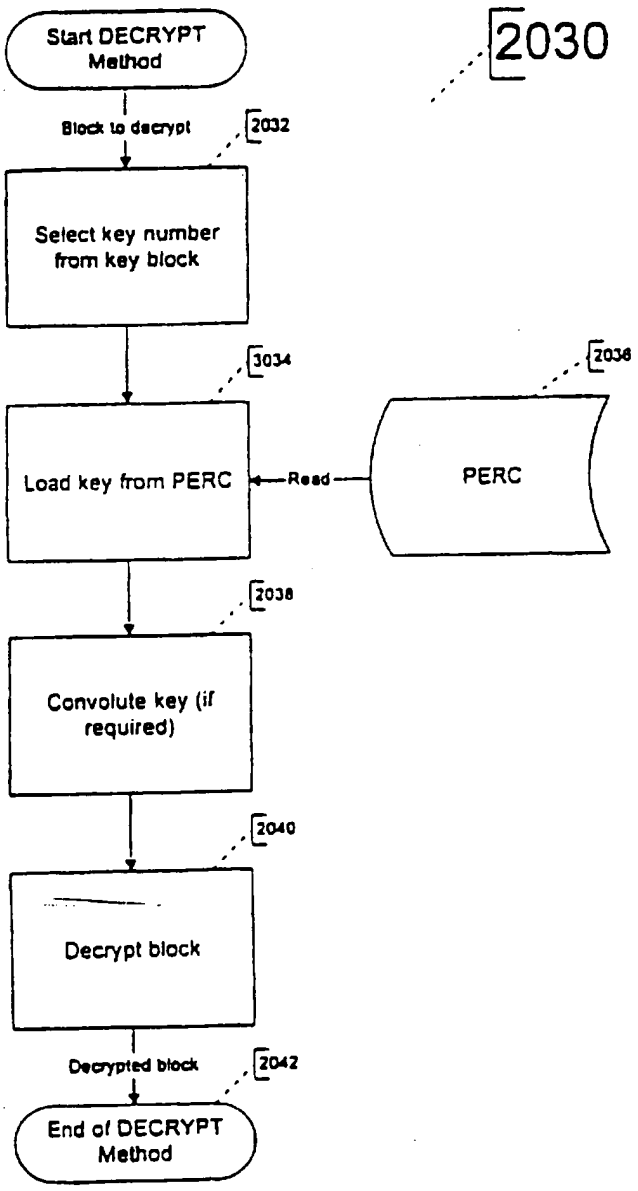
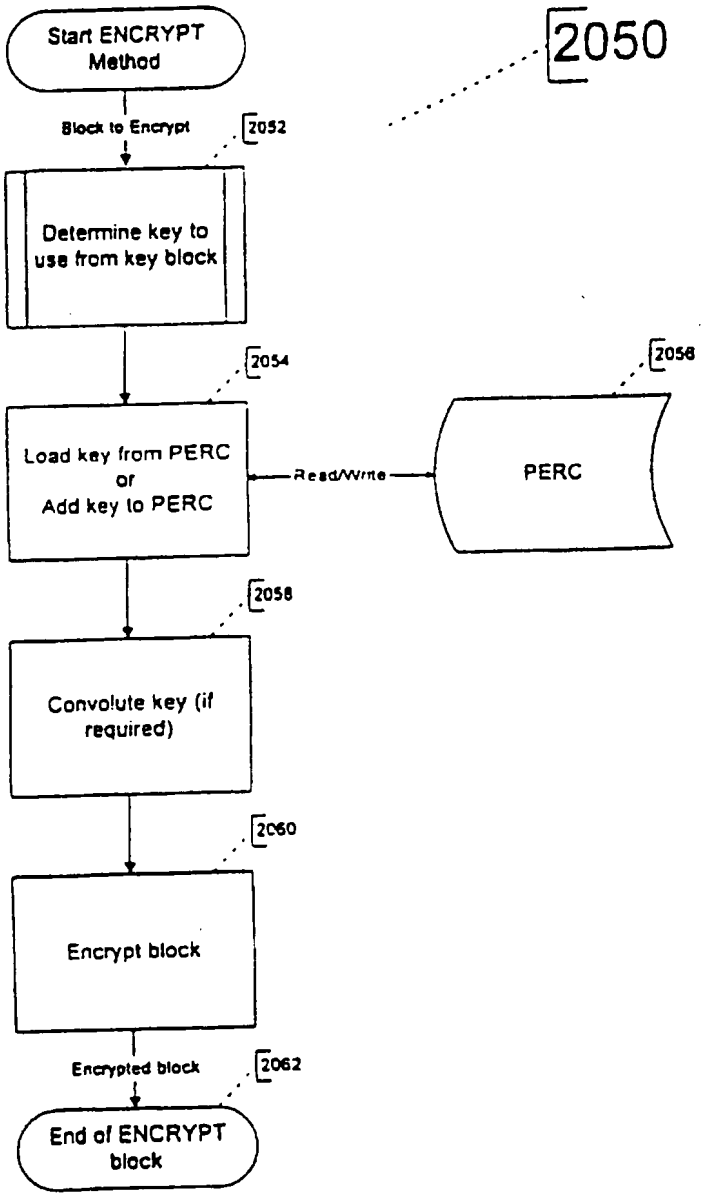
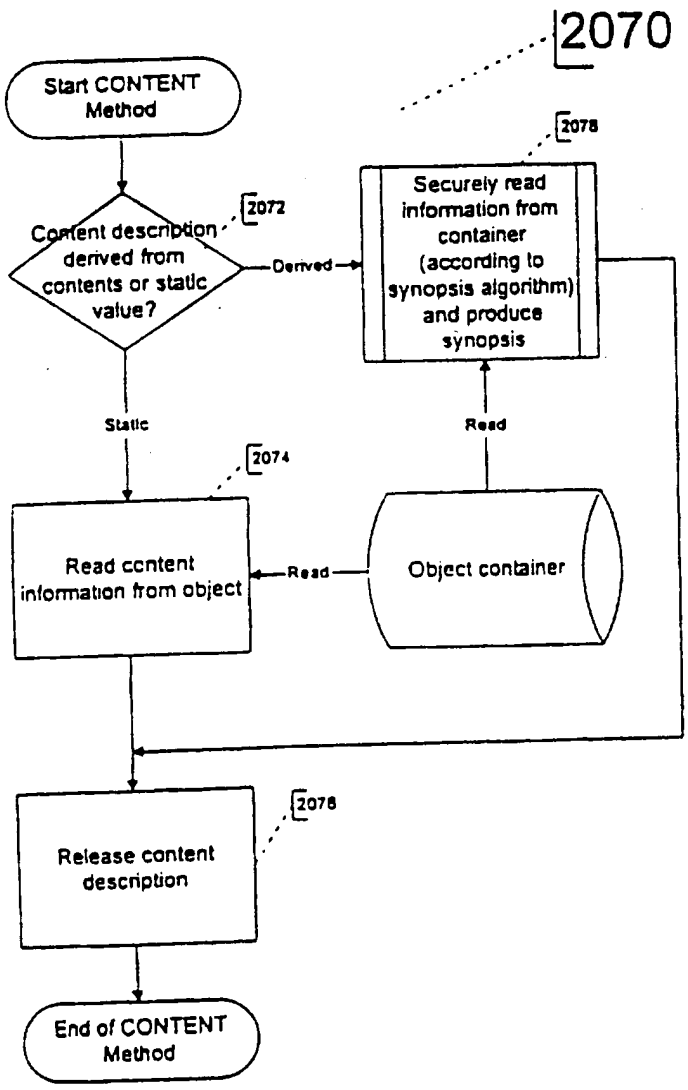


Figure 55a



ENCRYPT Method Process Flow

Figure 55b



CONTENT Method Process Flow

Figure 56

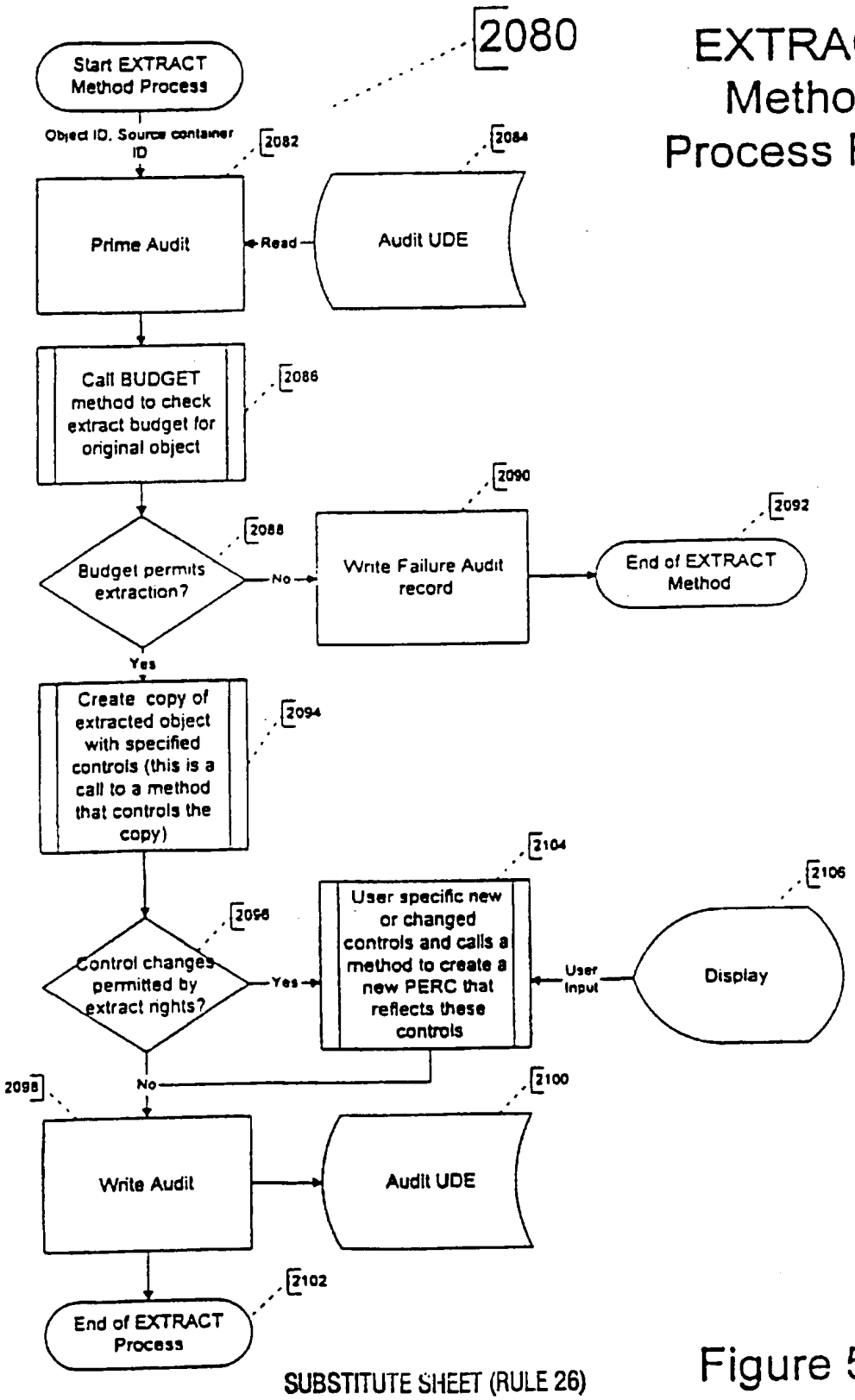

2080
EXTRACT Method Process Flow
Start EXTRACT Method Process
Object ID, Source container ID
2082
2084
Prime Audit
Read
Audit UDE
Call BUDGET method to check extract budget for original object
2086
2090
2092
2088
Budget permits extraction?
No
Write Failure Audit record
End of EXTRACT Method
Yes
Create copy of extracted object with specified controls (this is a call to a method that controls the copy)
2094
2104
2106
2096
Control changes permitted by extract rights?
Yes
User specific new or changed controls and calls a method to create a new PERC that reflects these controls
User Input
Display
No
2098
2100
Write Audit
Audit UDE
2102
End of EXTRACT Process

Figure 57a

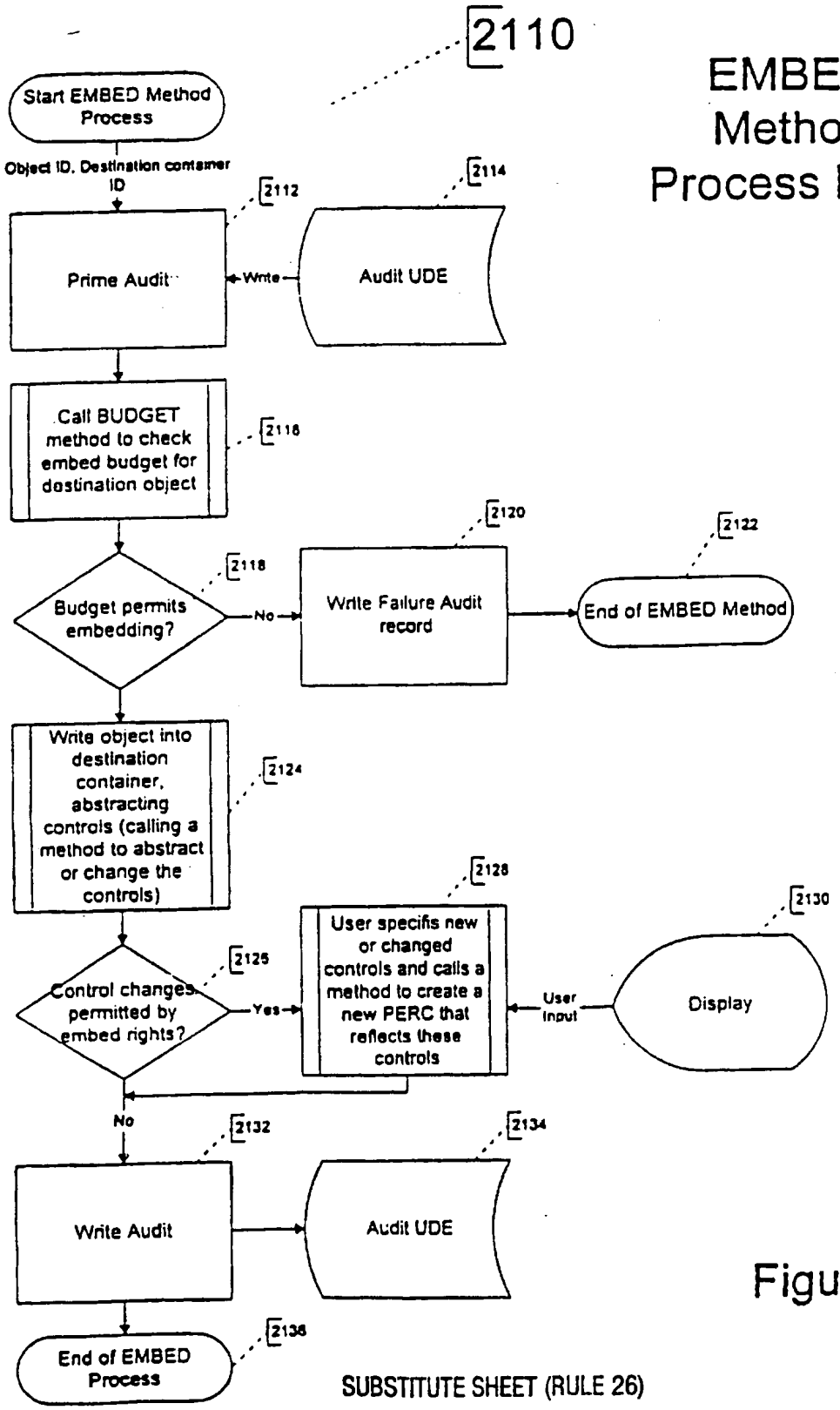
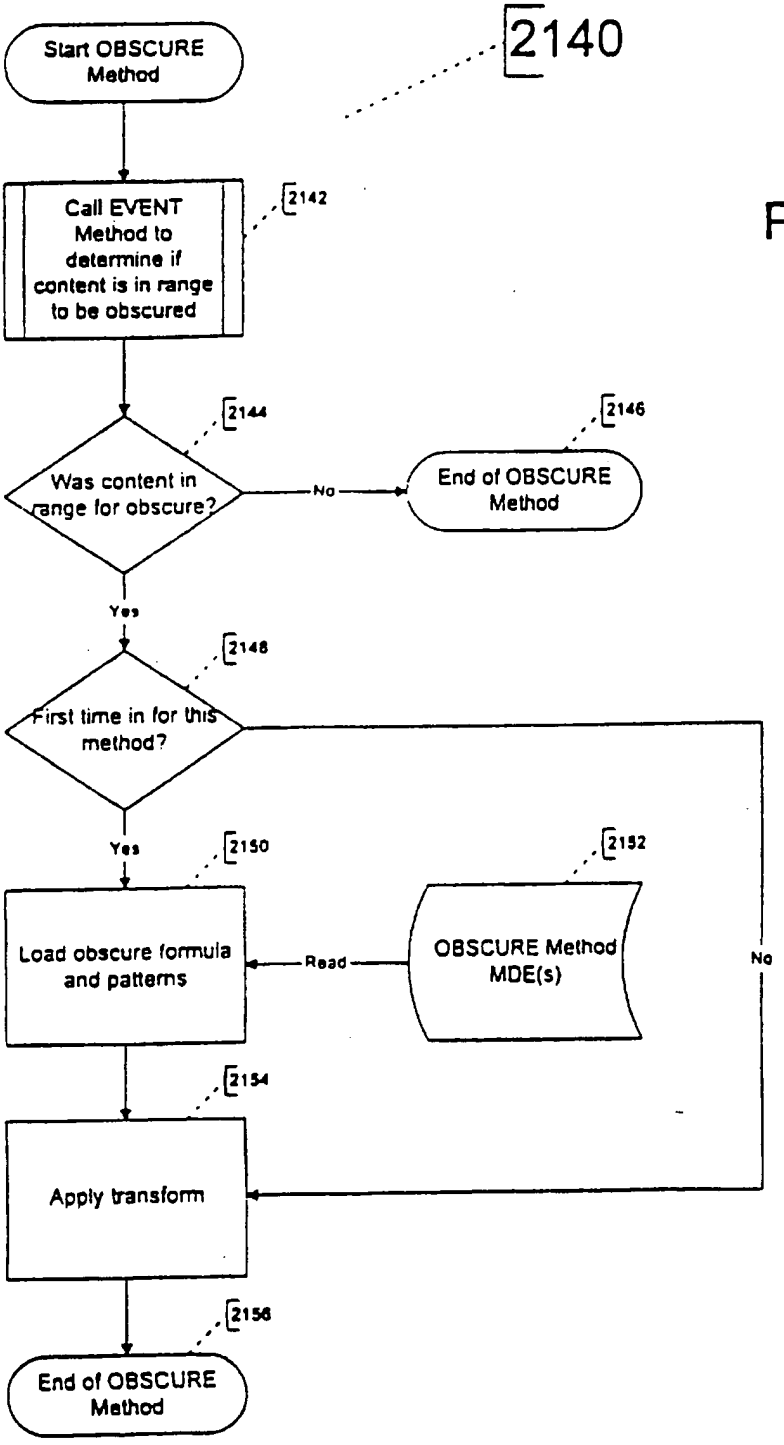
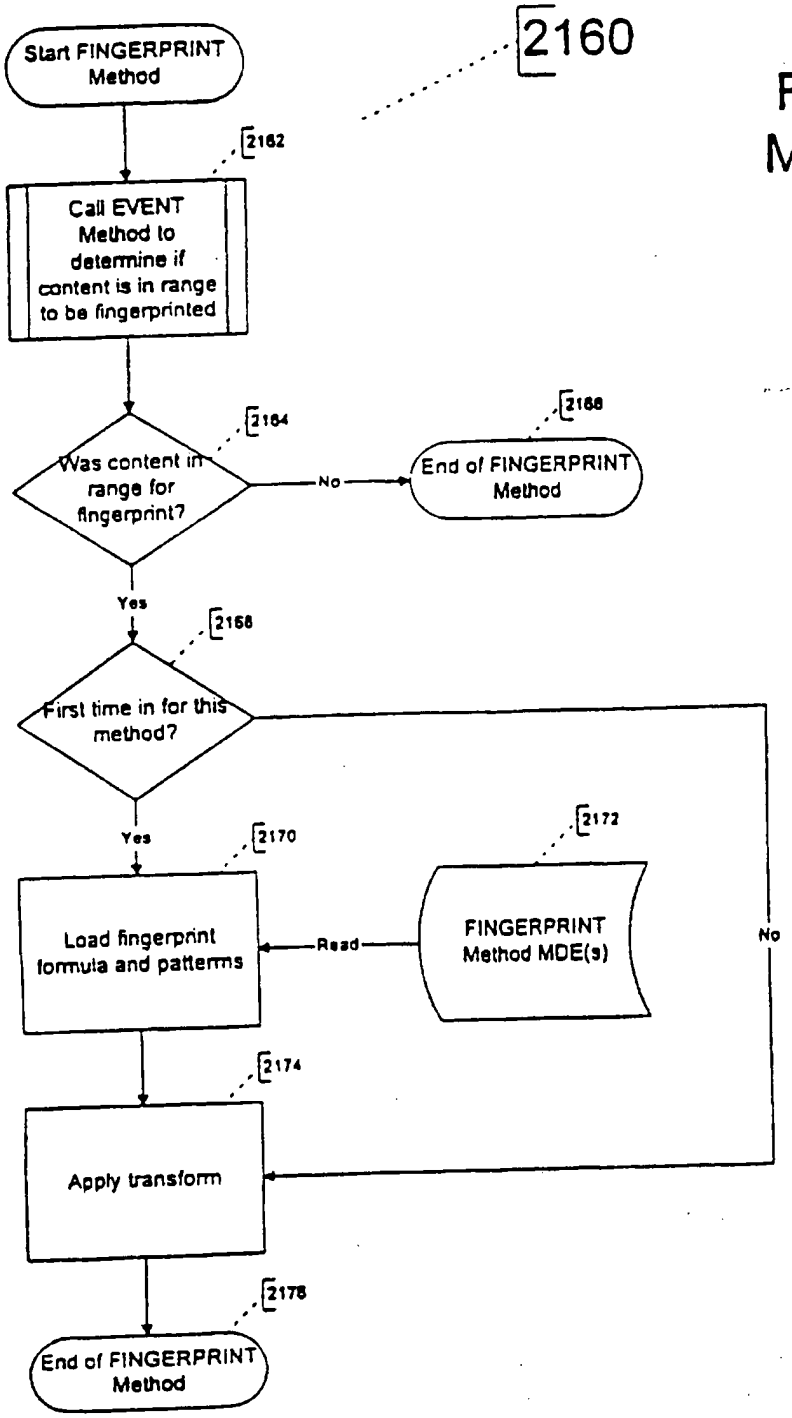


Figure 57b



OBSCURE Method Process Flow

Figure 58a



FINGERPRINT Method Process Flow

Figure 58b

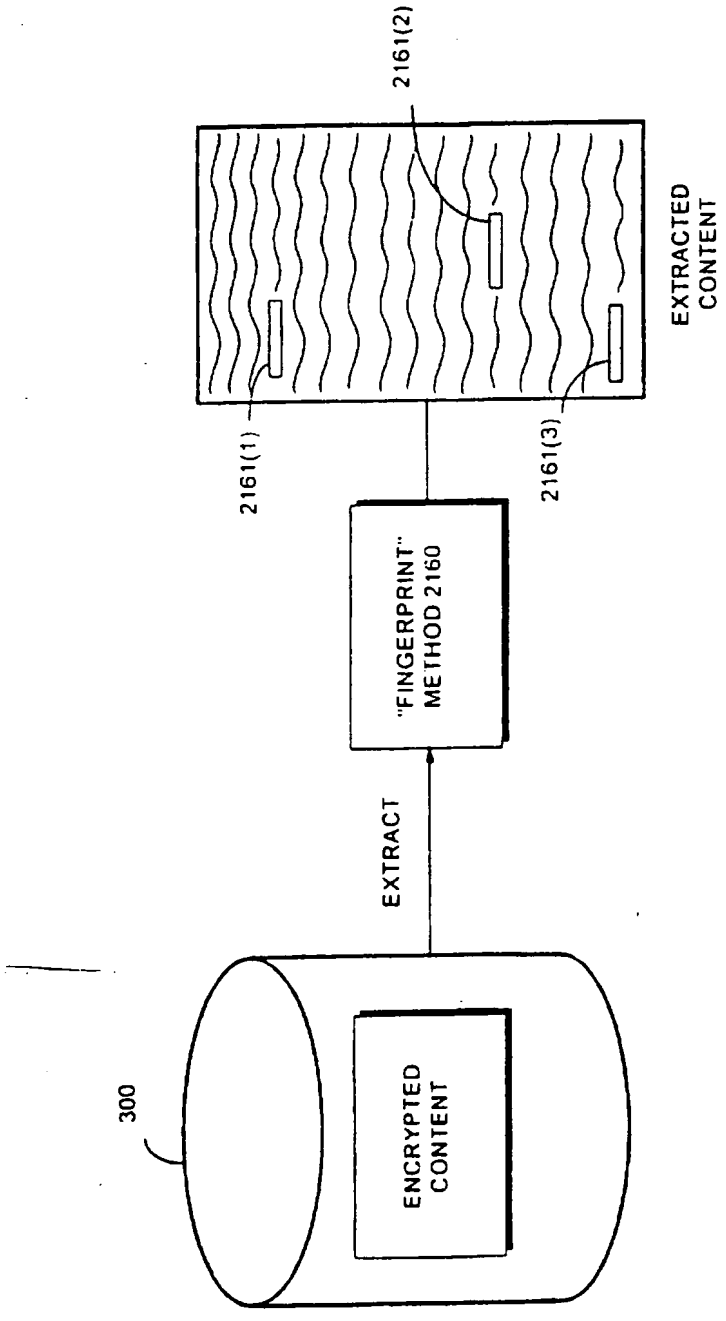
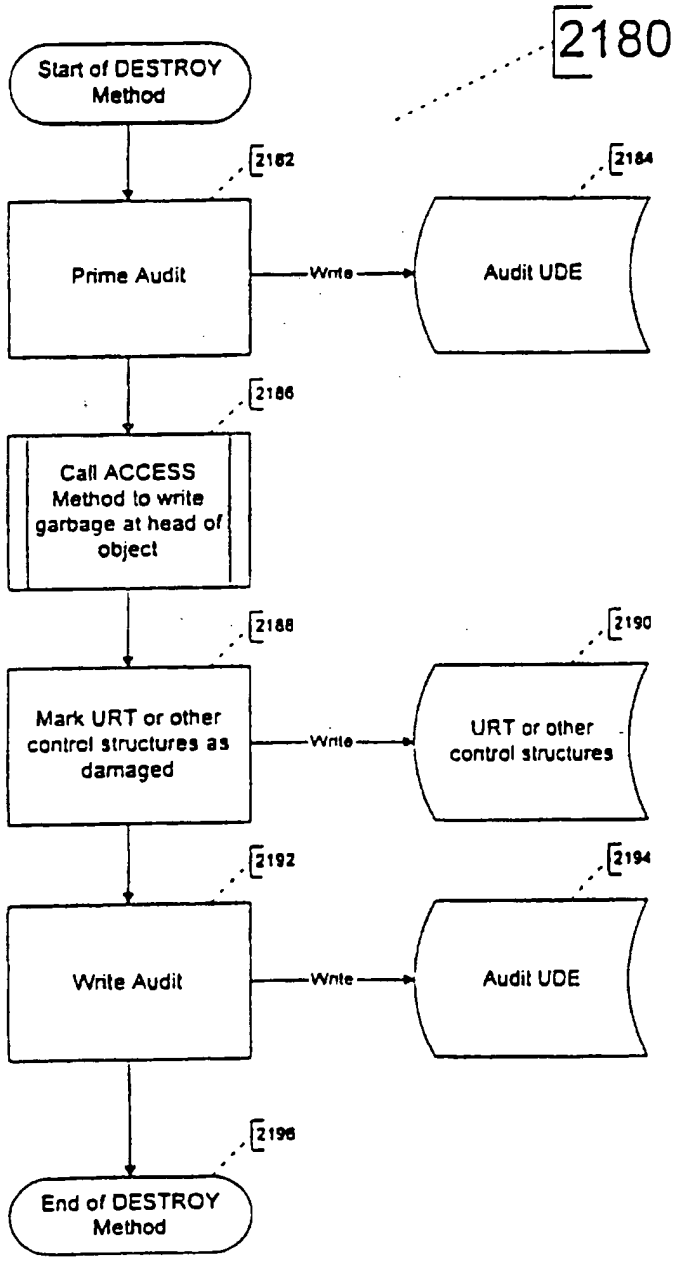
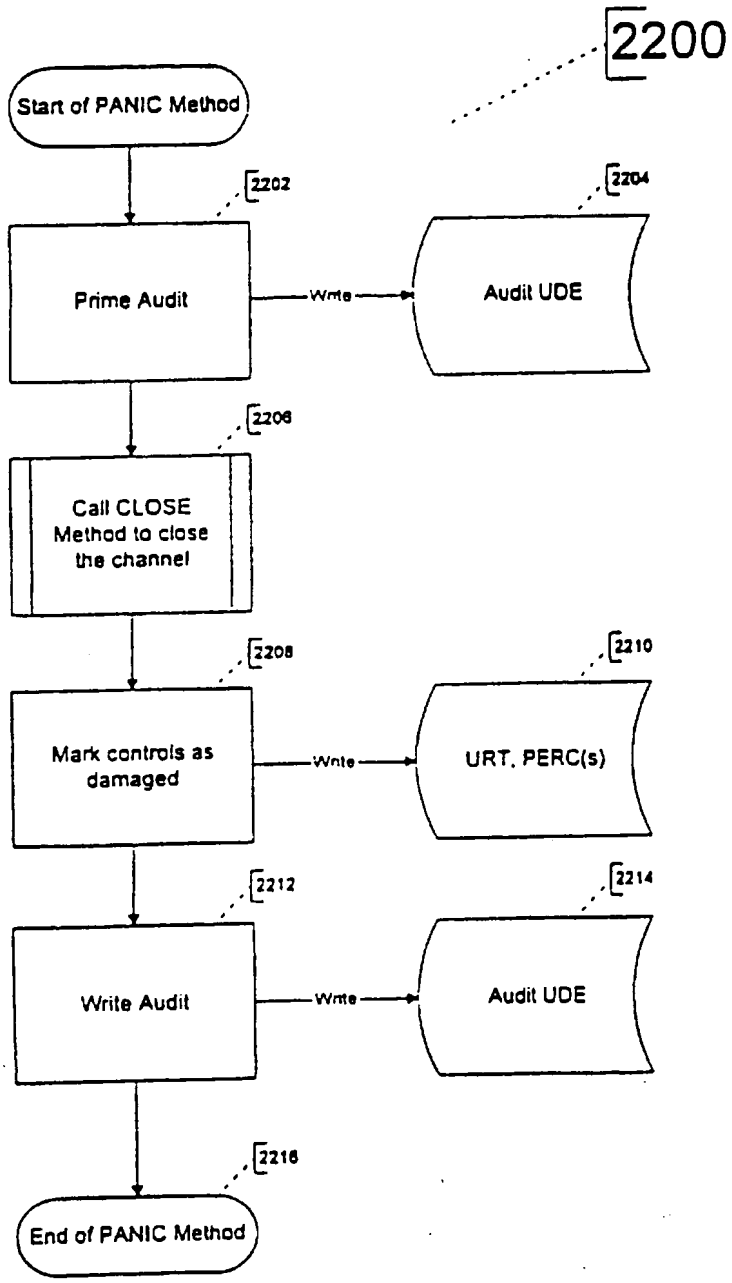


FIG. 58C



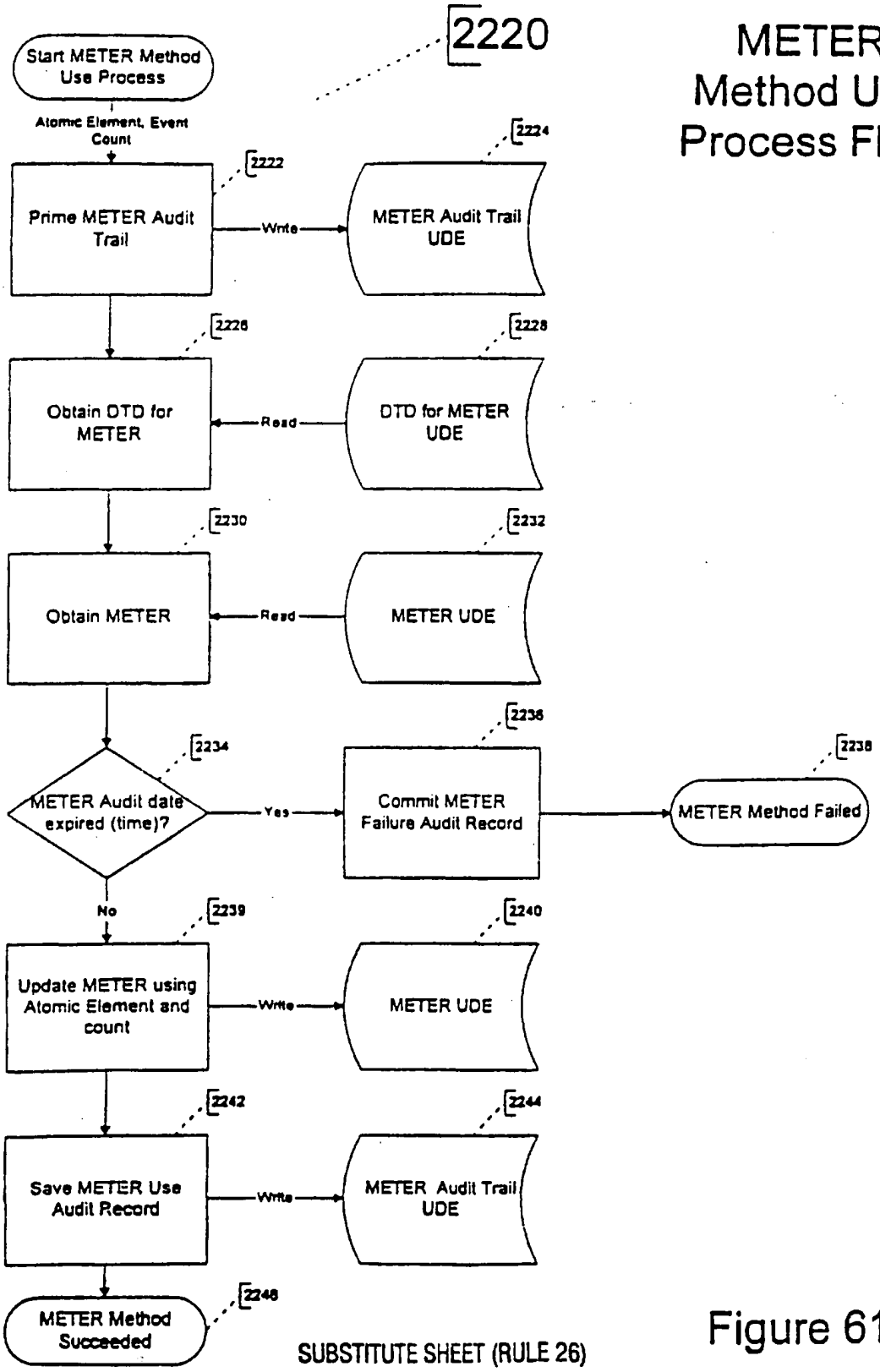
DESTROY Method Process Flow

Figure 59



PANIC Method Process Flow

Figure 60

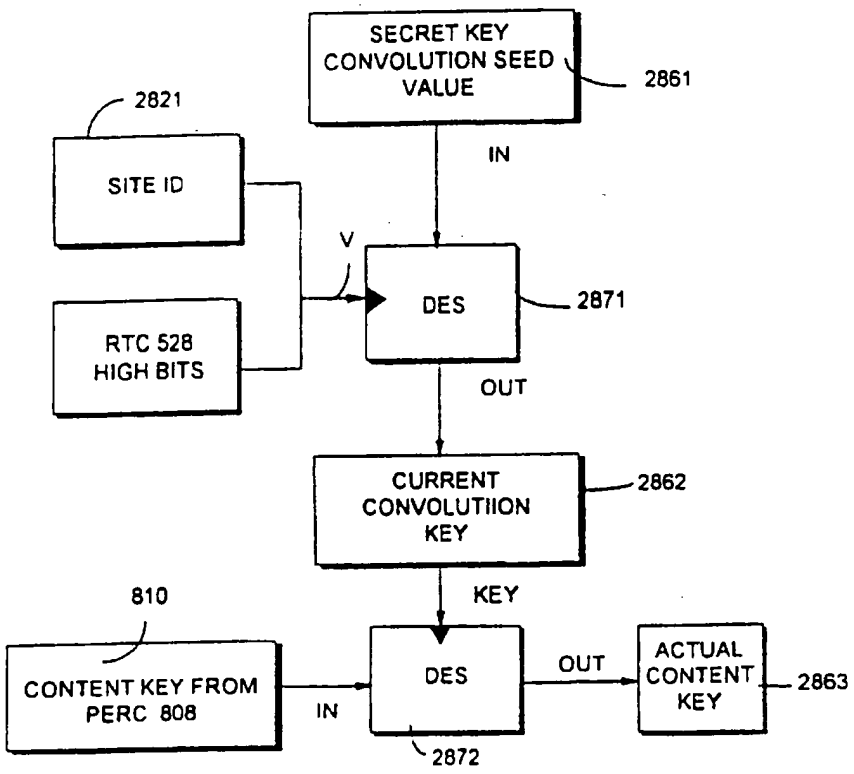


METER Method Use Process Flow

Figure 61

FIG. 62

KEY CONVOLUTION PROCESS



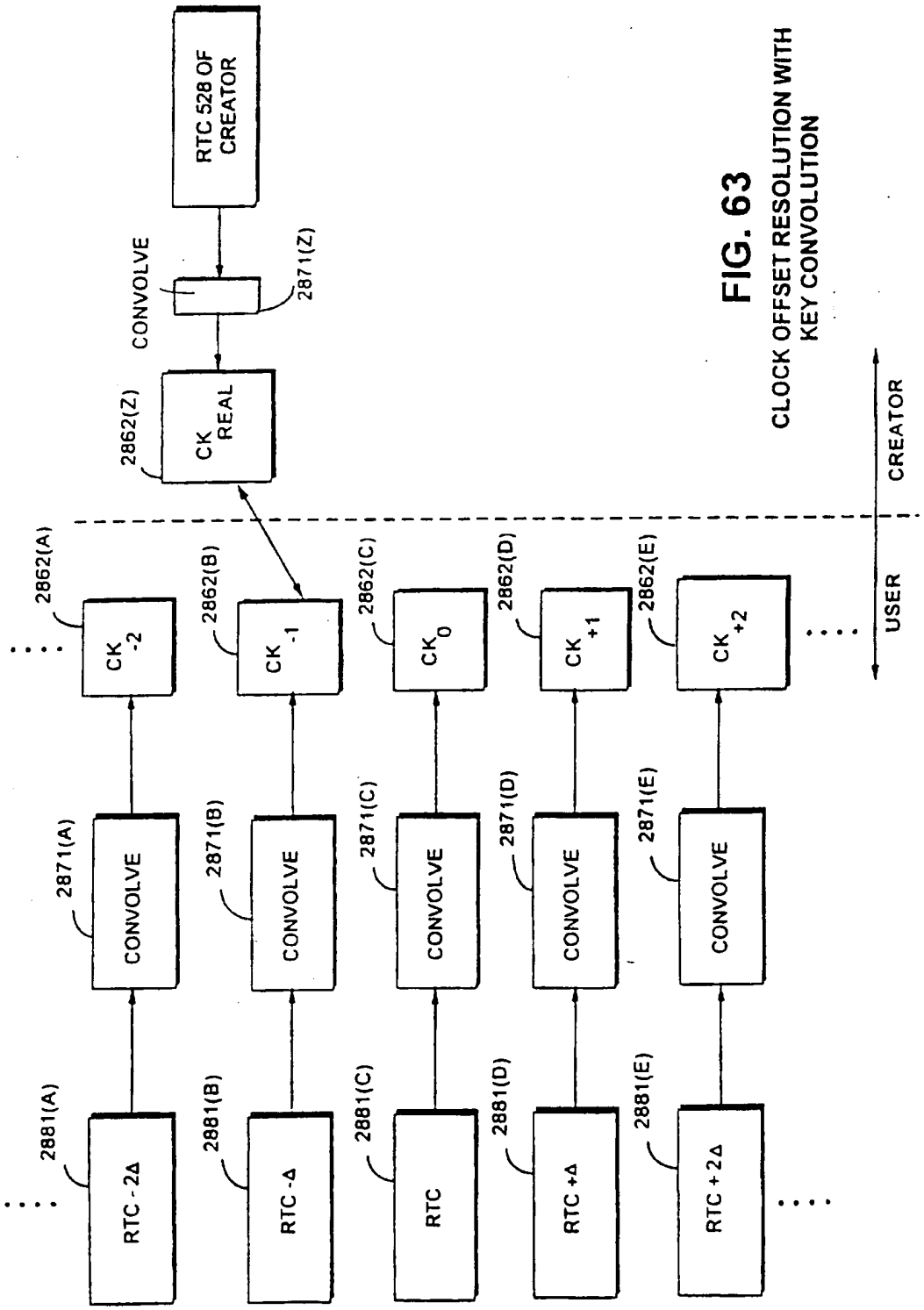


FIG. 63
CLOCK OFFSET RESOLUTION WITH KEY CONVOLUTION

FIG. 64 SPU KEY INITIALIZATION/INSTALLATION

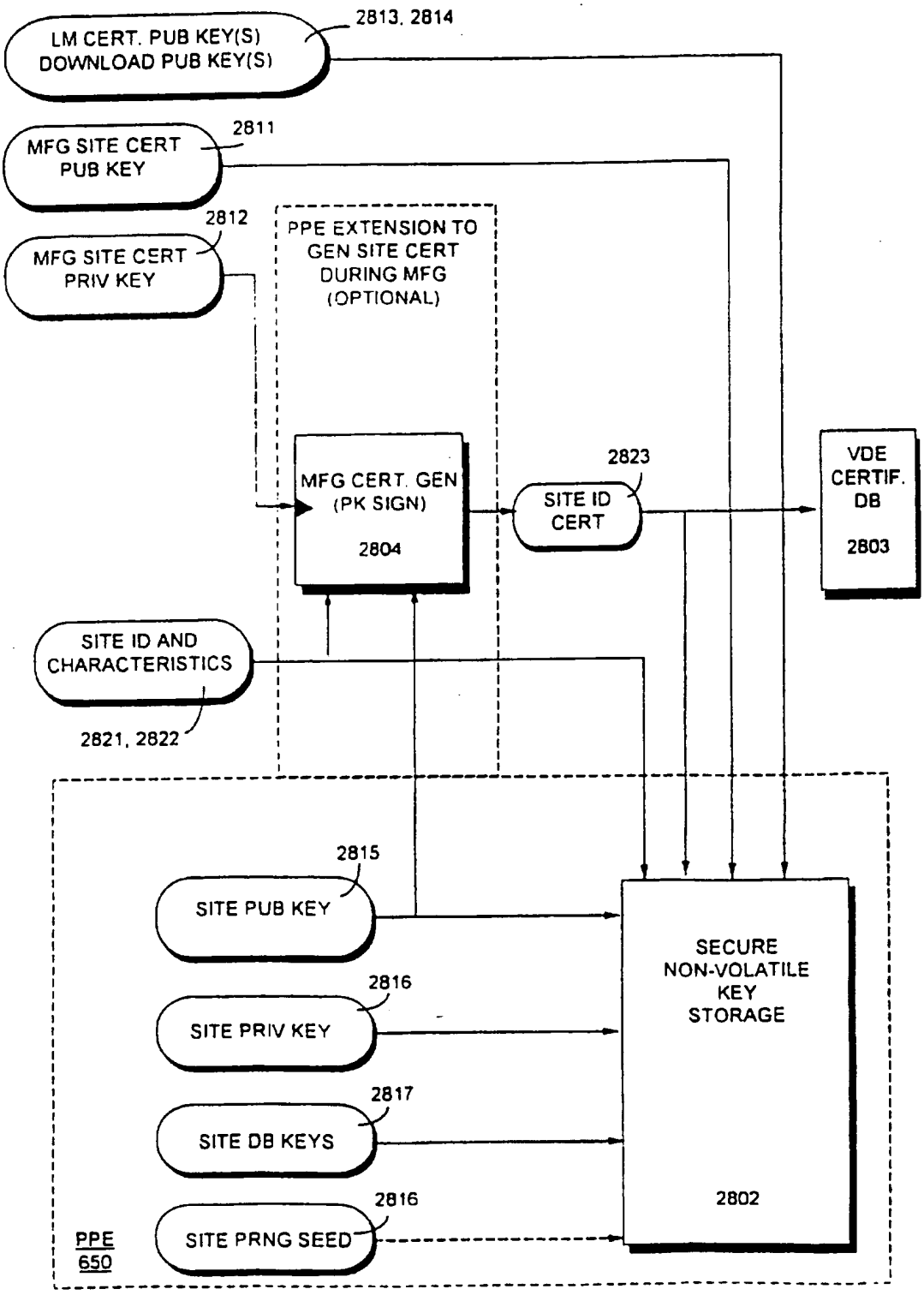
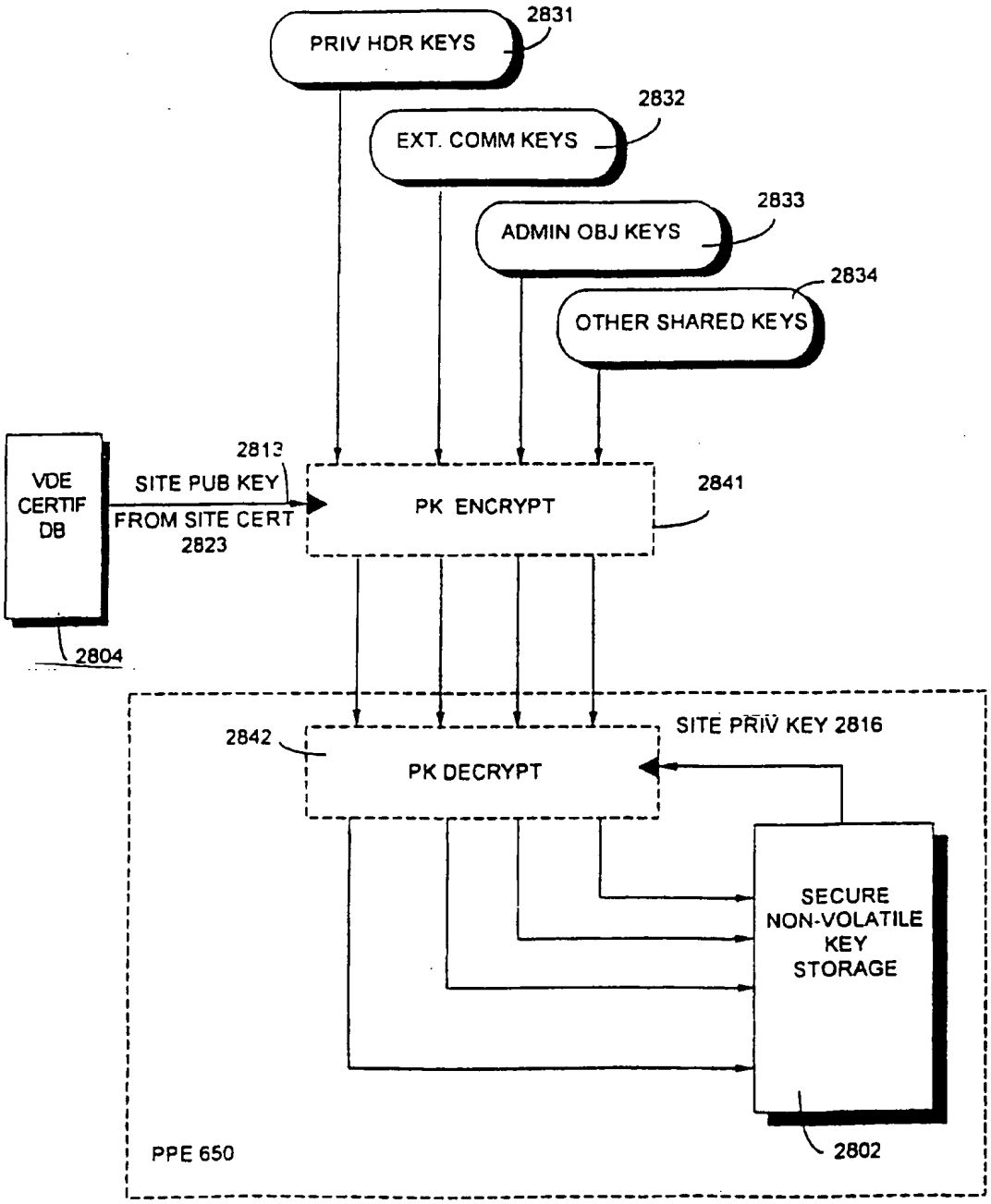


FIG. 65 KEY INSTALLATION & UPDATE



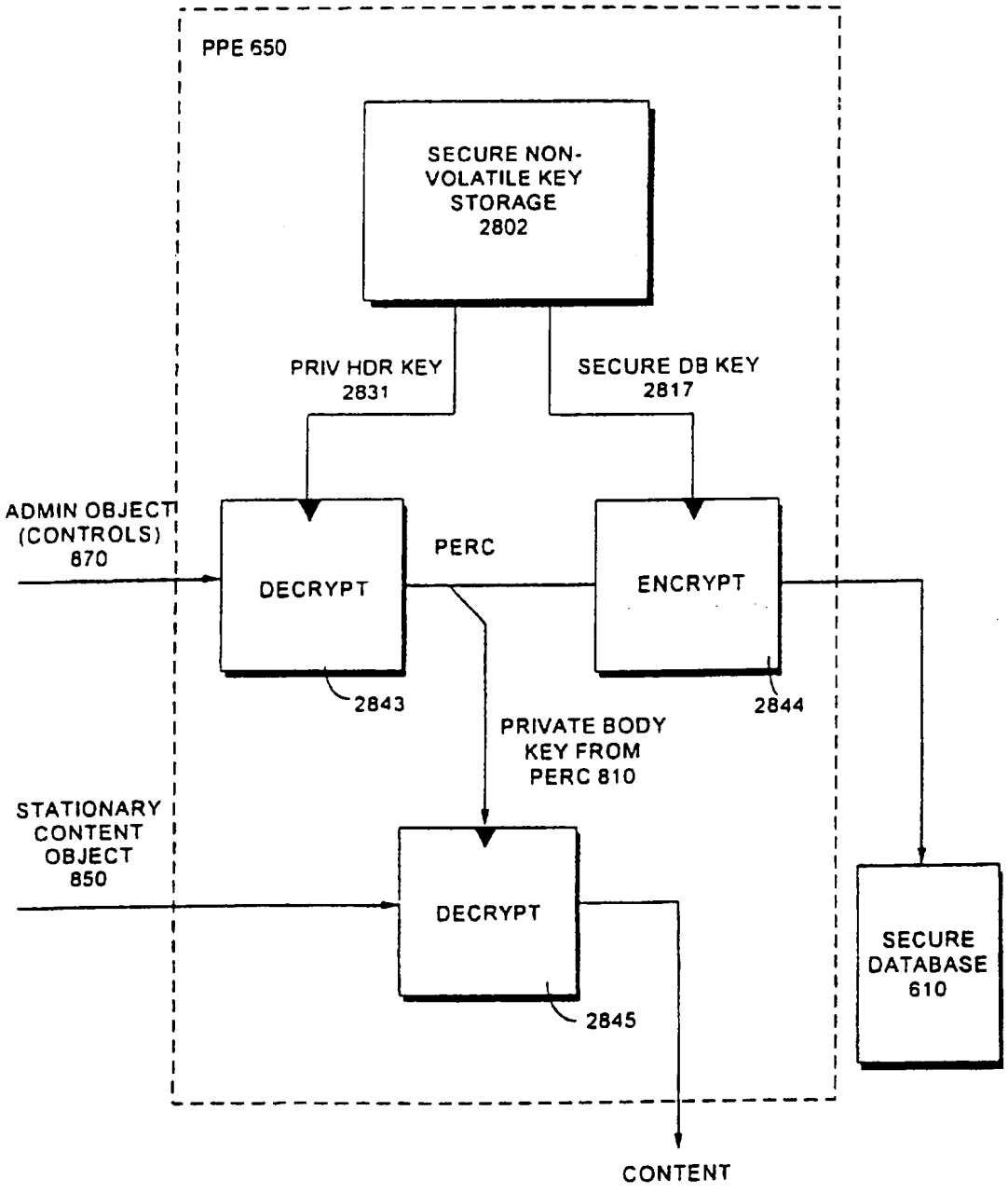


FIG. 66 STATIONARY OBJECT DECRYPTION

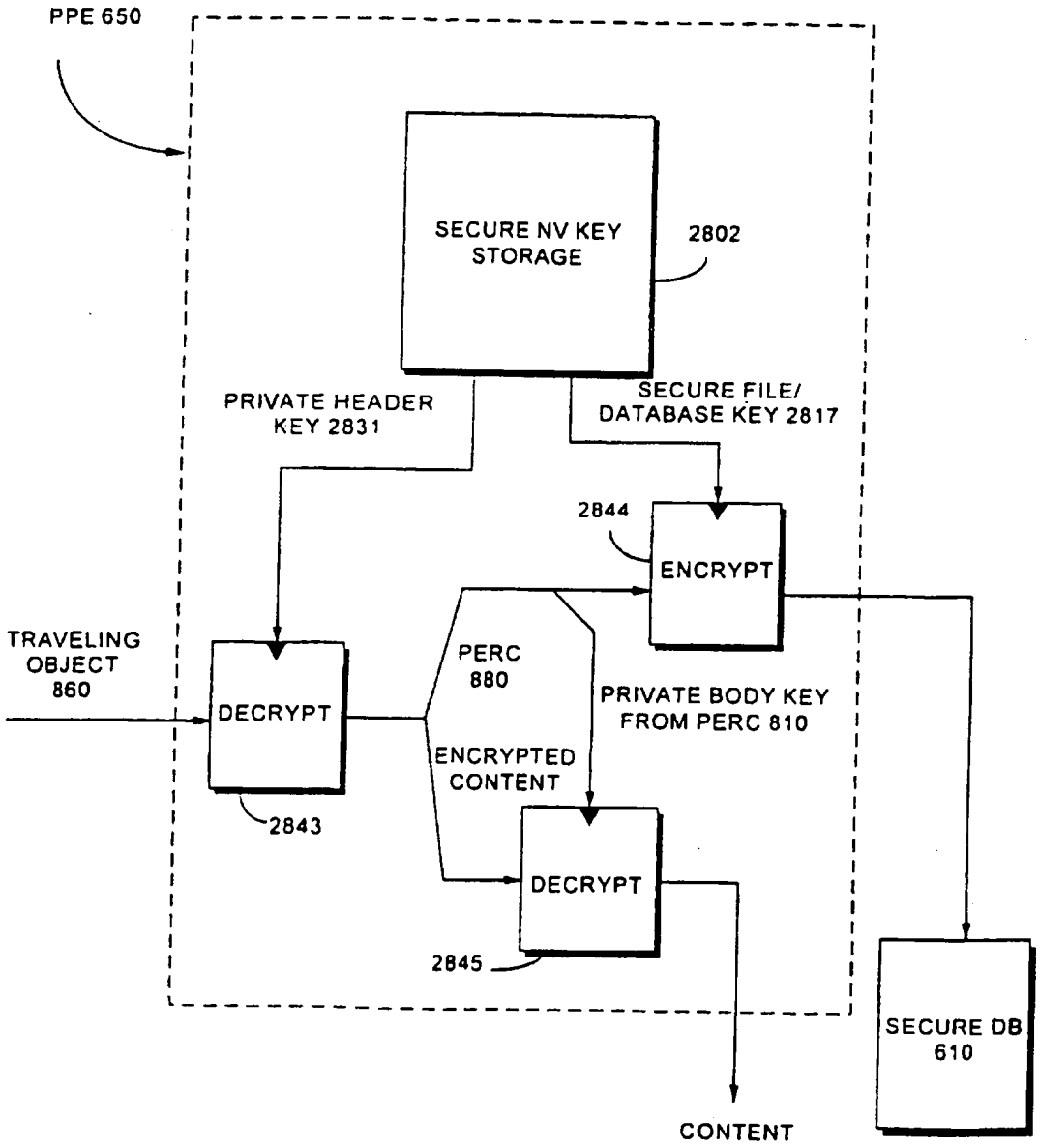


FIG. 67 TRAVELING OBJECT DECRYPTION

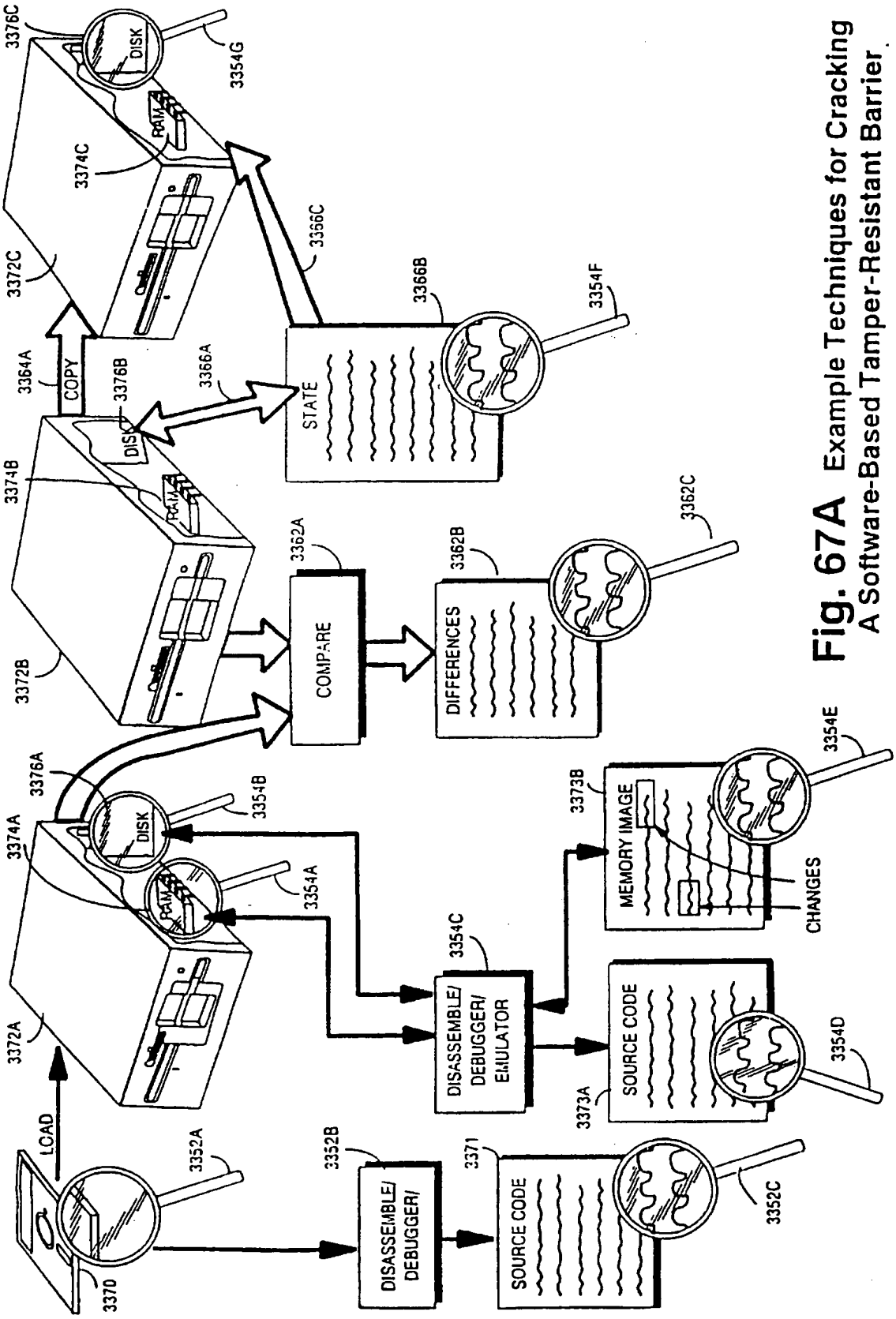
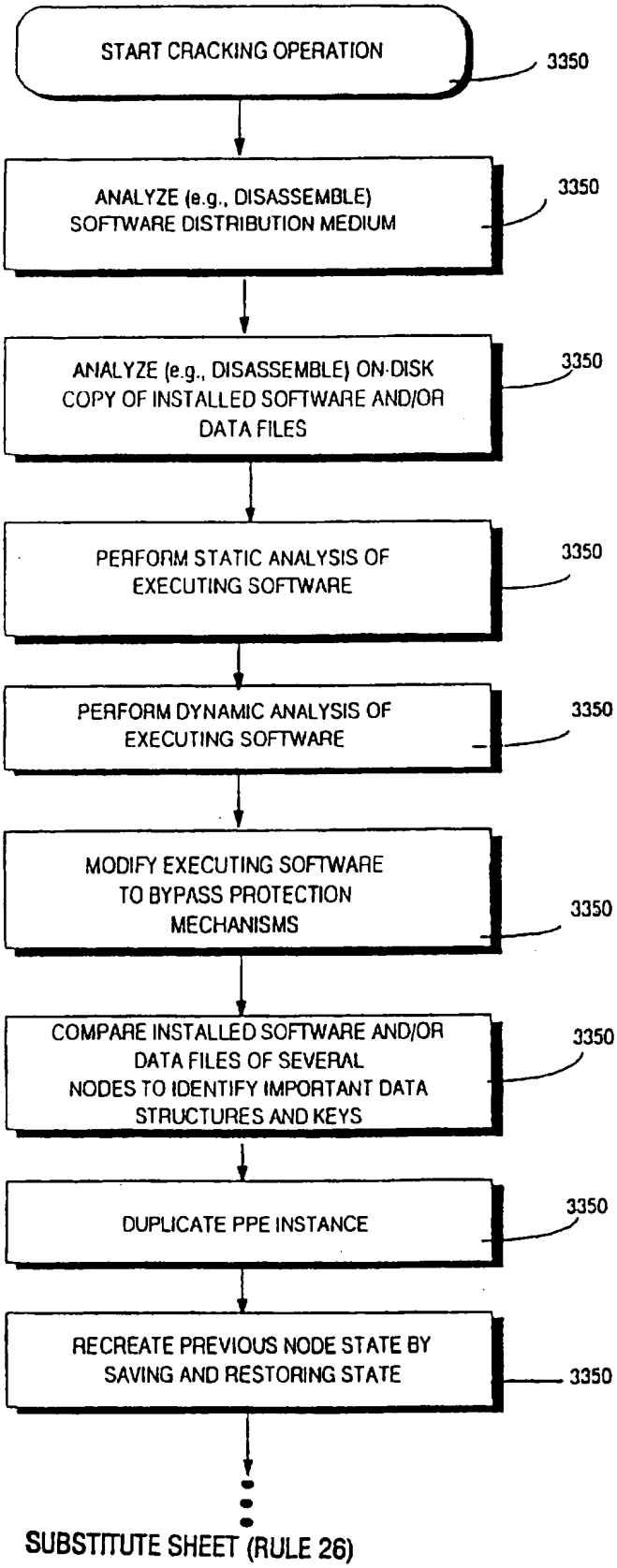


Fig. 67A Example Techniques for Cracking A Software-Based Tamper-Resistant Barrier

Fig. 67B
Example Cracking Operation



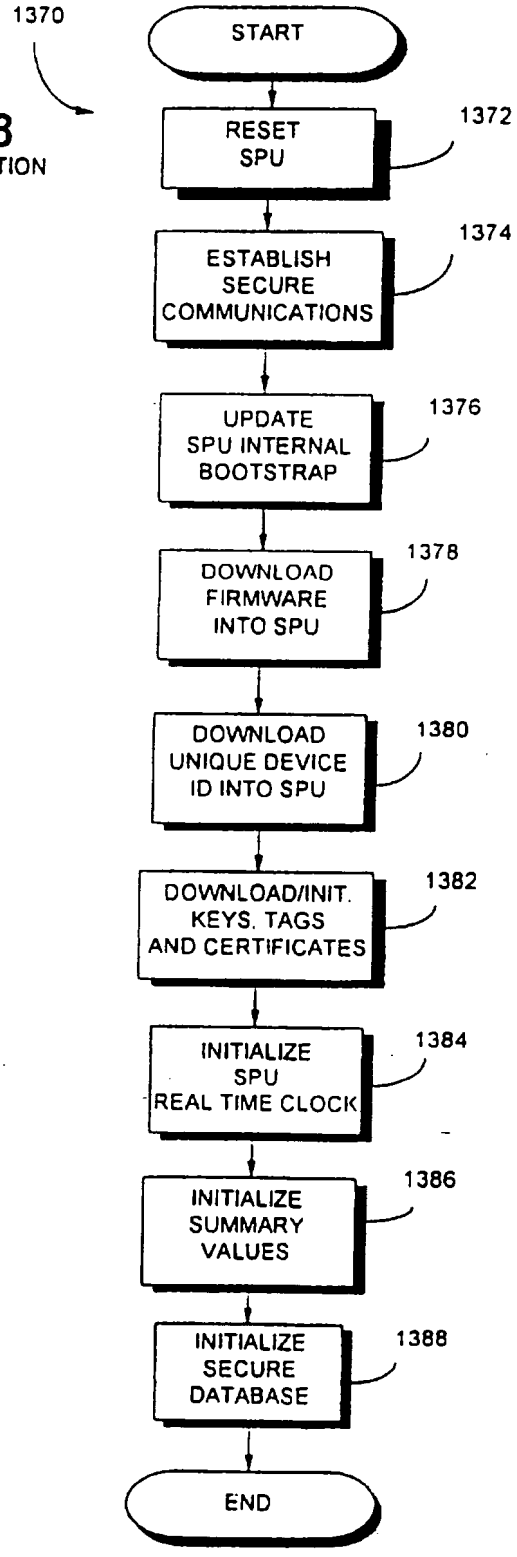


FIG. 68
SPU INITIALIZATION

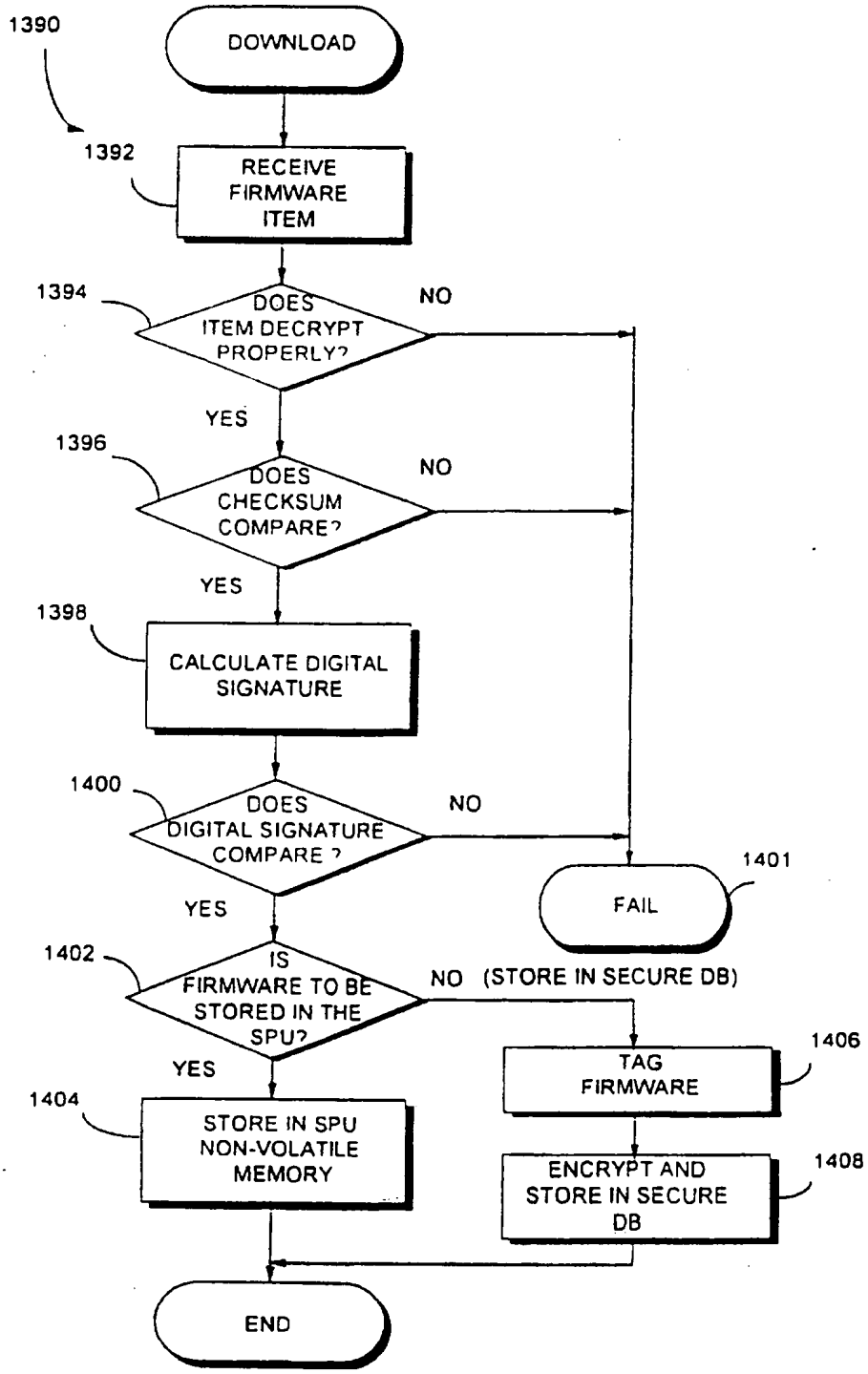


FIG. 69
SPU FIRMWARE DOWNLOAD

Fig. 69A Software Distributed In Encrypted Form

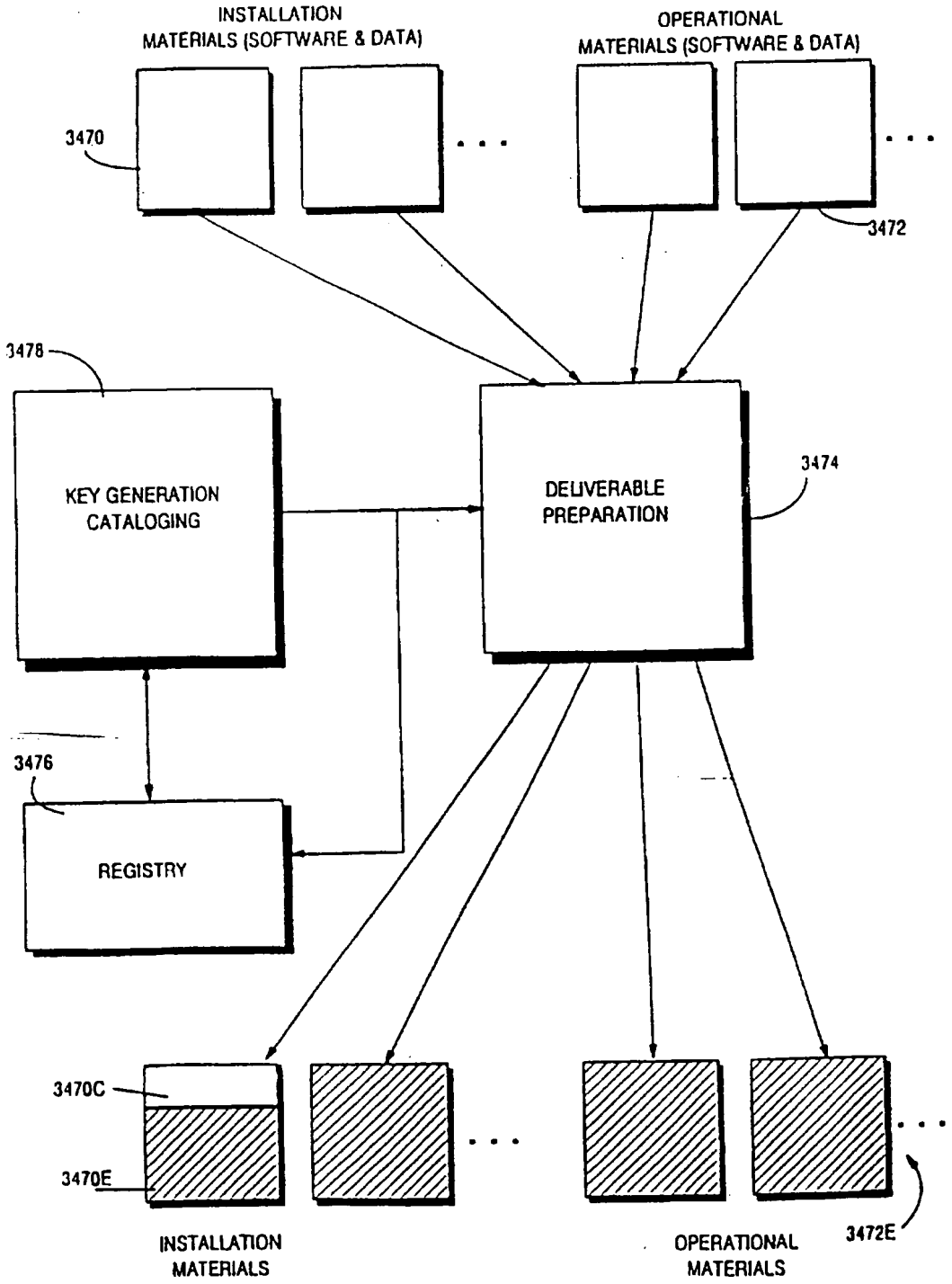


Fig. 69B Example Installation Routine

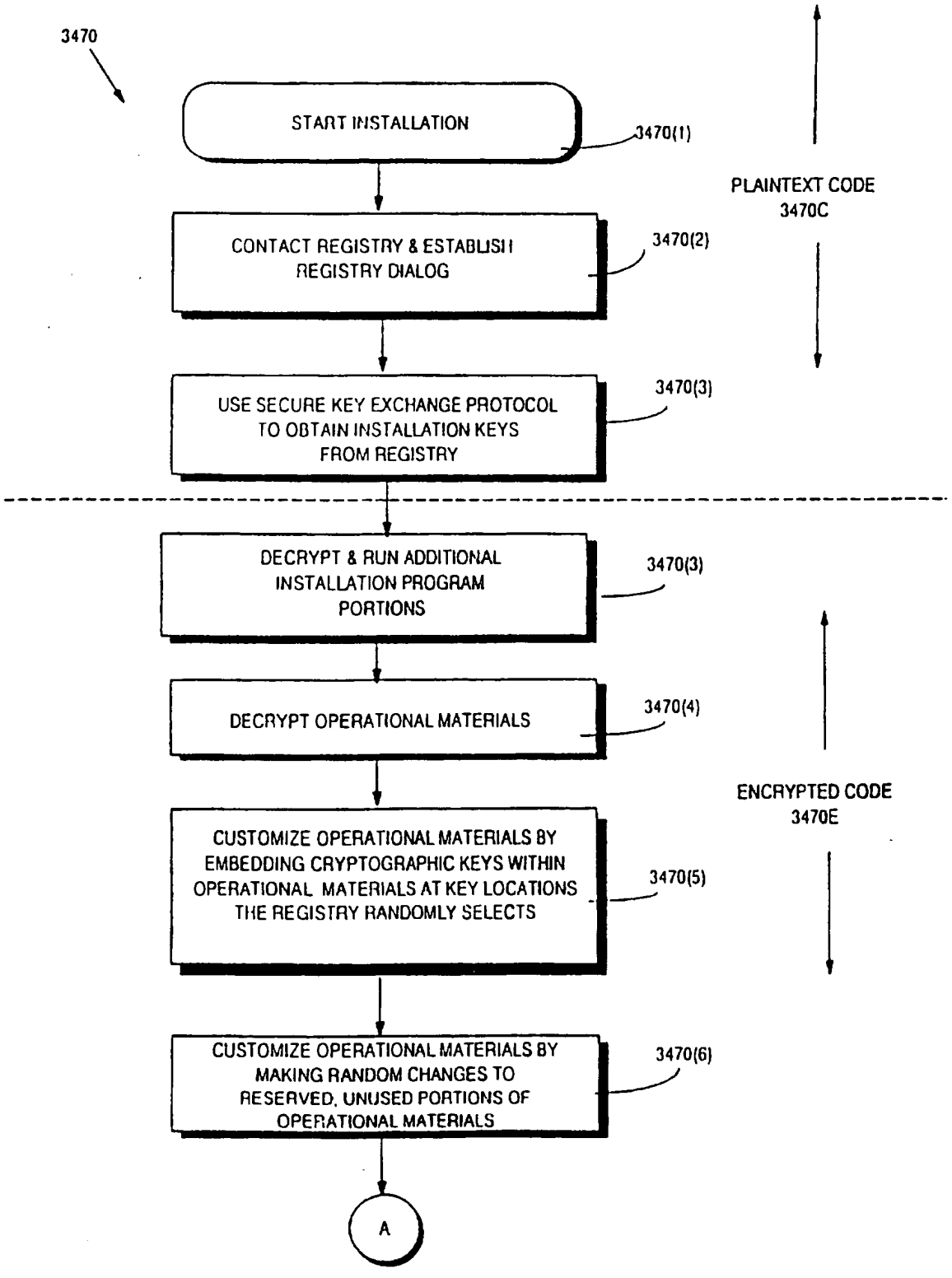
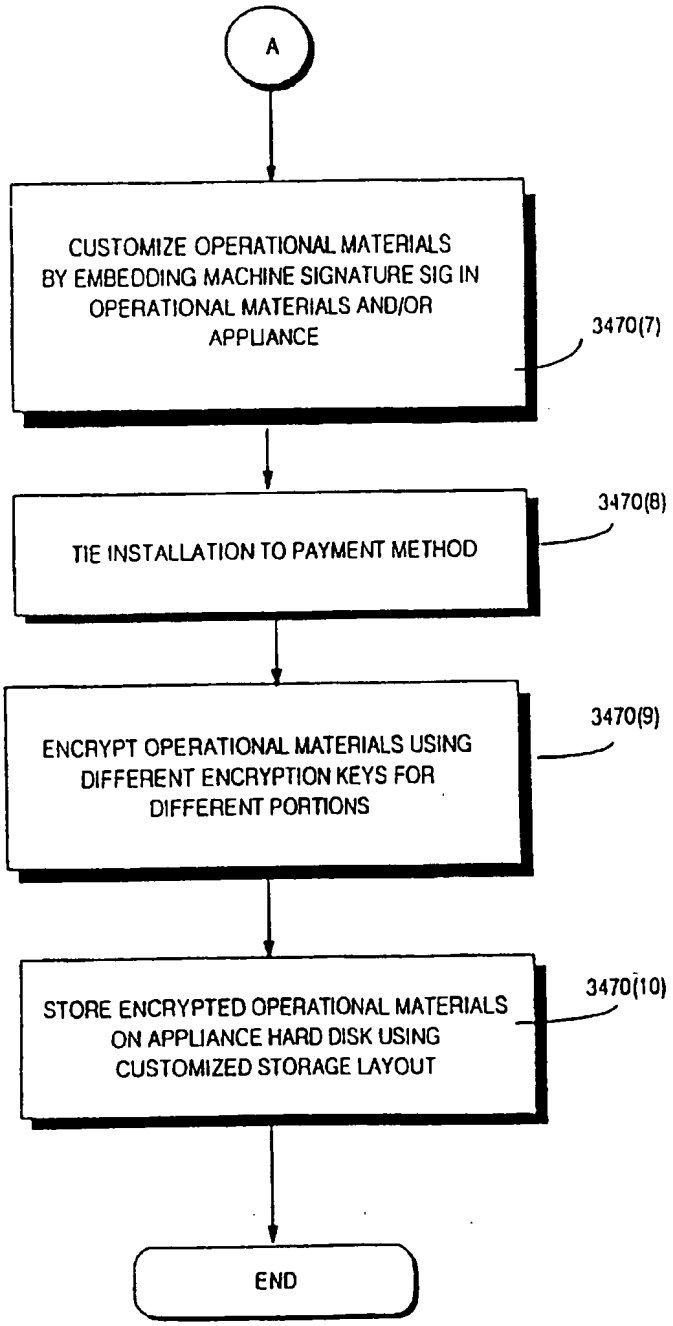


Fig. 69C Example Installation Routine



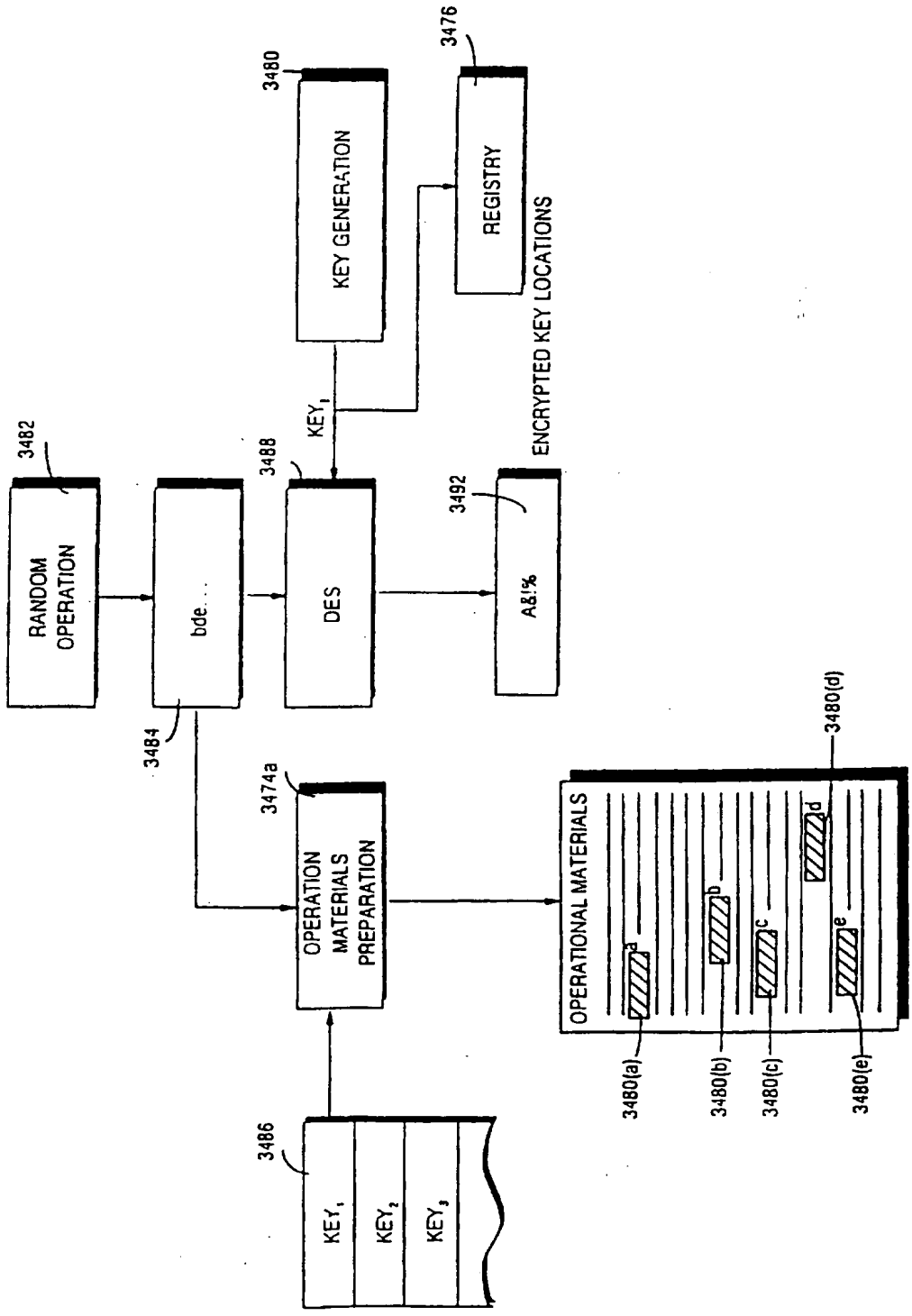


Fig. 69D Embedding Keys At Different Locations With Operational Materials

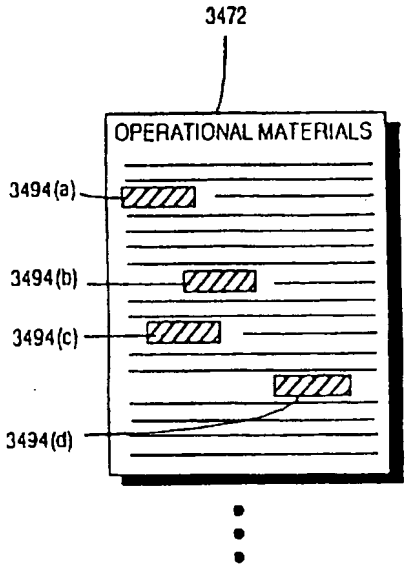


Fig. 69E
Example Locations For Random Modifications and Fingerprints

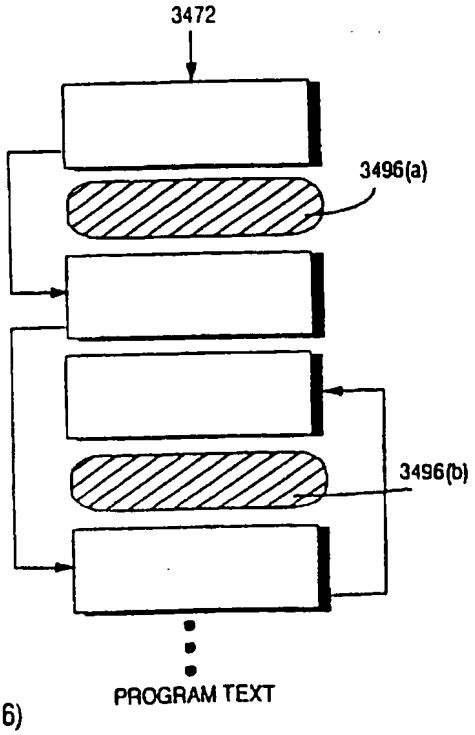


Fig. 69F
Example Customized Static Storage Layout

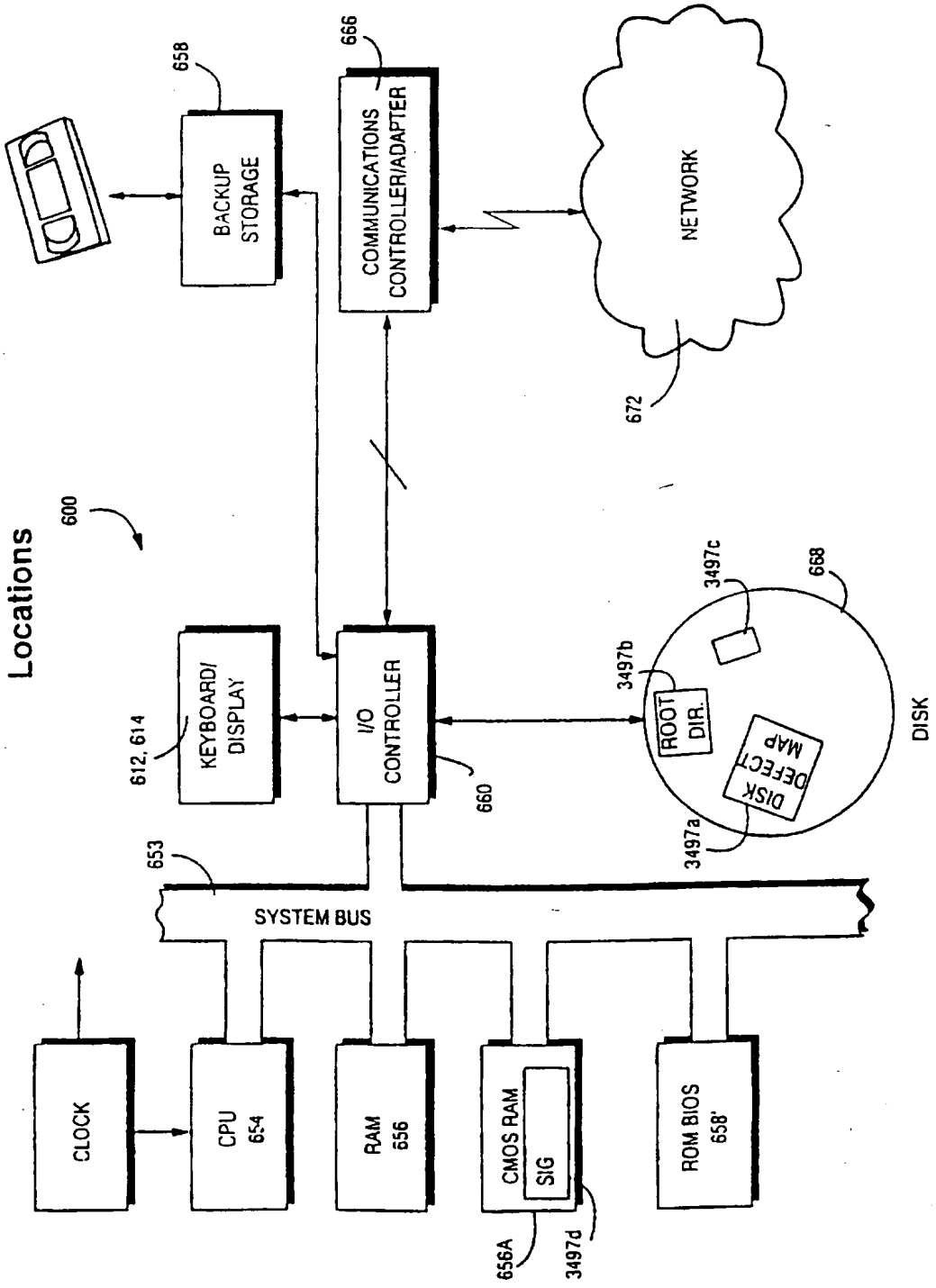
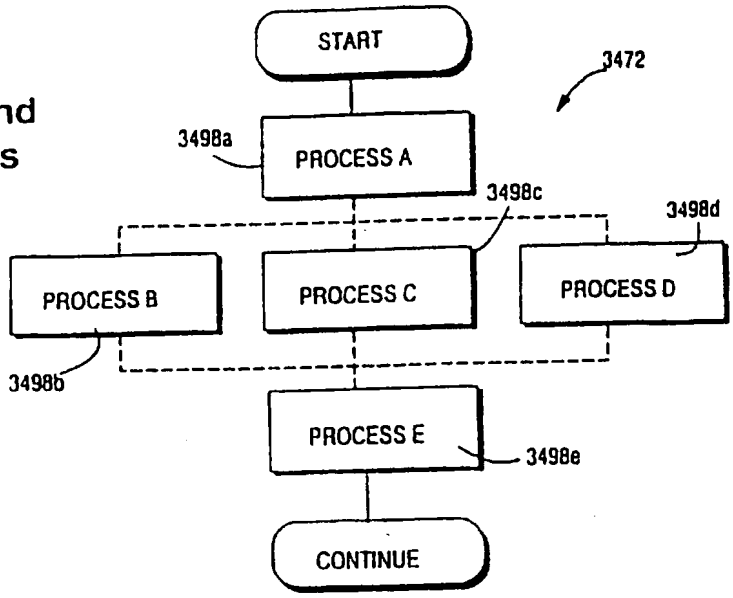


Fig. 69G
Example Machine Signature Locations

Fig. 69H
Sequence Dependent and Independent Processes



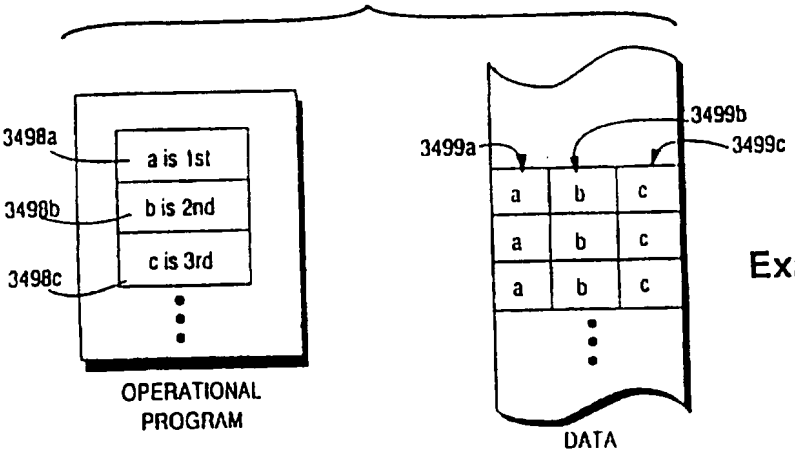


Fig. 69I
Example First Order

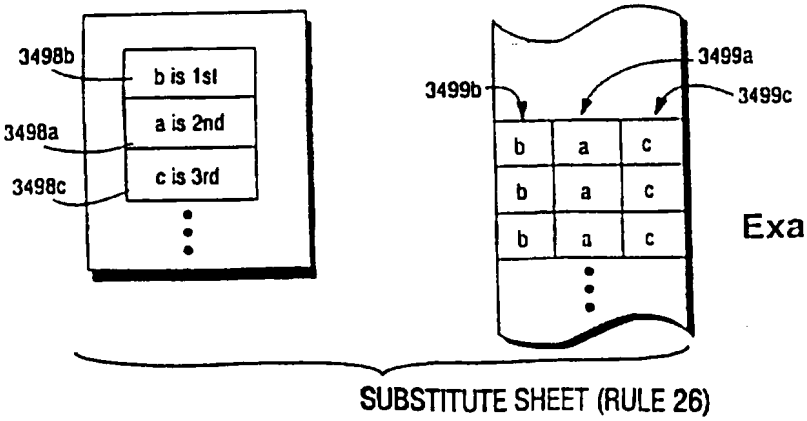


Fig. 69J
Example Second Order

Fig. 69K
Example Dynamic Protection Mechanisms

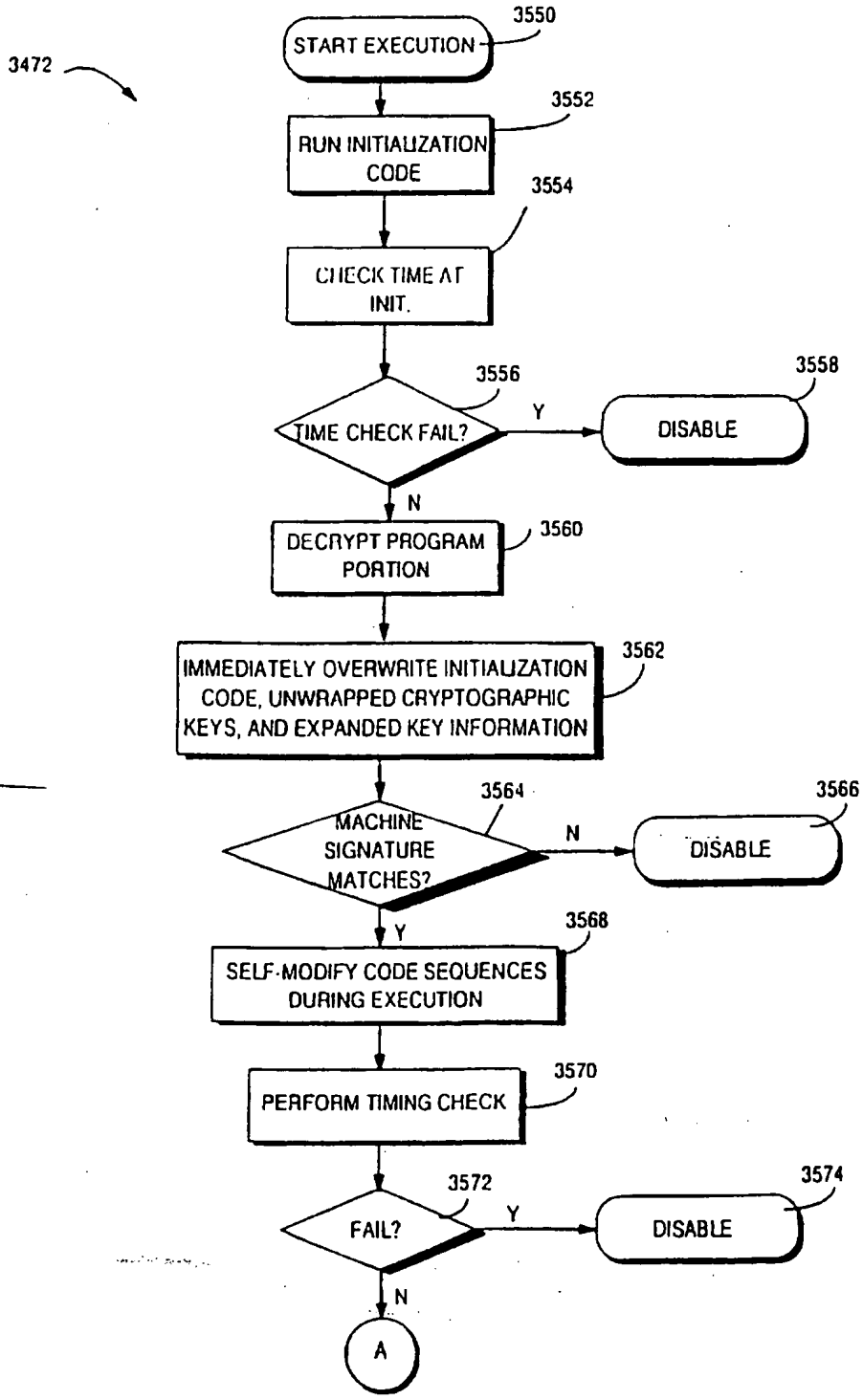


Fig. 69L
Example Dynamic Protection Mechanisms

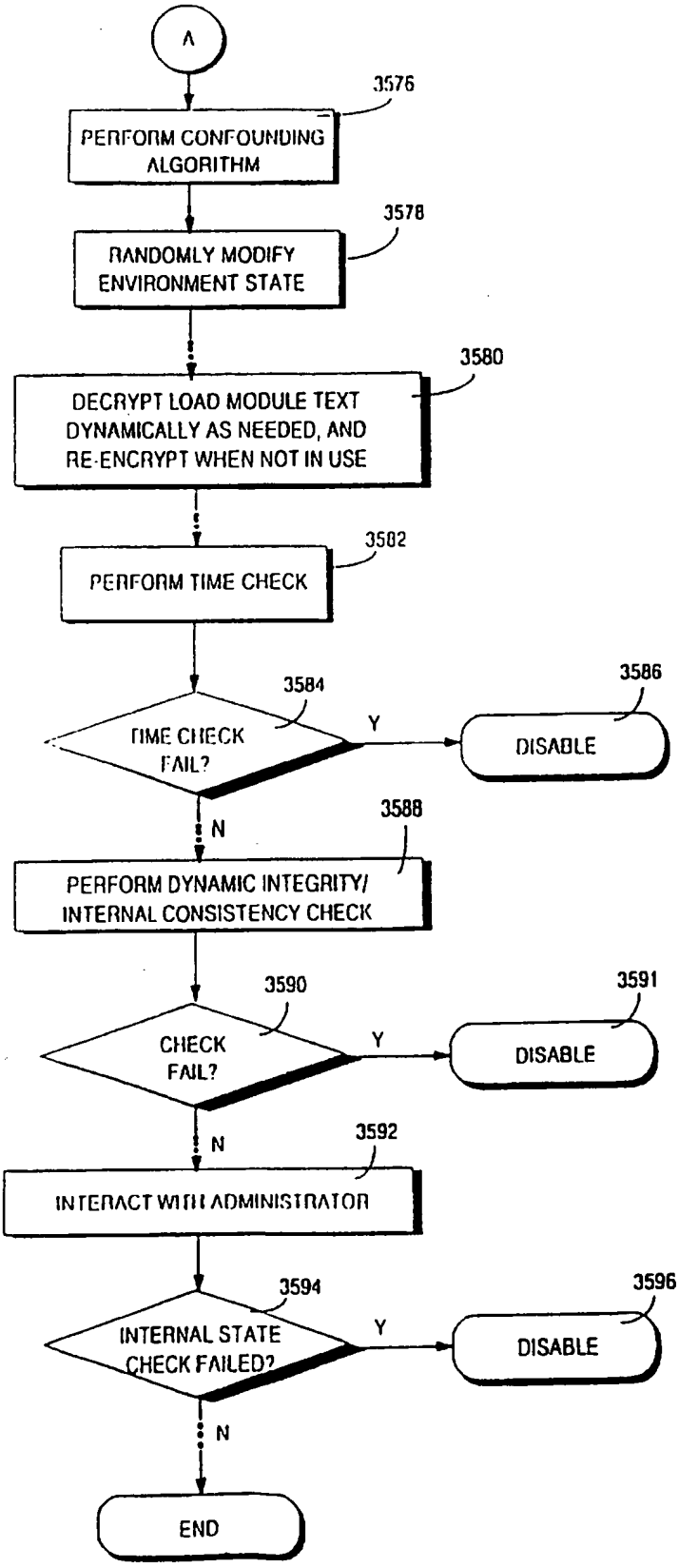
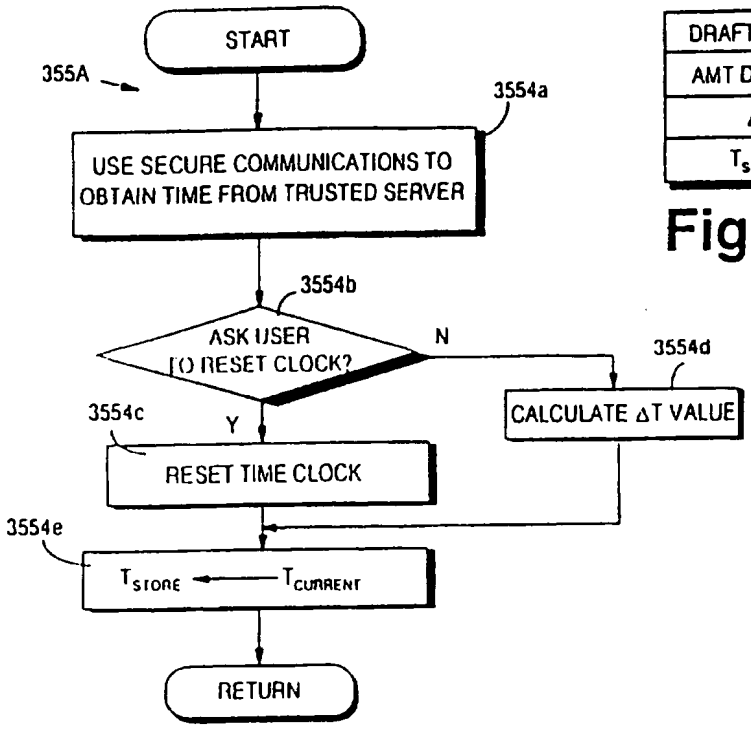


Fig. 69M
Example Time Check At Initialization Routine



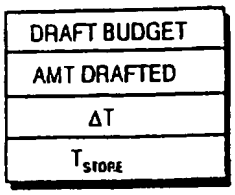
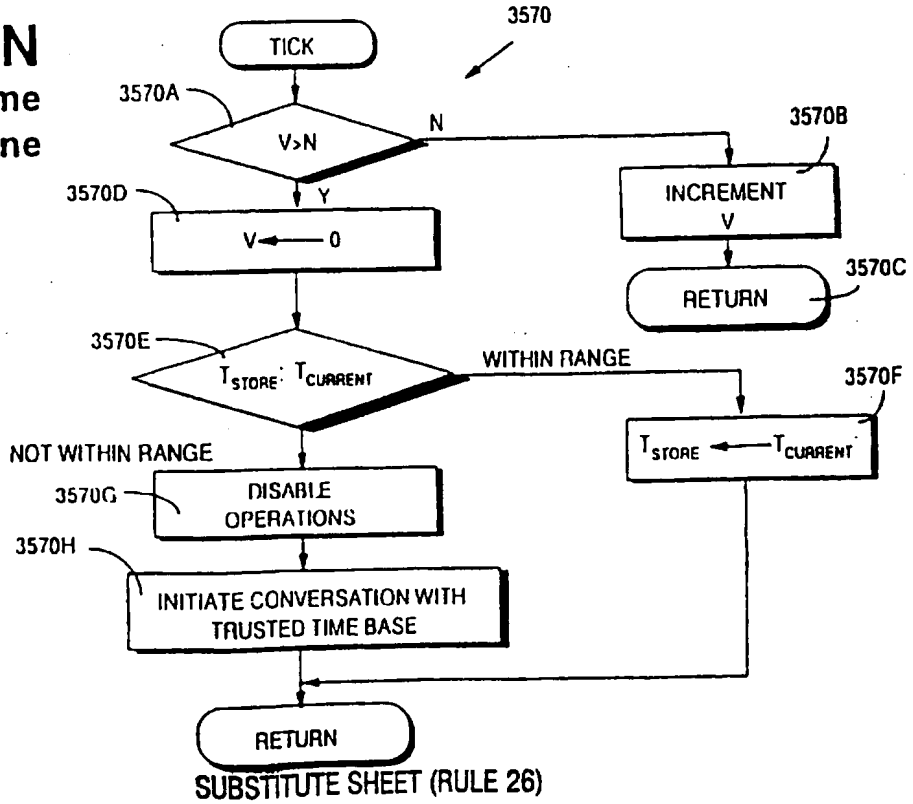


Fig. 69O

Fig. 69N
Example Time Check Routine



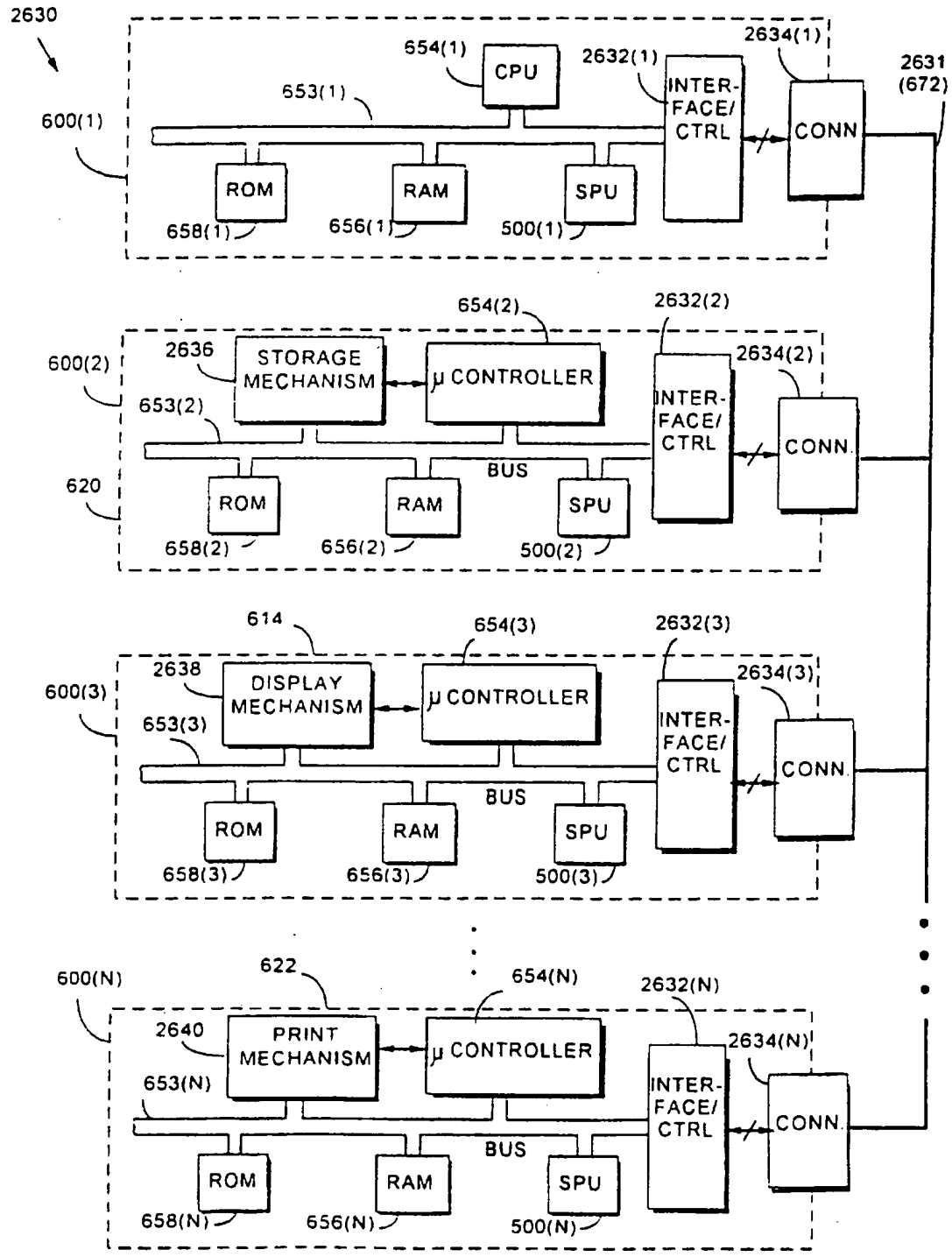


FIG. 70

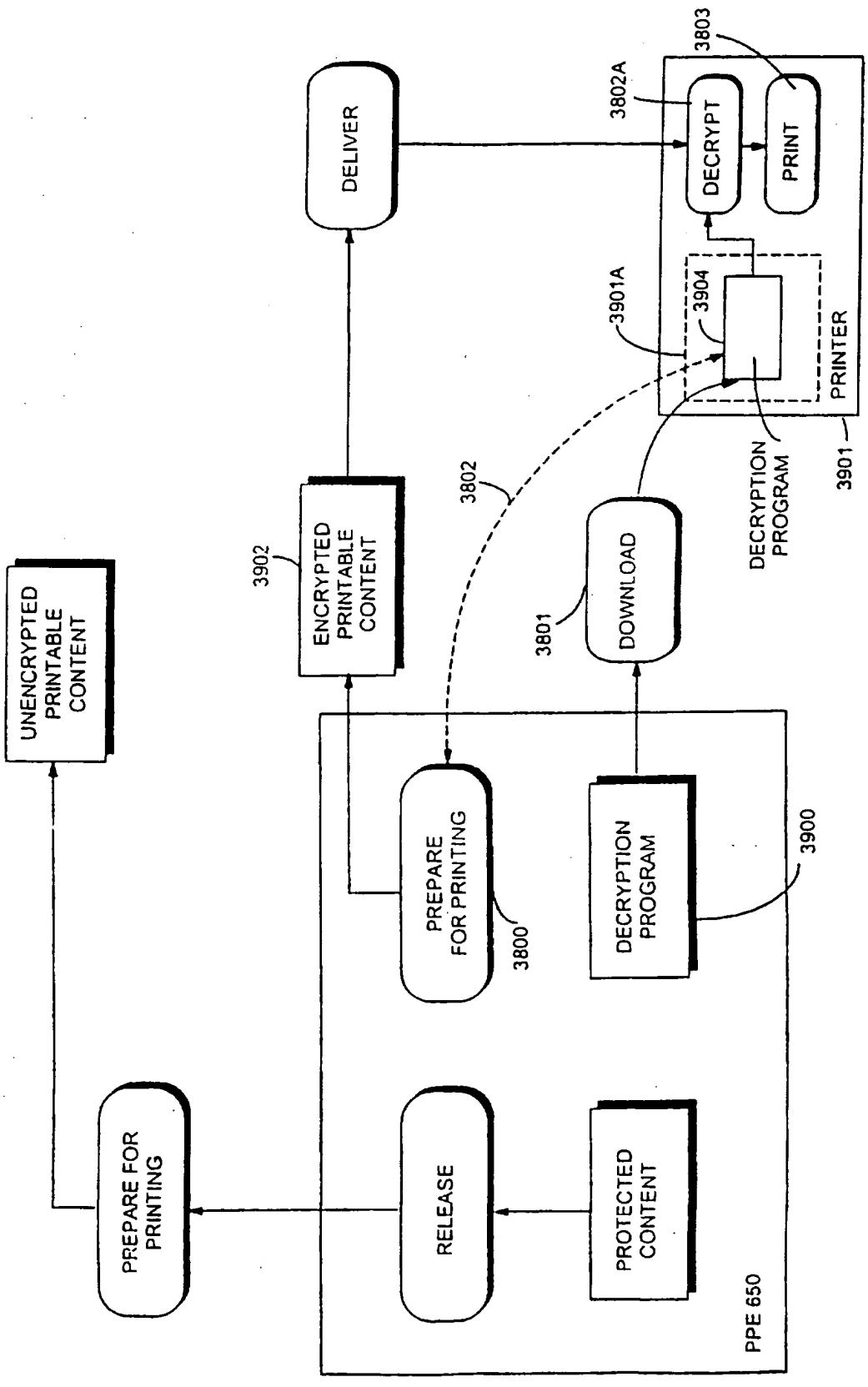


FIG. 70A

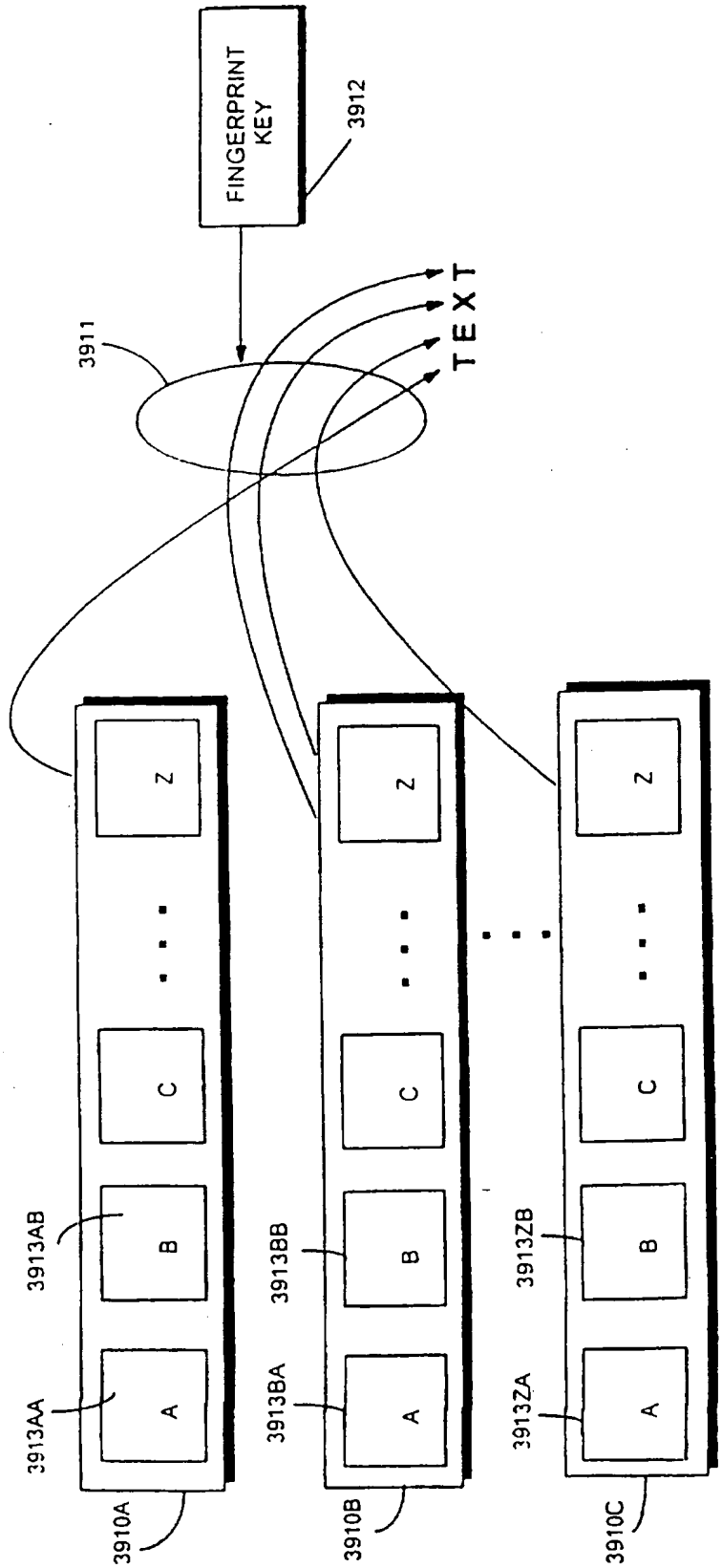
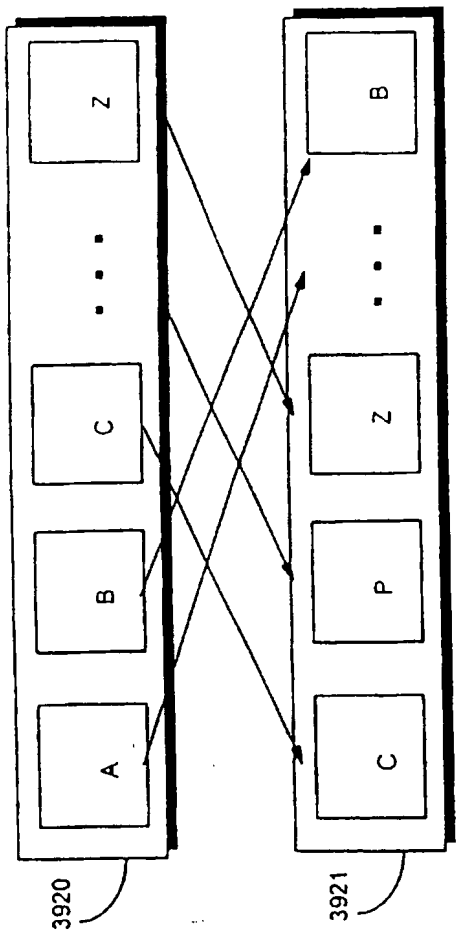


FIG. 70B

FIG. 70C



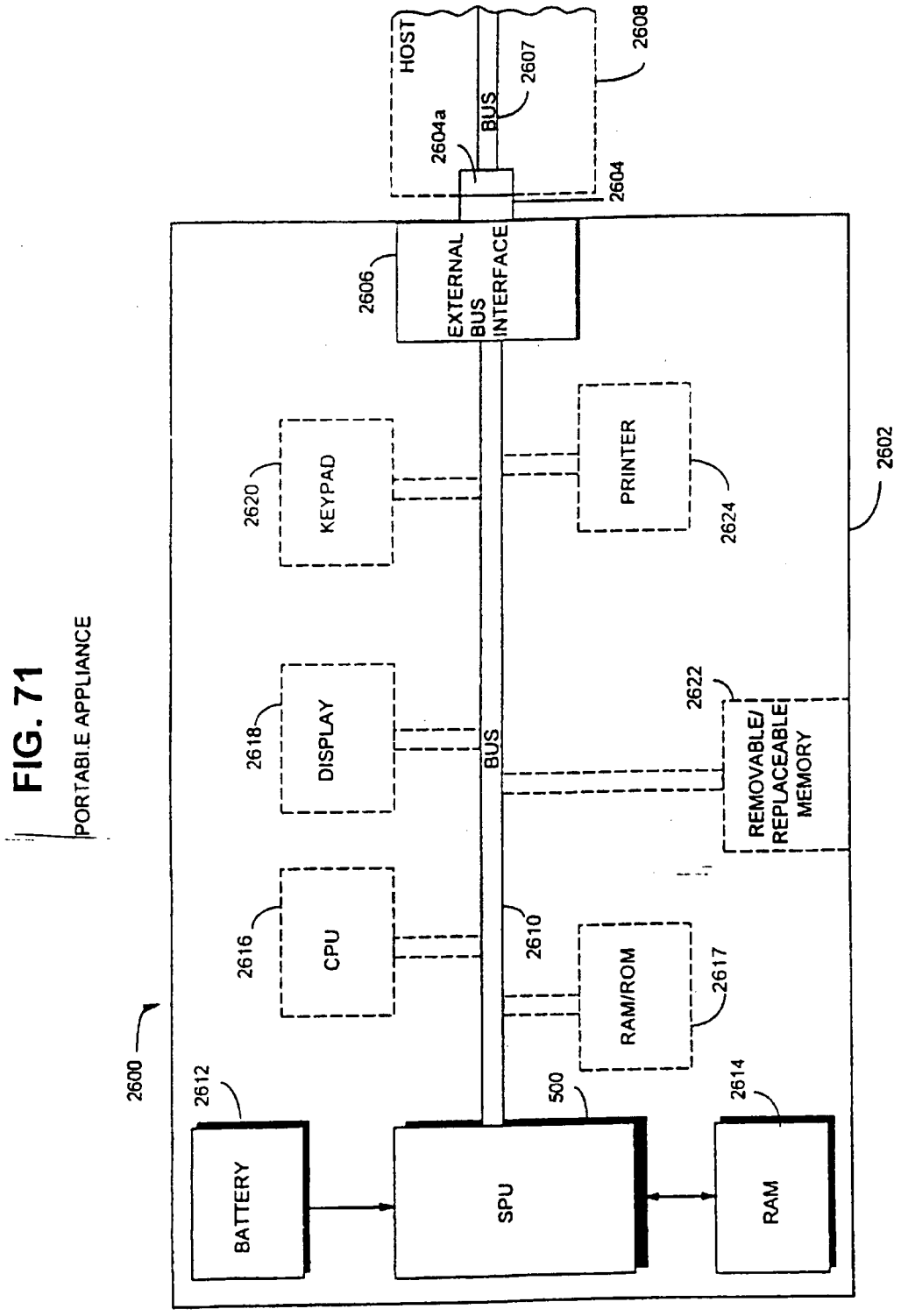


FIG. 71

PORTABLE APPLIANCE

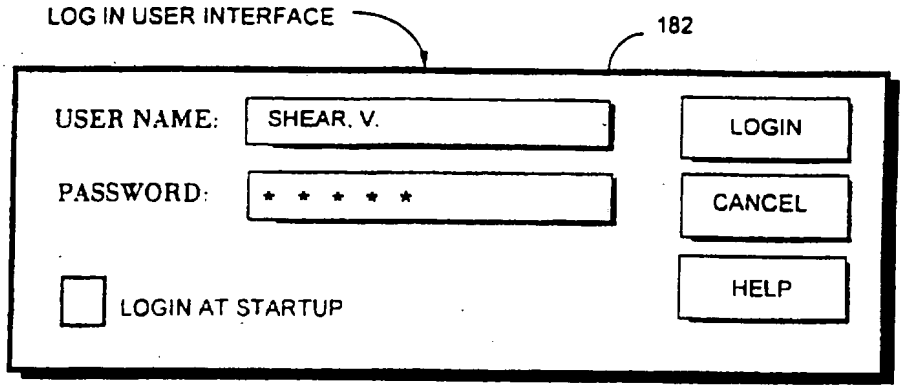


FIG. 72A

FIG. 72B

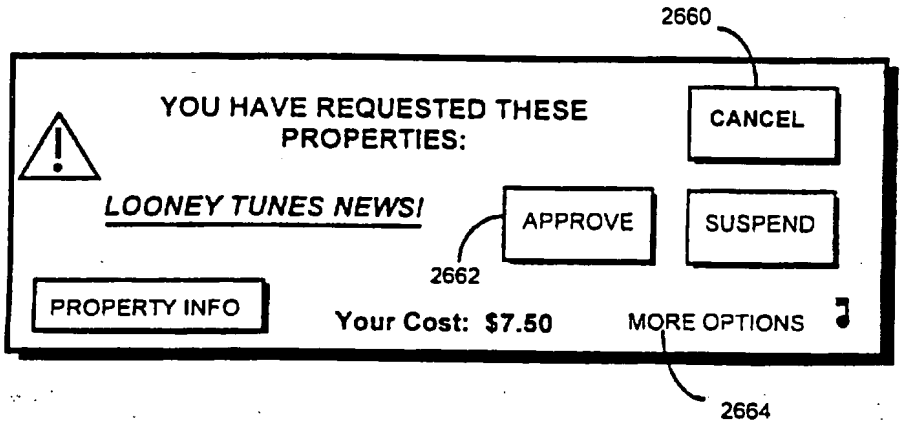


FIG. 72C

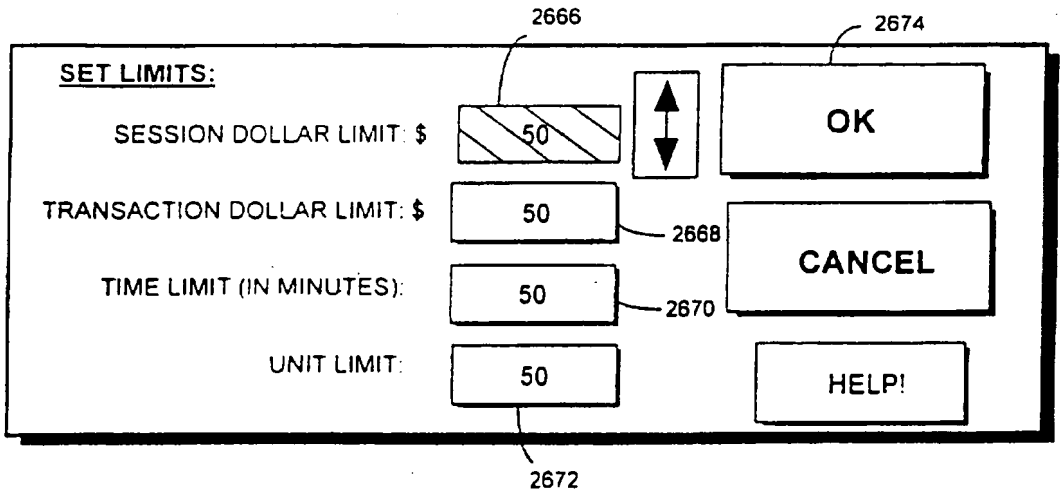


FIG. 72D

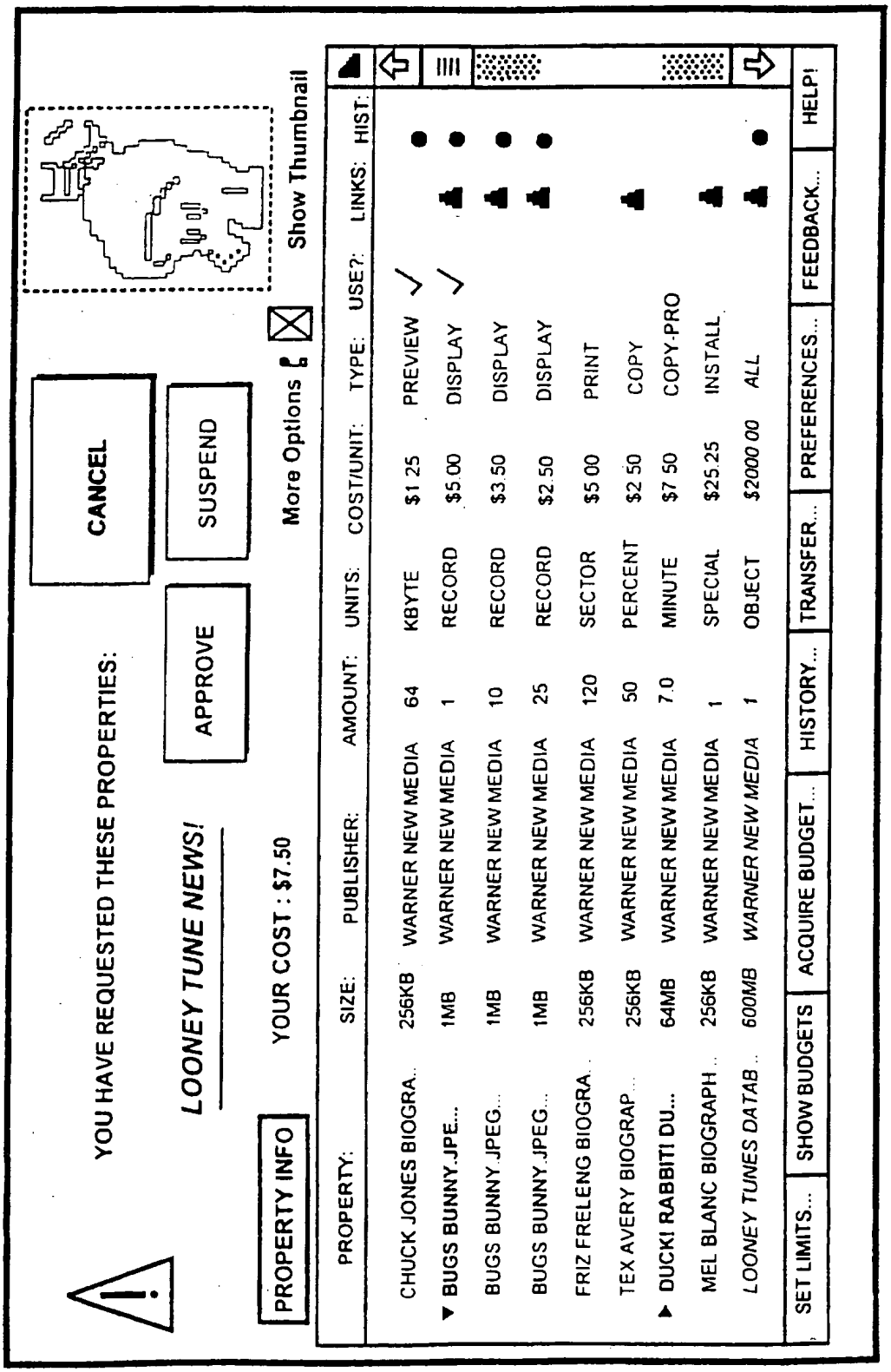


FIG. 73

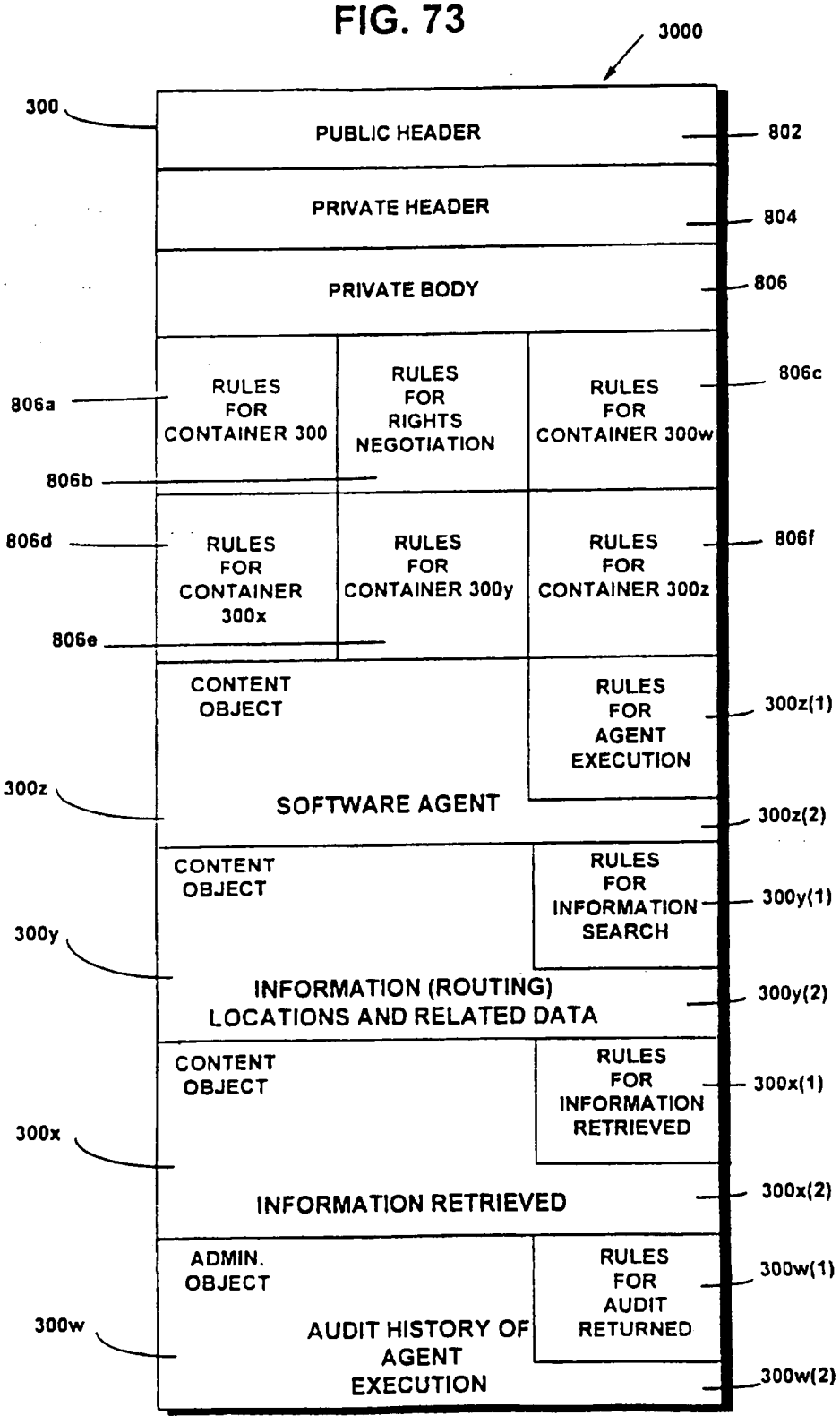


FIG. 74

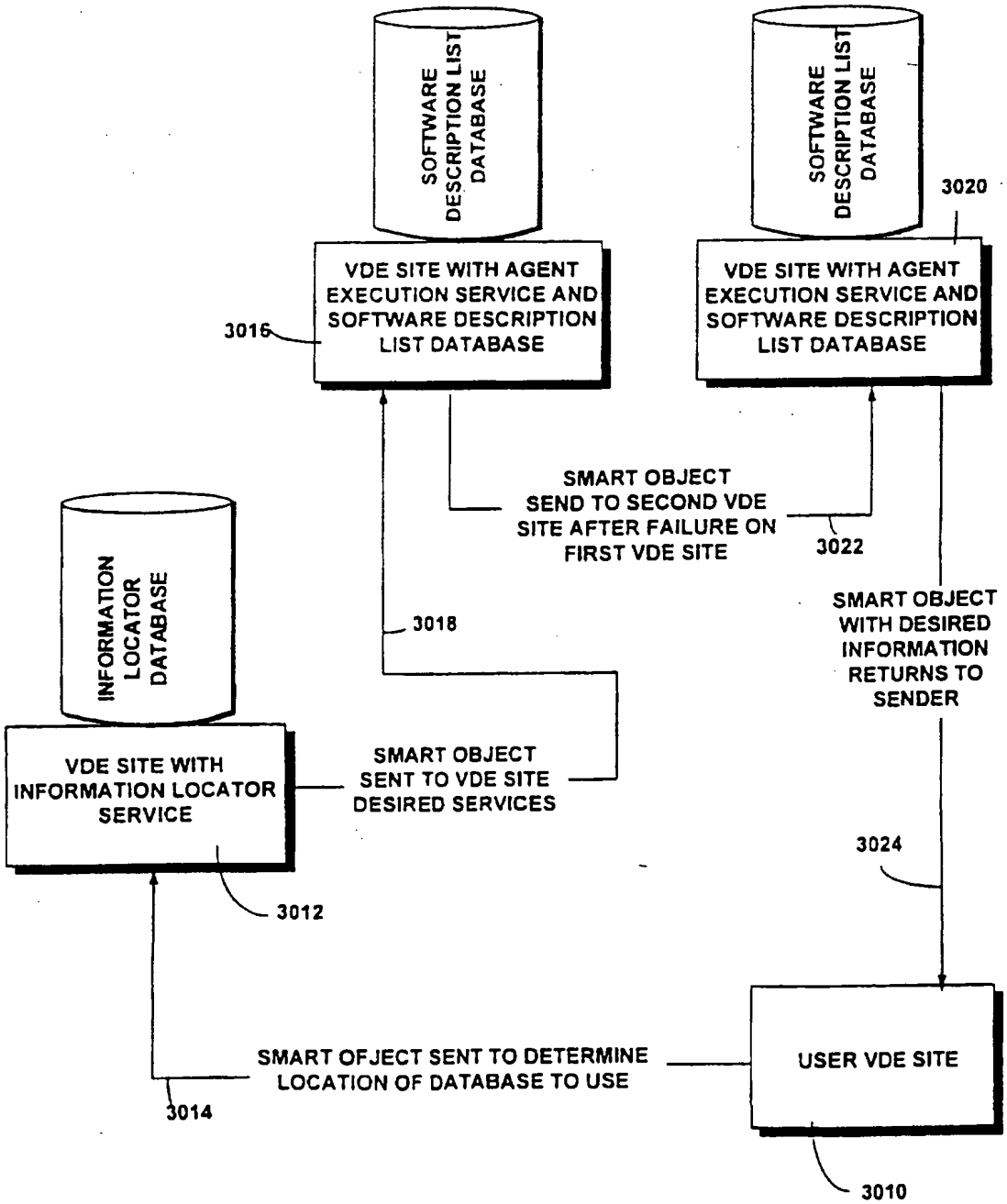


FIG. 75A

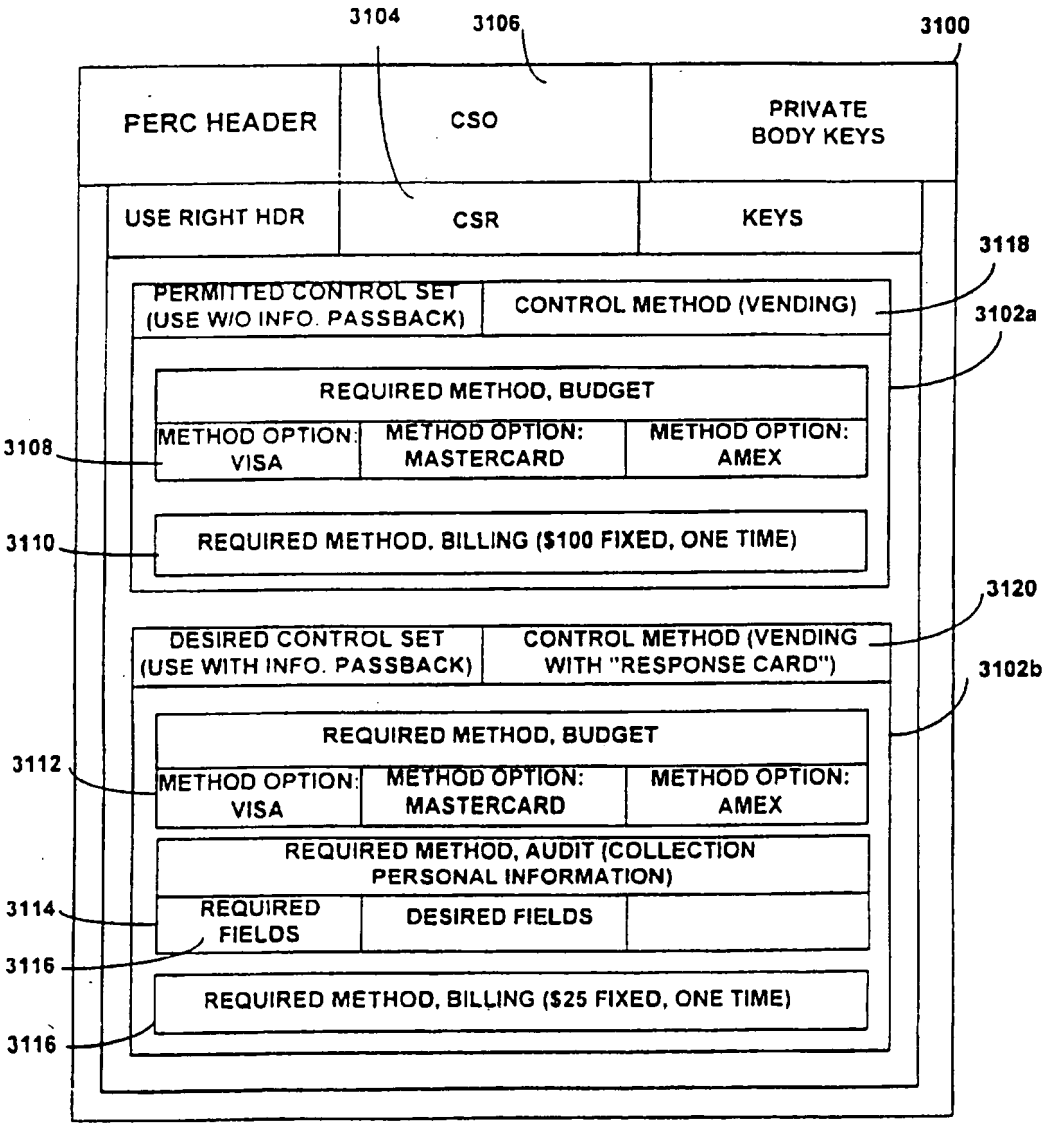


FIG. 75B

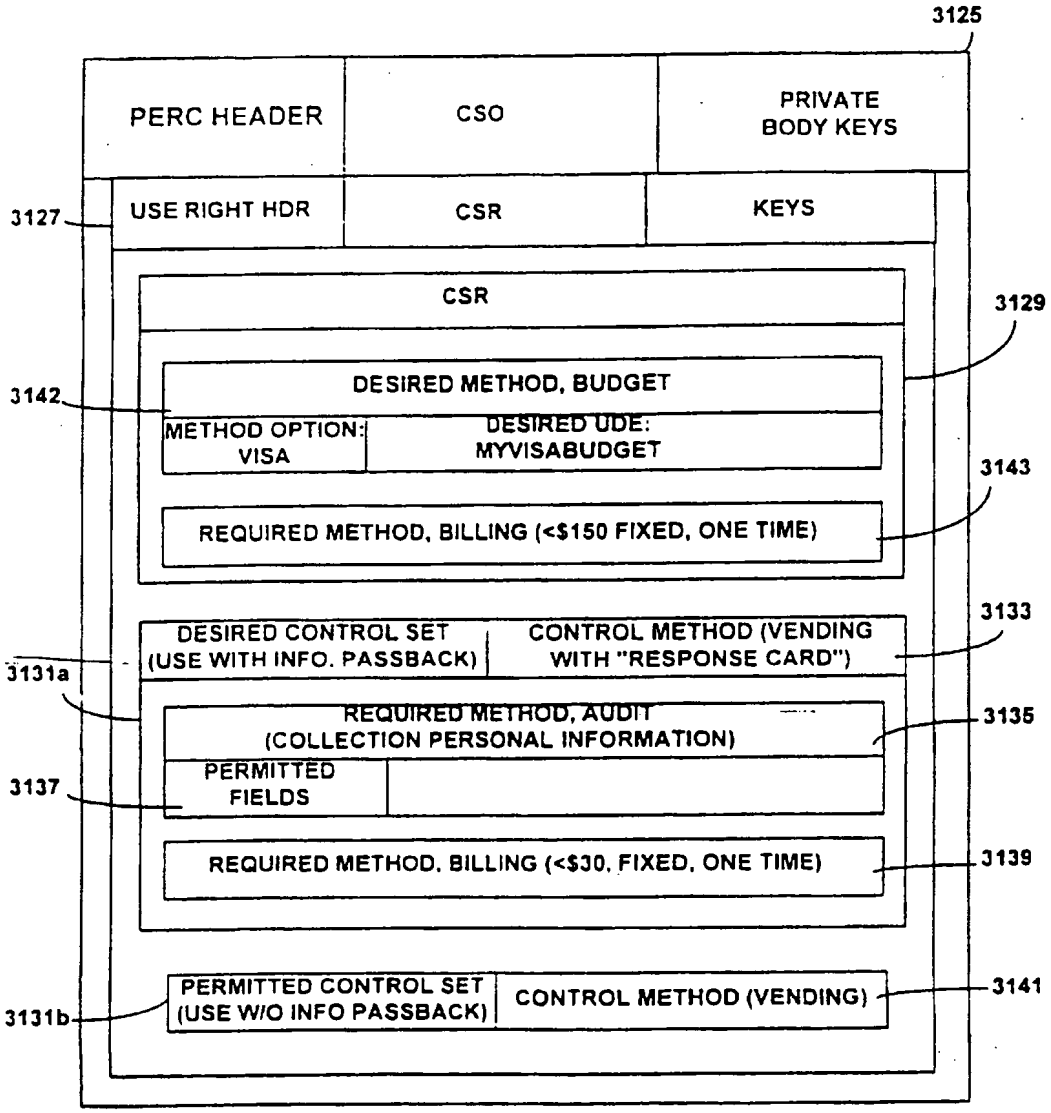


FIG. 75C

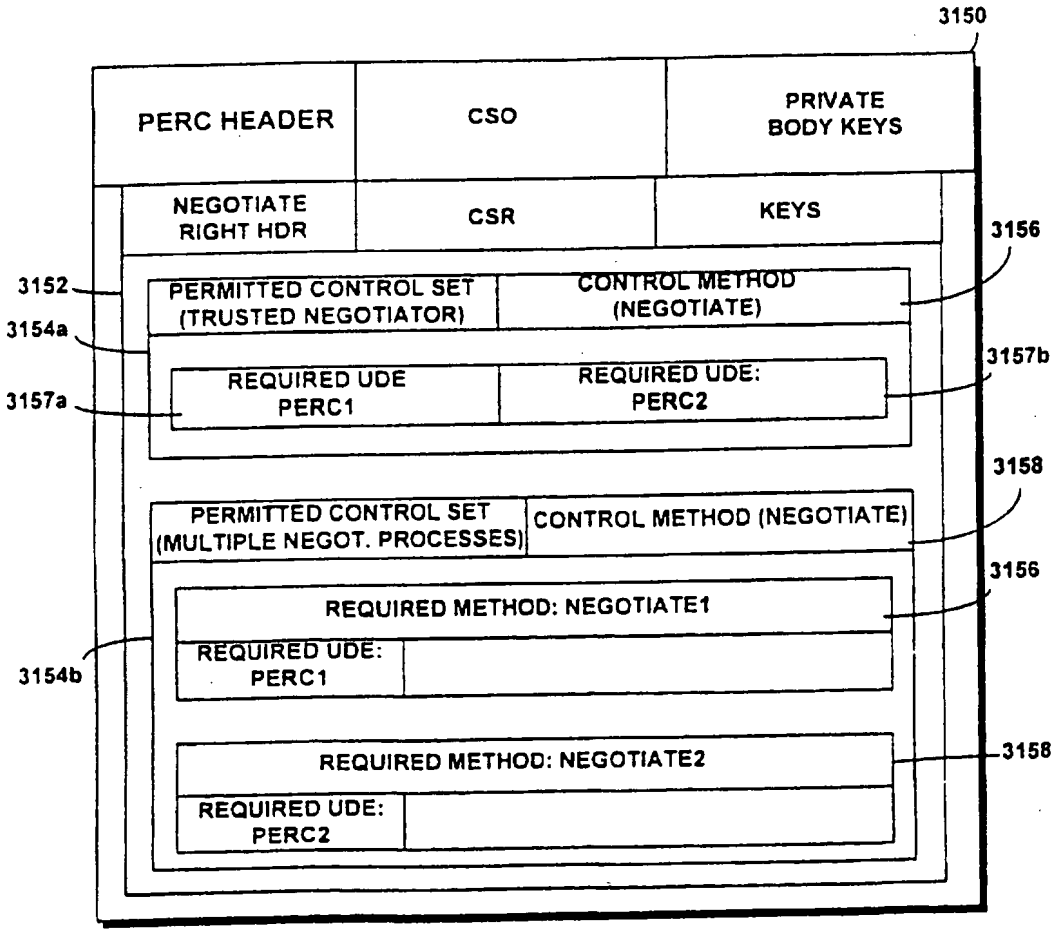
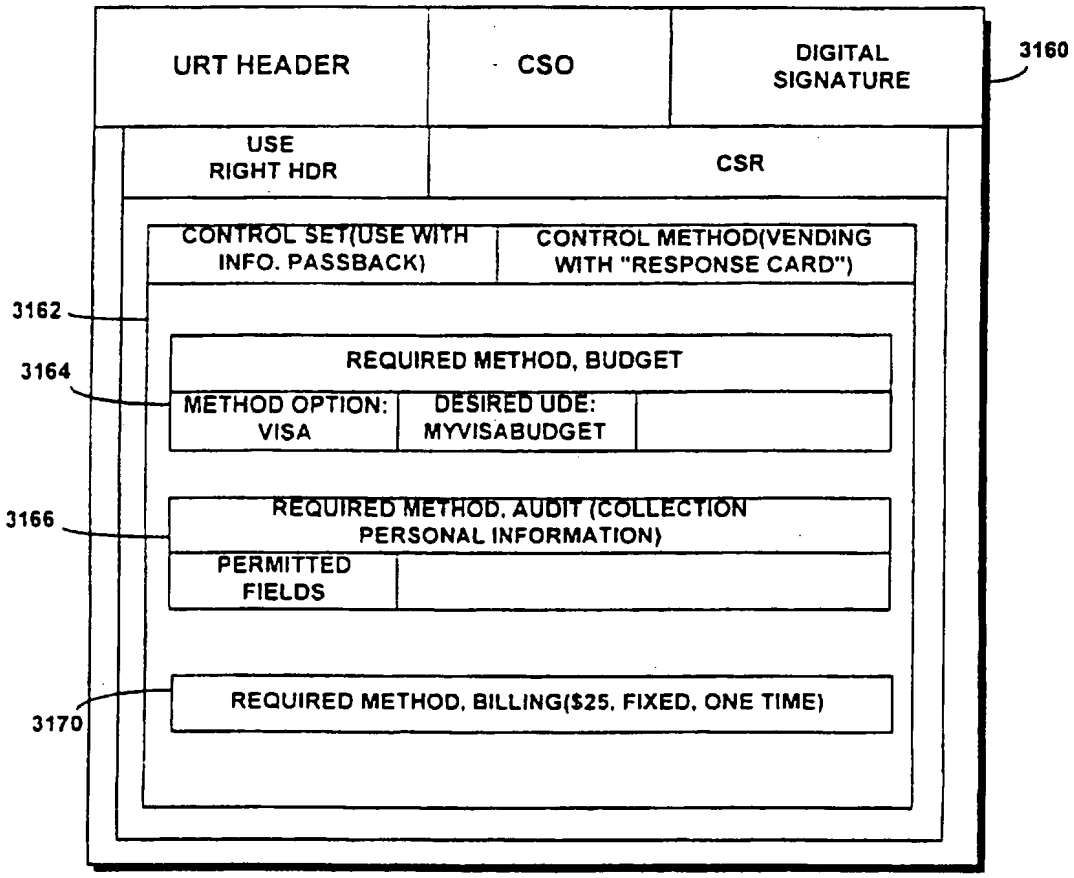


FIG. 75D



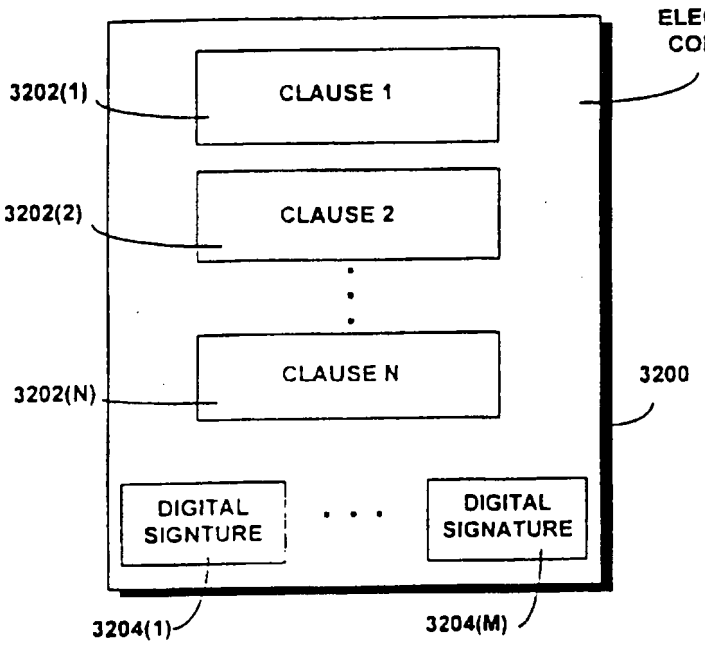


FIG. 75E

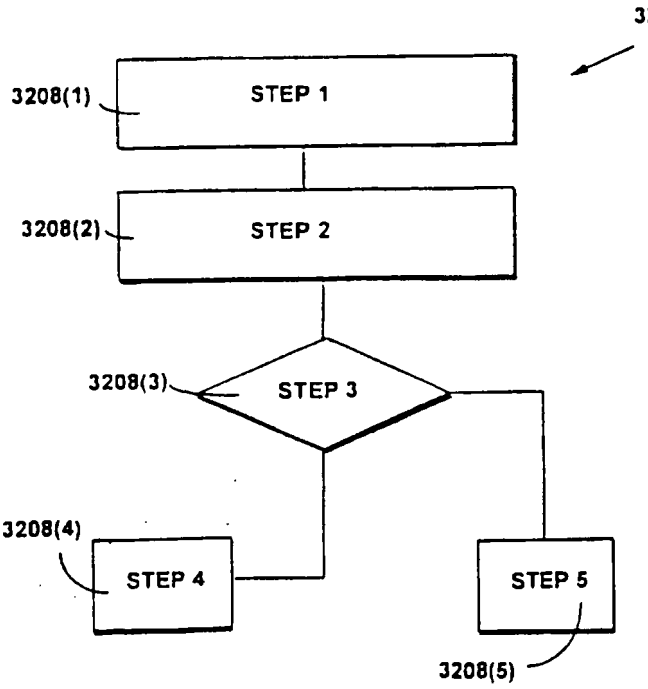


FIG. 75F

FIG. 76A

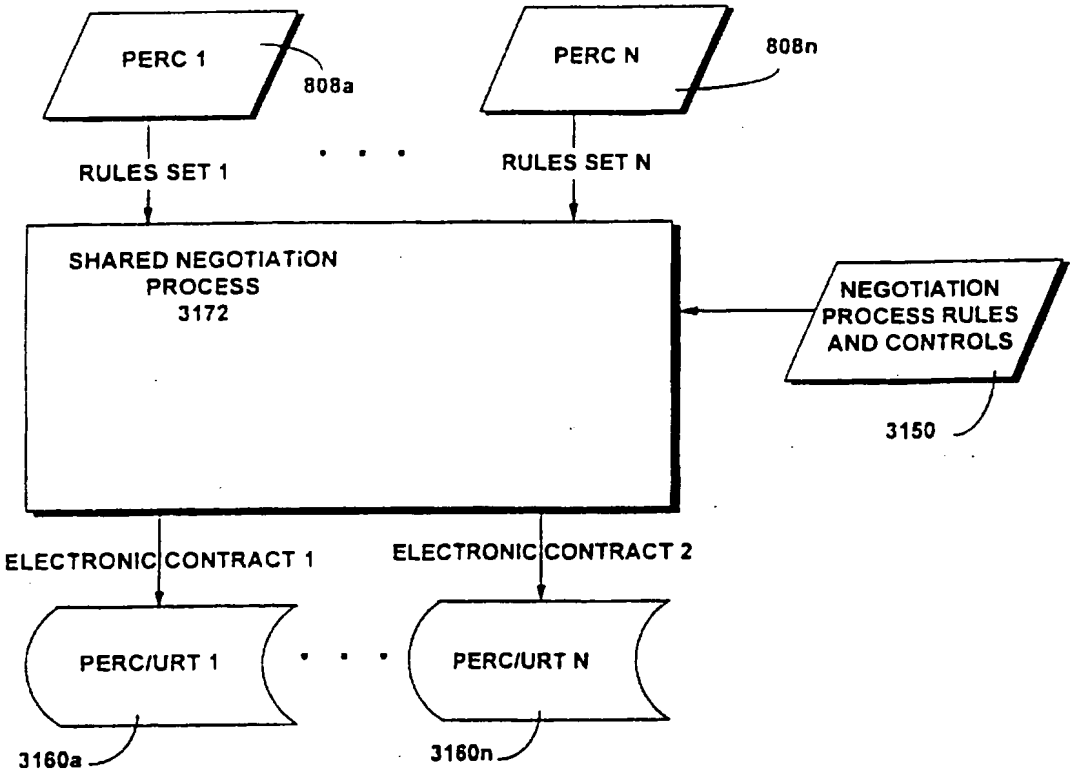


FIG. 76B

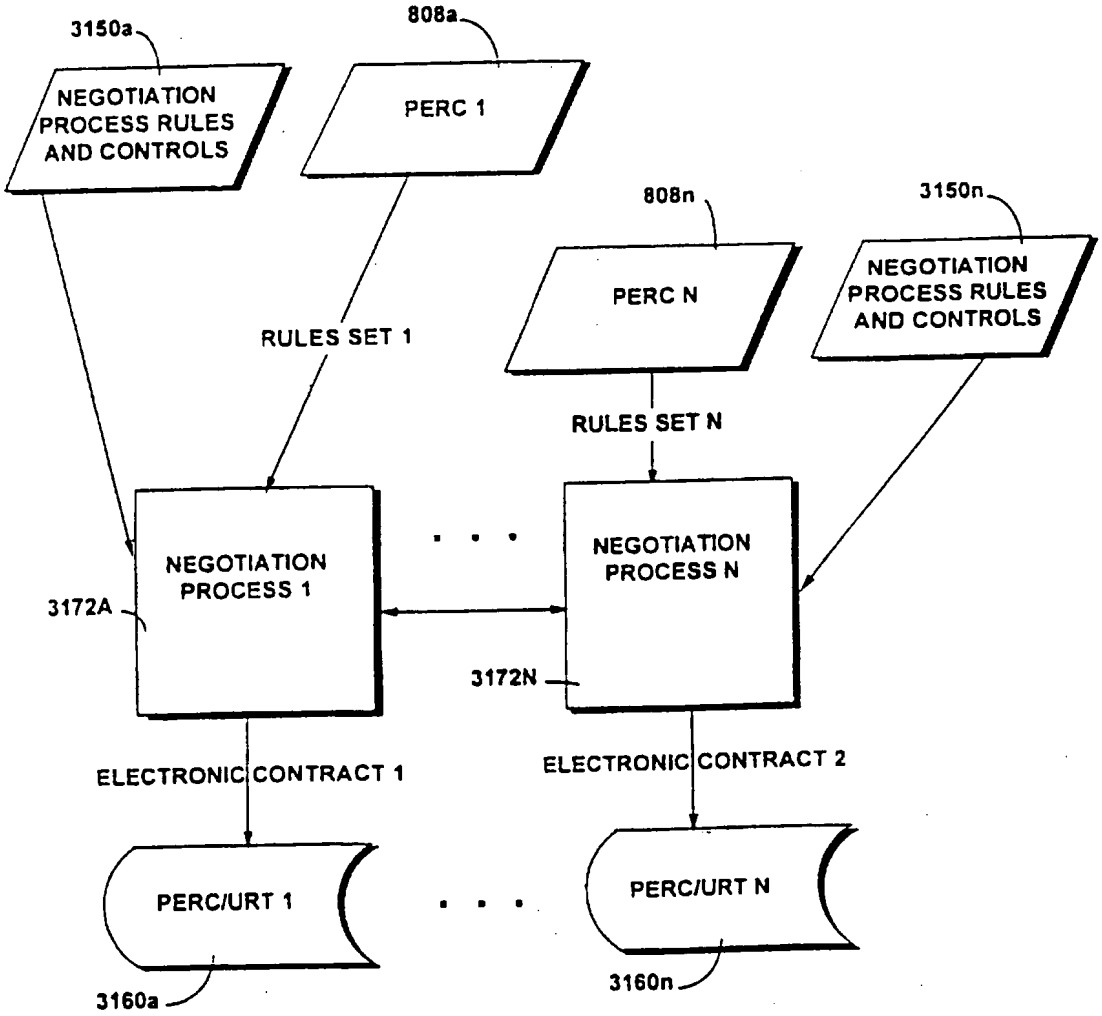
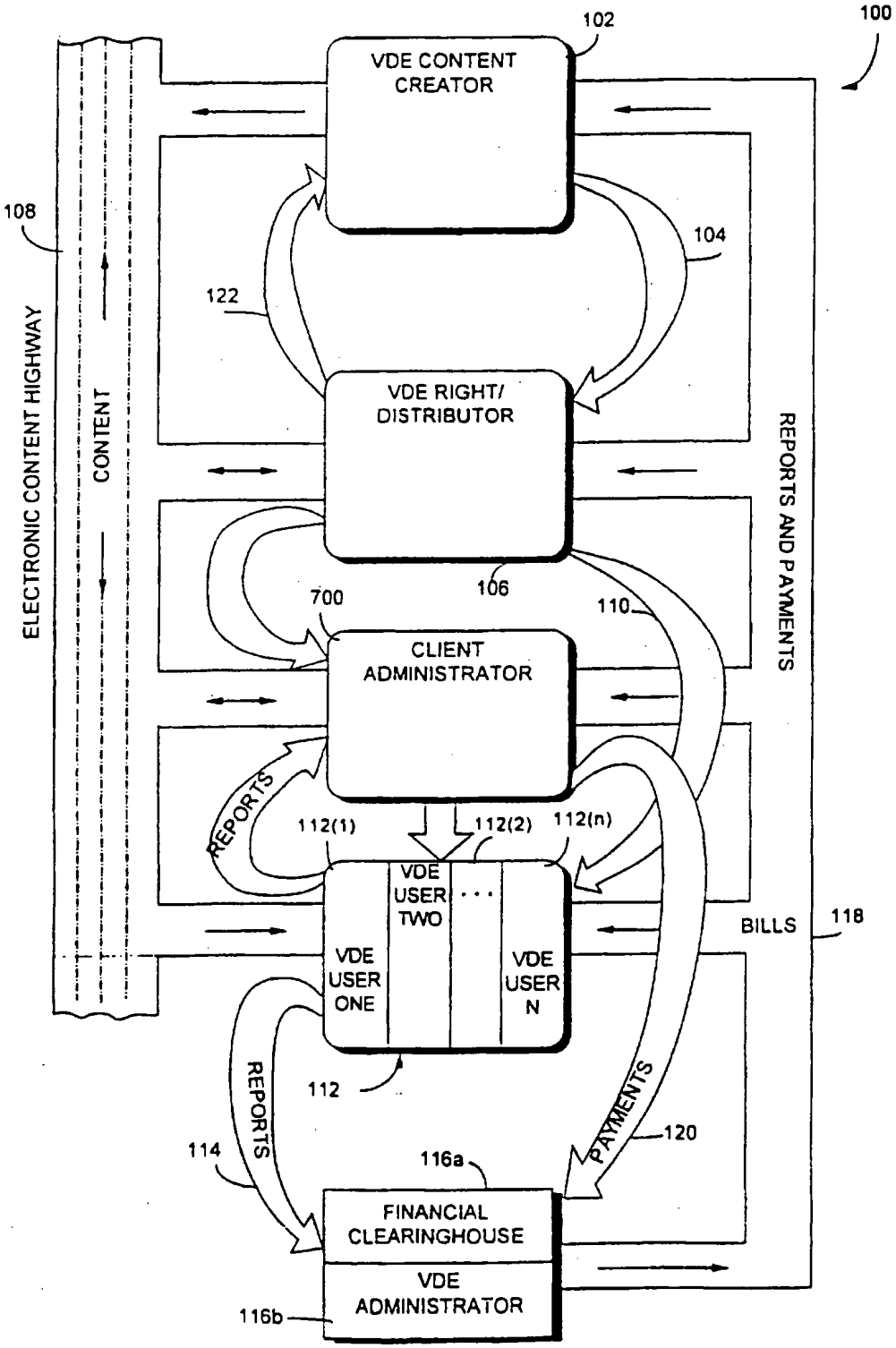


FIG. 77



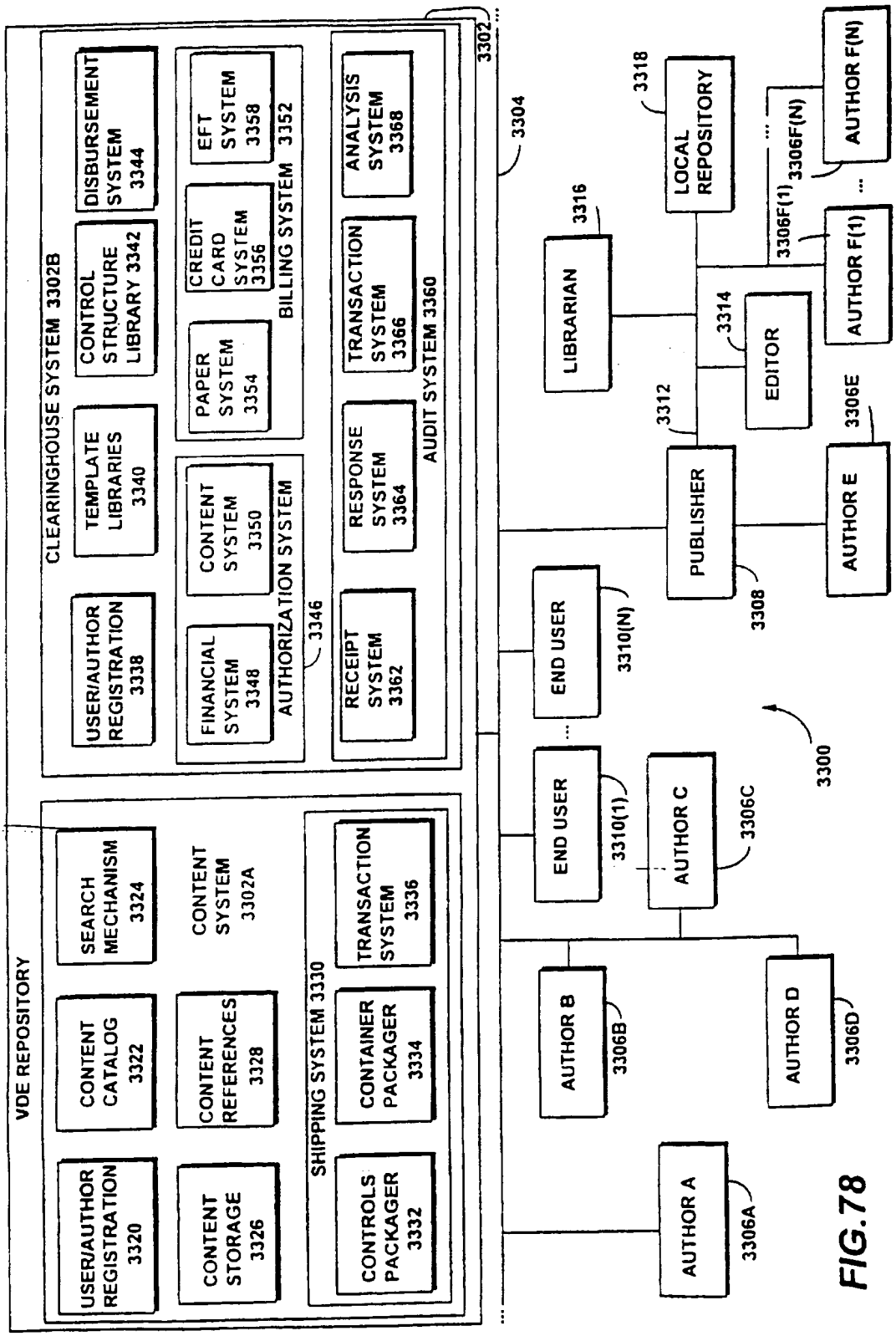


FIG.78

FIG. 79

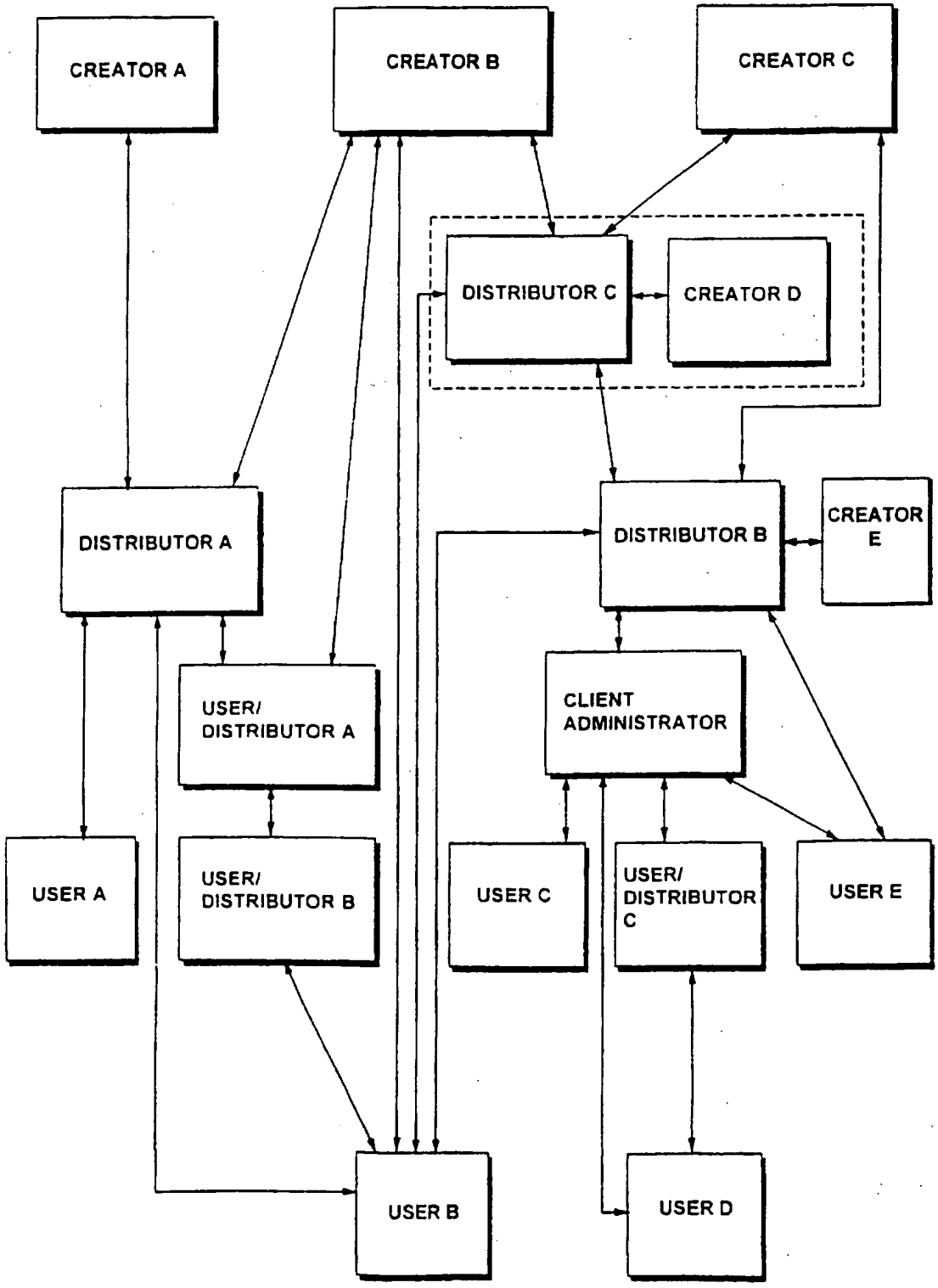
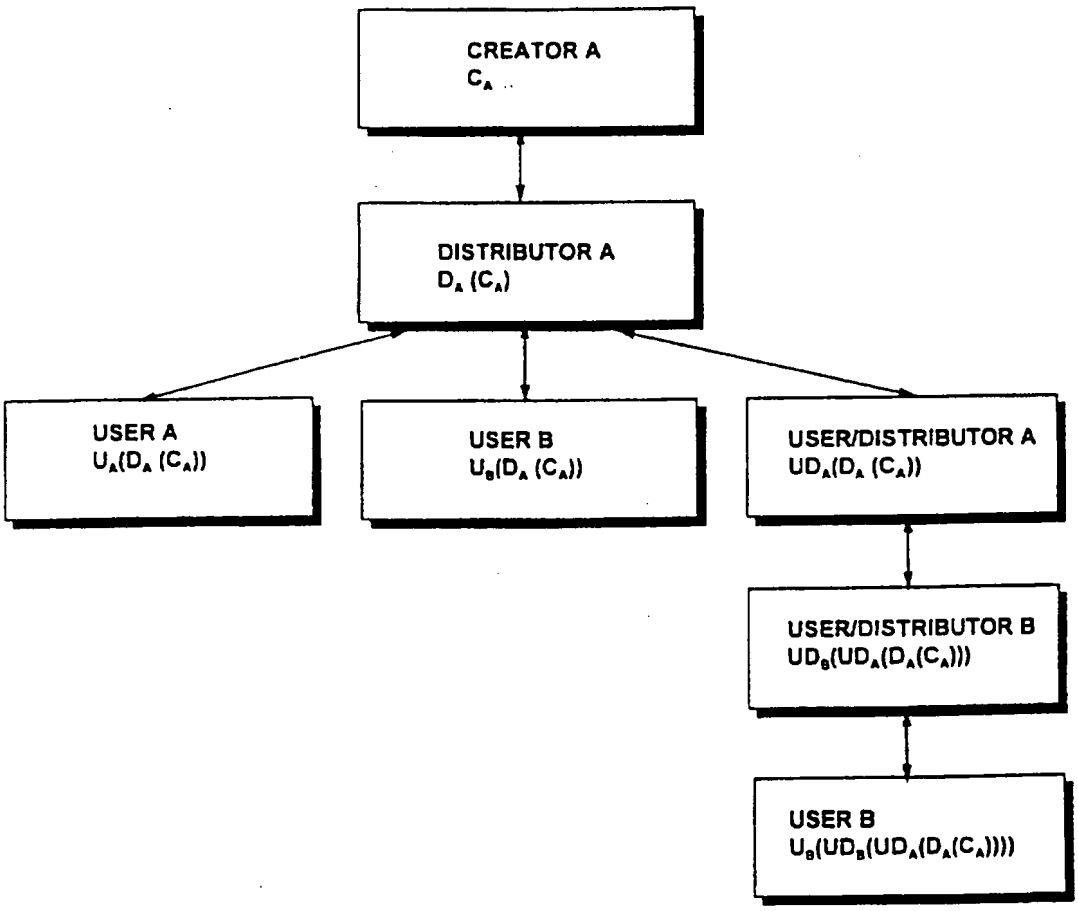


FIG. 80



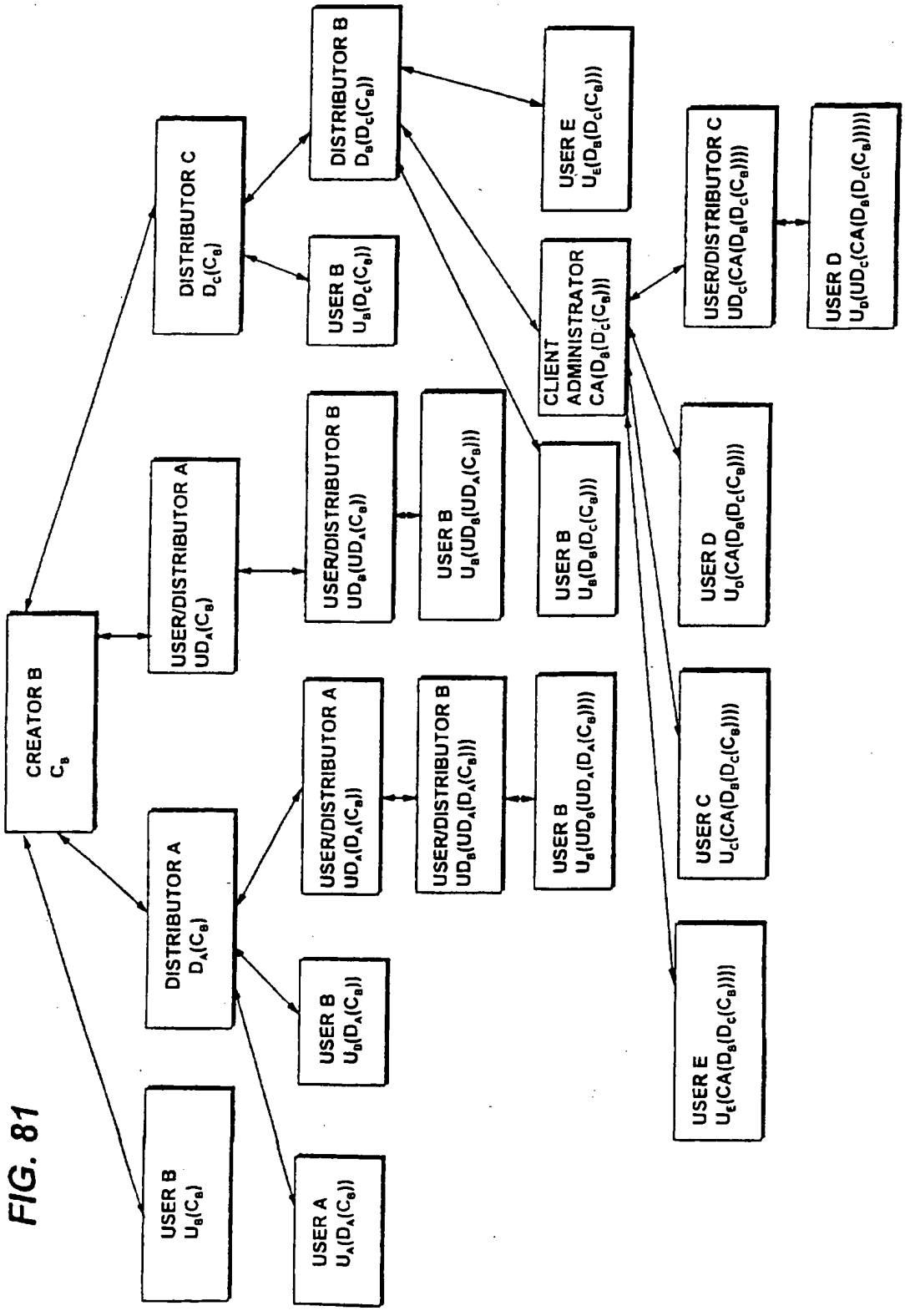


FIG. 81

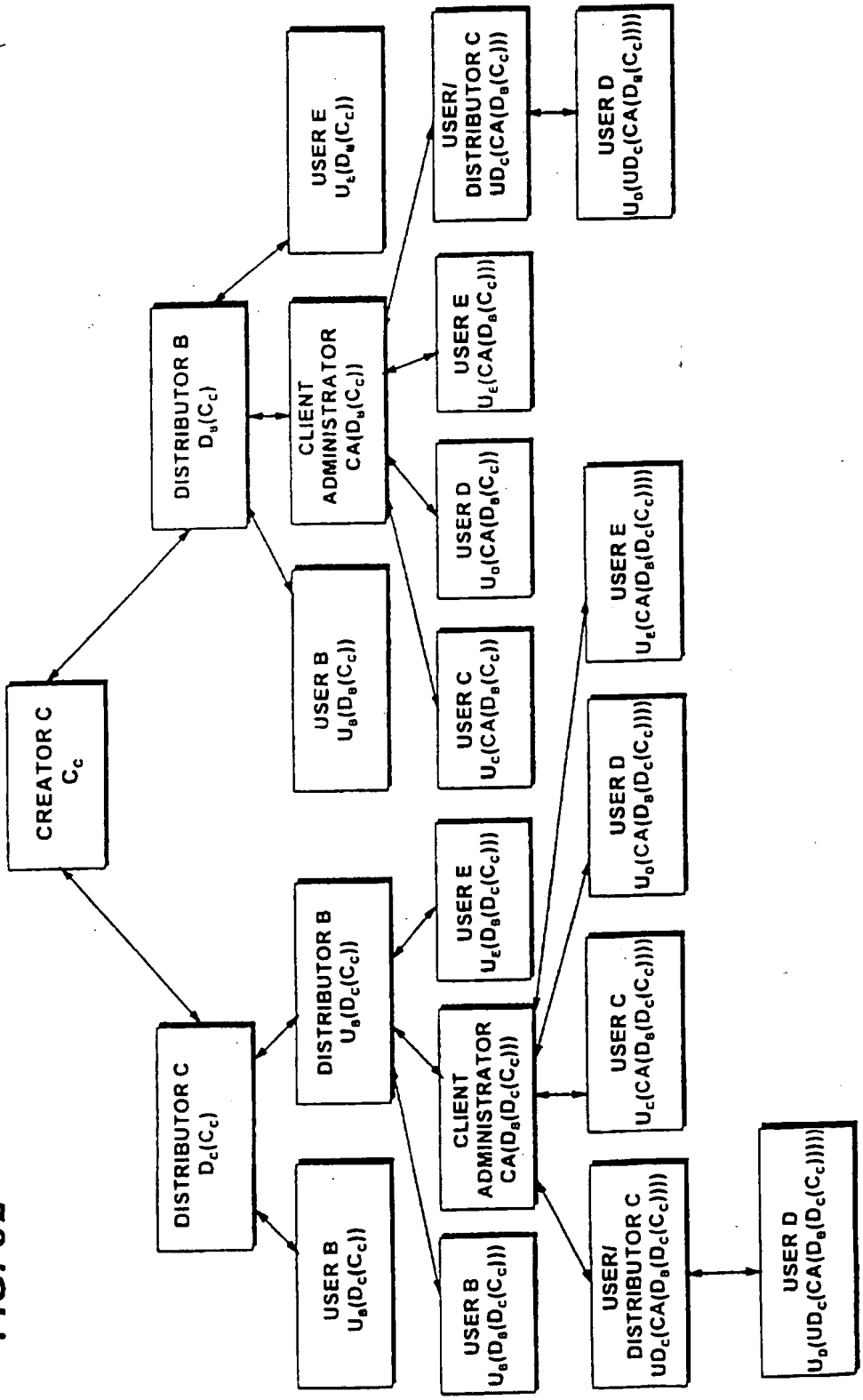


FIG. 82

FIG. 83

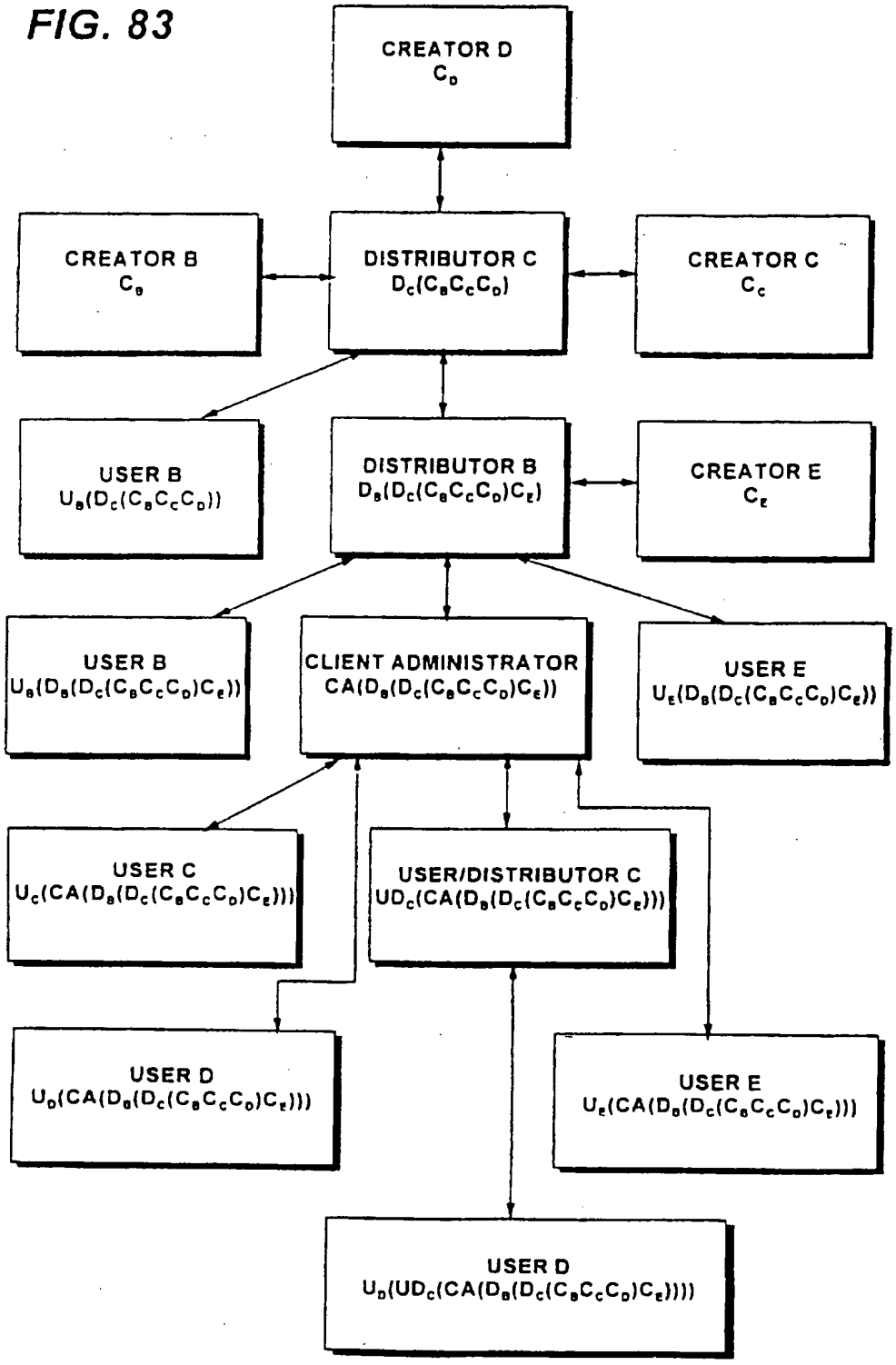


FIG. 84

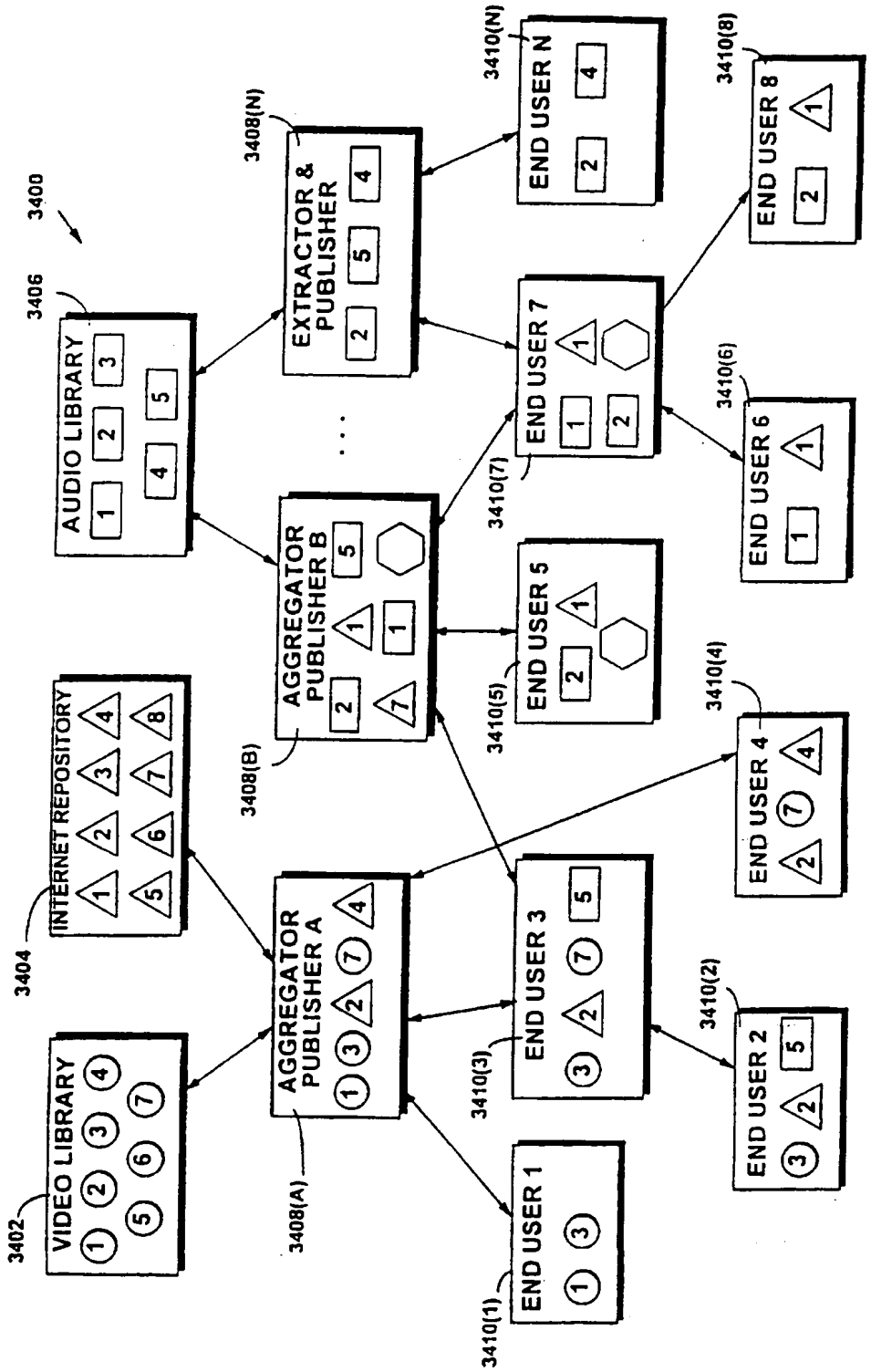
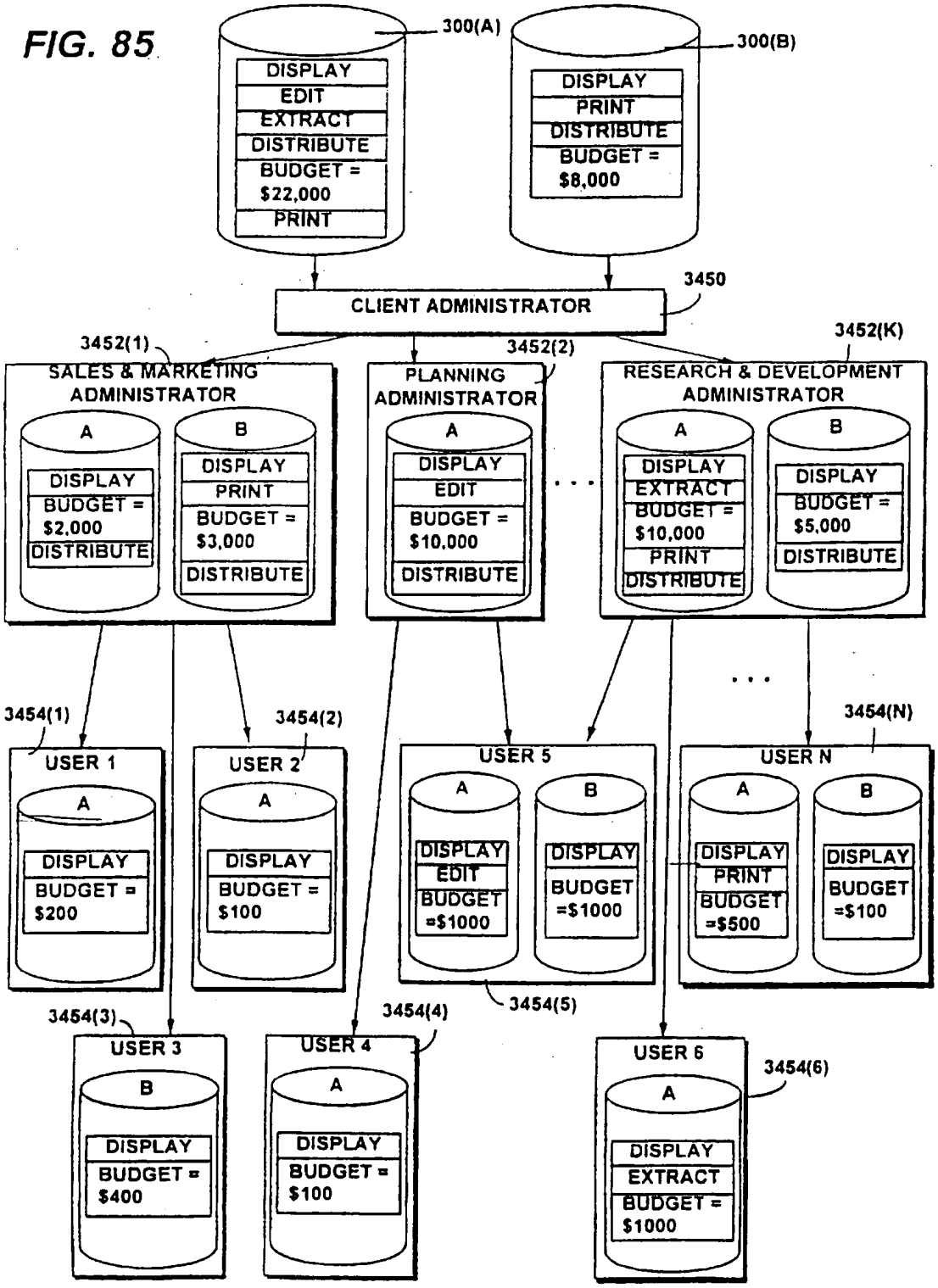


FIG. 85



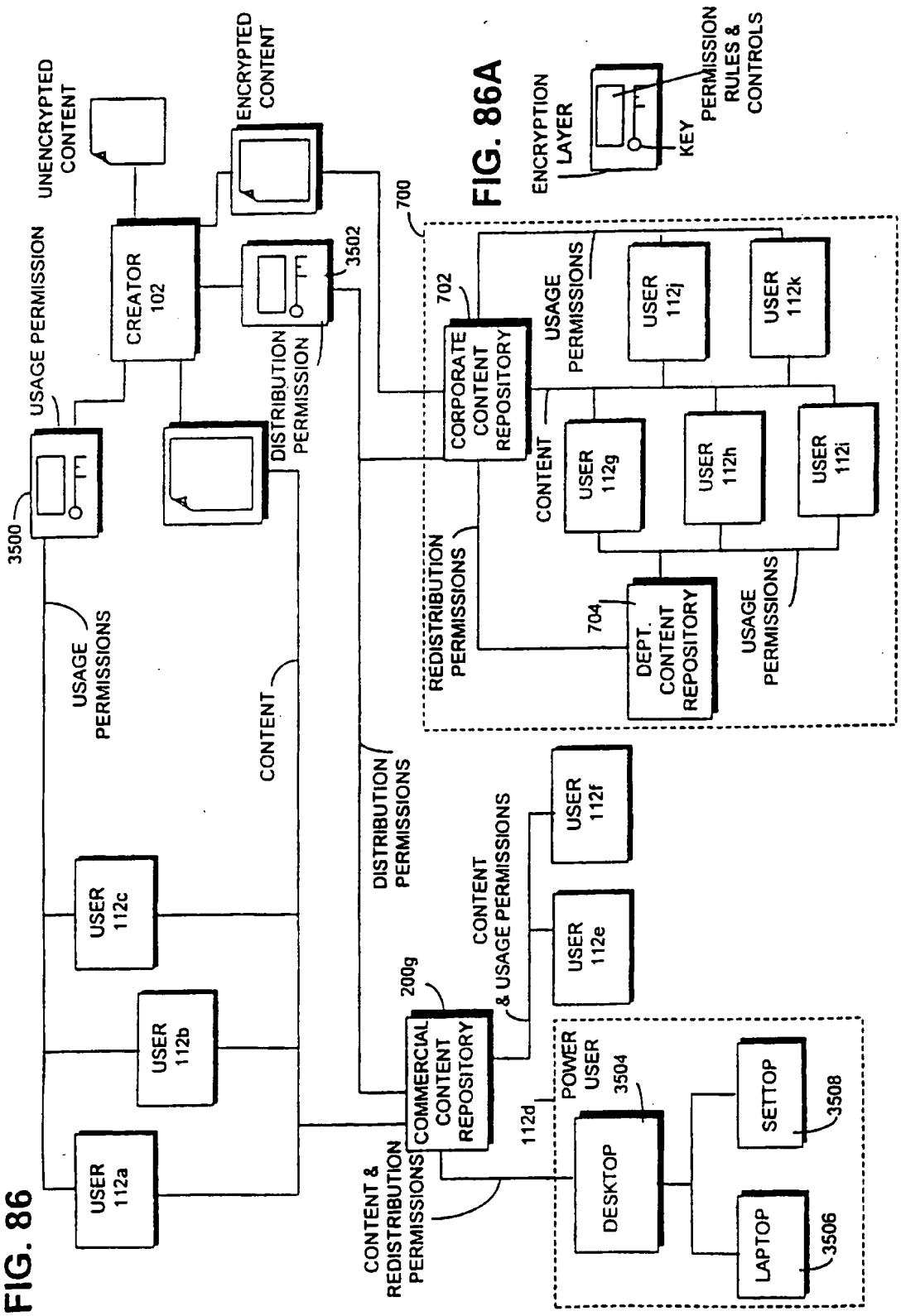


FIG. 86

FIG. 86A

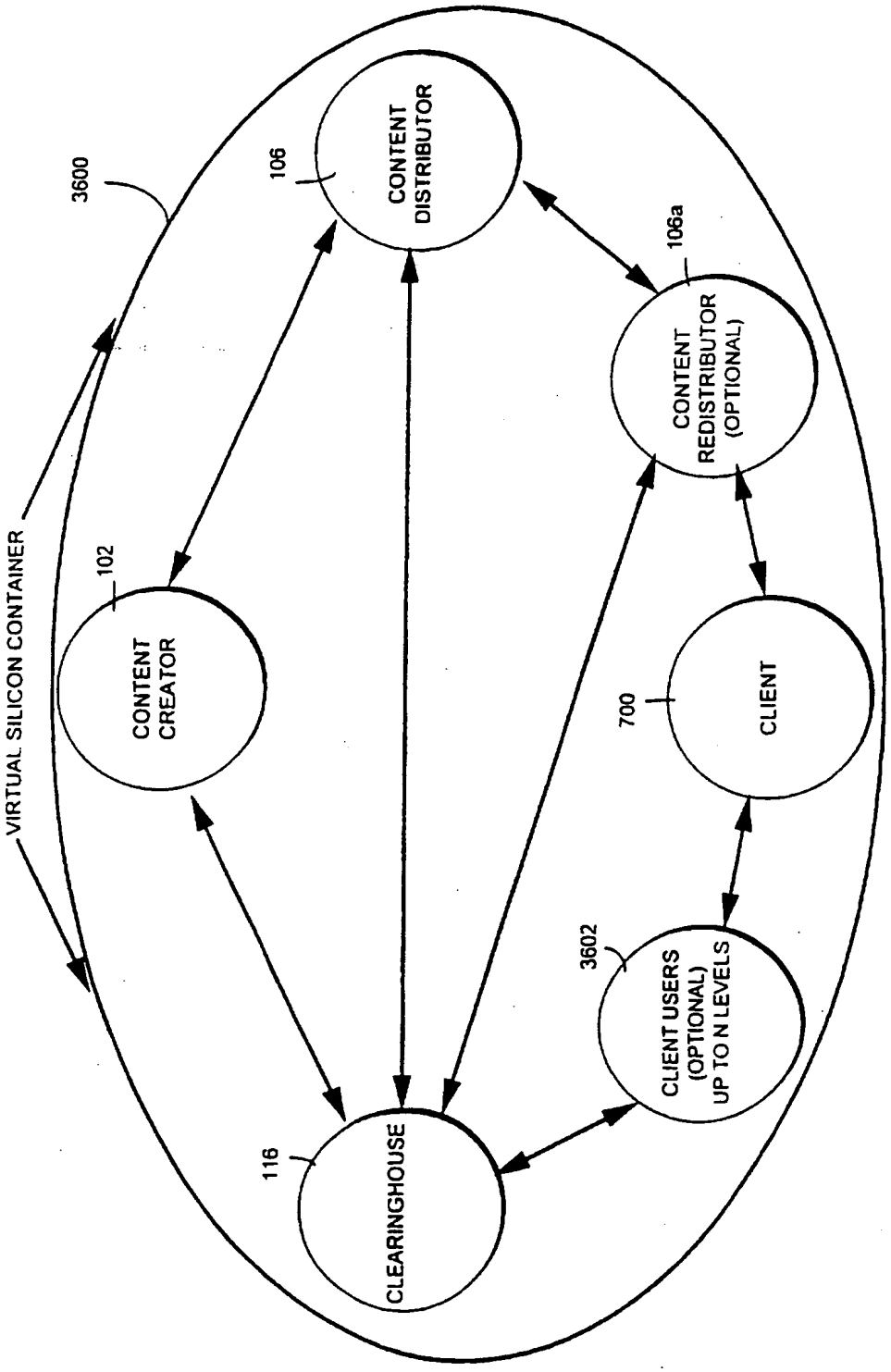


FIG. 87

INTERNATIONAL SEARCH REPORT

Im dional Application No
PCT/US 97/15243

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F1/00

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CHOUDHURY A K ET AL: "COPYRIGHT PROTECTION FOR ELECTRONIC PUBLISHING OVER COMPUTER NETWORKS" IEEE NETWORK: THE MAGAZINE OF COMPUTER COMMUNICATIONS, vol. 9, no. 3 May 1995, pages 12-20, XP000505280 see the whole document ---	18
Y	WO 90 02382 A (INDATA CORP) 8 March 1990 see abstract; figures 2,12,13,15 see page 18, paragraph 3 - page 21, paragraph 2 see page 23, last paragraph - page 24, paragraph 1	1-10
A	--- -/--	11-17

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

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
17 December 1997	29/12/1997

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx 31 651 epo nl, Fax: (+31-70) 340-3016	Powell, D

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int ional Application Nc
PCT/US 97/15243

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropnate, of the relevant passages	Relevant to claim No.
Y	EP 0 715 246 A (XEROX CORP) 5 June 1996 see the whole document	1-10
A		11-17
A	US 5 224 163 A (GASSER MORRIE ET AL) 29 June 1993 see the whole document	11-16
A	WO 94 01821 A (SECURE COMPUTING CORP) 20 January 1994 see the whole document	17
A	WO 94 03859 A (INT STANDARD ELECTRIC CORP) 17 February 1994 see the whole document	17

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. Jonal Application No
PCT/US 97/15243

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9002382 A	08-03-90	AU 4188289 A	23-03-90
		EP 0472521 A	04-03-92
		US 5247575 A	21-09-93
EP 0715246 A	05-06-96	US 5638443 A	10-06-97
		JP 8263439 A	11-10-96
US 5224163 A	29-06-93	NONE	
WO 9401821 A	20-01-94	US 5596718 A	21-01-97
		AU 663406 B	05-10-95
		AU 4672693 A	31-01-94
		EP 0649546 A	26-04-95
		JP 7509086 T	05-10-95
WO 9403859 A	17-02-94	EP 0606401 A	20-07-94
		JP 7502847 T	23-03-95

IFW

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Group Art Unit	3621
Examiner Name	Evens J. Augustin
Total Number of Pages in This Submission	
Attorney Docket Number	111325-235000

ENCLOSURES *(check all that apply)*

- ☒ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment / Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☒ Information Disclosure Statement
- ☐ Certified Copy of Priority Document(s)
- ☐ Response to Missing Parts/ Incomplete Application
  - ☐ Response to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☐ Request for Refund
- ☐ CD, Number of CD(s) _____
- ☐ After Allowance Communication to Group
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to Group *(Appeal Notice, Brief, Reply Brief)*
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) *(please identify below)*:
  1. PTO Form 1449
  2. One Box including 112 cited references

Remarks

☒ The Director 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 9th Street, N.W., Suite 900 Washington, D.C. 20004-2128
Signature	/Marc S. Kaufman, Reg. # 35,212/
Date	February 20, 2008

CERTIFICATE OF MAILING OR TRANSMISSION

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450, or facsimile transmitted to the U.S. Patent and Trademark Office (Fax No. (571) 273-8300) on the date shown below.

Name *(Print/Type)*			
Signature		Date	

FEE TRANSMITTAL FOR FY 2008

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT $180.00

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai NGUYEN, et al.
Examiner Name	Evens J. Augustin
Art Unit	3621
Attorney Docket No.	111325-235000

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____

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

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, **except for the filing fee**

☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-20238.

FEE CALCULATION

1. BASIC FILING, SEARCH AND EXAMINATION FEES

	FILING FEES		SEARCH FEES		EXAMINATION FEES		
Application Type	**Fee ($)**	**Small Entity Fee ($)**	**Fee ($)**	**Small Entity Fee ($)**	**Fee ($)**	**Small Entity Fee ($)**	**Fees Paid ($)**
Utility	310	155	510	255	210	105	
Design	210	105	100	50	130	65	
Plant	210	105	310	155	160	80	
Reissue	310	155	510	255	620	310	
Provisional	210	105	N/A	N/A	N/A	N/A	

2. EXCESS CLAIM FEES

Fee Description	**Fee ($)**	**Small Entity Fee ($)**
Each claim over 20 (including Reissues)	50	25
Each independent claim over 3 (including Reissues)	210	105
Multiple dependent claims	370	185

Total Claims		**Extra Claims**	**Fee ($)**	**Fee Paid ($)**	**Multiple Dependent Claims**	
	- 20 or HP =		x	=	**Fee ($)**	**Fee Paid ($)**

HP = highest number of total claims paid for, if greater than 20

Indep. Claims		**Extra Claims**	**Fee ($)**	**Fee Paid ($)**
	- 3 or HP =		x	=

HP = highest number of independent claims paid for, if greater than 3

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper, the application size fee due is $260 ($130 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets		**Extra Sheets**		**Number of each additional 50 or fraction thereof**		**Fee ($)**		**Fee Paid ($)**
	- 100 =		/ 50 =	(round up to a whole number)	x		=	

4. OTHER FEE(S)

	Fees Paid ($)
Non-English Specification, $130 fee (no small entity discount)	
Other: **Information Disclosure Filing Fee**	**$180.00**

SUBMITTED BY

Signature	/Marc S. Kaufman, Reg. # 35,212/	Registration No. (Attorney/Agent) 35,212	Telephone (202) 585-8000
Name (Print/Type)	Marc S. Kaufman		Date February 20, 2008

CERTIFICATE OF MAILING OR TRANSMISSION [35 CFR 1.8(a)]

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on _____.

Signature: _____

Name: _____

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

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:	)	Confirmation No.: 8299
Mai **NGUYEN** et al.	)	Group Art Unit: 3621
Application No.: 10/956,070	)	Examiner: Augustin, Evens J.
Filed: October 4, 2004	)	
For: **SYSTEM AND METHOD FOR RIGHTS OFFERING AND GRANTING USING SHARED STATE VARIABLES**	)	Date: February 20, 2008

INFORMATION DISCLOSURE STATEMENT

United States Patent and Trademark Office
Customer Window
Randolph Building
401 Dulany Street
Alexandria, VA 22314

Dear Sir:

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. Pursuant to 37 C.F.R. § 1.98(a)(2)(ii), copies of the cited U.S. patents (*i.e.*, Reference Cite Nos. 1–101) are not enclosed. Copies of the cited Foreign patents (*i.e.*, Reference Cite Nos. 102–173) are enclosed. Copies of the cited non-patent references (*i.e.*, Reference Cite Nos. 174–213) are enclosed. The references have been cited in recent oppositions in the European Patent Office relating to cases owned by assignee.

The Commissioner is hereby authorized to charge the **Deposit Account No. 19-2380** in the amount of **$180.00** representing filing fees.

It is requested that the accompanying PTO/SB/08A be considered and made of record in the above-identified application. To assist the Examiner, the documents are listed on the attached form PTO/SB/08A. 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, or credit any overpayment to Deposit Account No. 19-2380.

02/21/2008 HDELETE1 00000030 192380 10956070
01 FC:1806 180.00 DA

Respectfully submitted,
NIXON PEABODY LLP

Date: February 20, 2008

By: /Marc S. Kaufman, Reg. # 35,212/
Marc S. Kaufman
Registration No. 35,212

NIXON PEABODY LLP
CUSTOMER NO.: 22204
401 9th Street, N.W., Suite 900
Washington, DC 20004
Tel: 202-585-8000
Fax: 202-585-8080

PTO/SB/08A (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 1 of 9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

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
	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.

PTO/SB/08A (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 2 of 9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

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.

PTO/SB/08A (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 3 of 9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

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
	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.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet	4	of	9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] Number[4]	Foreign Patent Document: Kind Code[5] (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	T[6]
	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 Electroincs 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.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet	5	of	9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] Number[4]	Foreign Patent Document: Kind Code[5] (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	T[6]
	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.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(use as many sheets as necessary)

Sheet 6 of 9

Complete if Known	
Application Number	10/956,070
Filing Date	October 4, 2004
First Named Inventor	Mai Nguyen et al.
Art Unit	3621
Examiner Name	Augustin, Evens J.
Attorney Docket Number	111325/235000

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No.[1]	Foreign Patent Document: Country Code[3] Number[4] Kind Code[5] (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	T[6]
	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.

Substitute for form 1449A/PTO				*Complete if Known*	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(use as many sheets as necessary)*				**Application Number**	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai Nguyen et al.
				Art Unit	3621
				Examiner Name	Augustin, Evens J.
Sheet	7	of	9	Attorney Docket Number	111325/235000

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]
	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.

PTO/SB/08A (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				*Complete if Known*	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(use as many sheets as necessary)*				**Application Number**	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai Nguyen et al.
				Art Unit	3621
				Examiner Name	Augustin, Evens J.
Sheet	8	of	9	Attorney Docket Number	111325/235000

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/correct/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, 2nd 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/encyrption/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 Encyption," 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.

PTO/SB/08A (10-01)
Approved for use through 10/31/2002. OMB 0651-0031
U.S. Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449A/PTO				*Complete if Known*	
INFORMATION DISCLOSURE STATEMENT BY APPLICANT *(use as many sheets as necessary)*				**Application Number**	10/956,070
				Filing Date	October 4, 2004
				First Named Inventor	Mai Nguyen et al.
				Art Unit	3621
				Examiner Name	Augustin, Evens J.
Sheet	9	of	9	Attorney Docket Number	111325/235000

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.

16) Family number: 12389386 (JP11031130 A2) | full-text | status | citations | < | > | ^ |

Title: SERVICE PROVIDING DEVICE

Priority: JP19970184866 19970710
Priority Map

Family:	Publication number	Publication date	Application number	Application date	Link
Family Explorer	JP11031130 A2	19990202	JP19970184866	19970710	

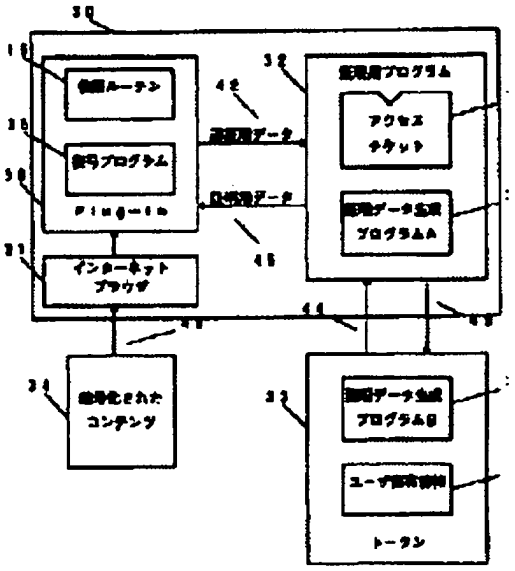
Assignee(s): FUJI XEROX CO LTD

Inventor(s): KOJIMA SHUNICHI ; KONO KENJI ; NAKAGAKI JUHEI

International class (IPC 8): G06F15/00 G09C1/00 H04L9/32 (Advanced/Invention); G06F15/00 G09C1/00 H04L9/32 (Core/Invention)

International class (IPC 1-7): G06F15/00 G09C1/00 H04L9/32

Abstract:
Source: JP11031130A2 PROBLEM TO BE SOLVED: To provide the utilization of service only to a user who has a legal right, minimizing the burden on the user and a service provider. SOLUTION: When a plug-in 38 of an Internet browser 31 is started, a verification program 15 in the plug-in 38 is started, communicates with a program 32 for certification and performs user authentication. A certification data generation program A36 of the program 32 cooperates with a certification data generation program B37 in a token 33, calculates based on a user inherent information 16 and an access ticket 13 and communicates with the program 15 in the plug-in 38 based on the calculation. As the result of the communication, the success of authentication by the program 15 is limited to only when the three of the user inherent information, the access ticket and enciphered contents correctly correspond with one another.



17) Family number: 12393236 (JP11032037 A2) | full-text | status | citations | < | > | ^ |

Title: CERTIFICATION DATA GENERATING DEVICE

Priority: JP19970188801 19970714
Priority Map

Family: Family Explorer	Publication number	Publication date	Application number	Application date	Link
	JP11032037 A2	19990202	JP19970188801	19970714	
	JP3641909 B2	20050427	JP19970188801	19970714	

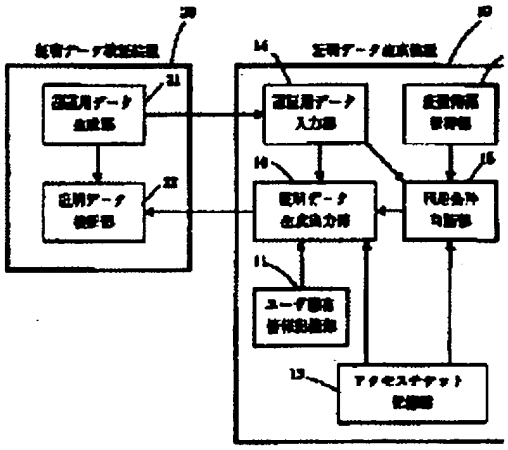
Assignee(s): FUJI XEROX CO LTD

Inventor(s): NAKAGAKI JUHEI ; SHIN YOSHIHIRO

International class (IPC 8): G06F15/00 G06F9/06 G09C1/00 H04L9/32 (Advanced/Invention); G06F15/00 G06F9/06 G09C1/00 H04L9/32 (Core/Invention)

International class (IPC 1-7): G06F15/00 G06F9/06 G09C1/00 H04L9/32

Abstract:
Source: JP11032037A2 PROBLEM TO BE SOLVED: To pre-pay access qualification to purchase or rent without imposing any surplus load on a certification data generating device side. SOLUTION: A pre-paid purchase ticket T_2 is stored in an access ticket storing part 13. Next, (T_1', n_2) is inputted to a certification data-inputting part 14. A use condition judging part 15 extracts a corresponding access ticket (t_2, L_2, n_2), checks whether or not a use condition L_2 is fulfilled, and reduces frequency information V, when the use condition is fulfilled. A certification data generating and outputting part 16 calculates certification data R by using auxiliary certification decision $(t)_2$ and the use condition L_2 extracted by the use condition decision part 15 and (du) read from a user specific information storing part 11, and outputs T_1. A user performs access to a program in a purchase state or a rent state by using the T_1.



12) Family number: 13081077 (JP11205306 A2) | ✉ | 🖶 | full-text | status | citations | < | > | ^ | ☐ | 🛒

Title: AUTHENTICATION SYSTEM AND AUTHENTICATION METHOD

Priority: JP19980006267 19980116
Priority Map

Family:	Publication number	Publication date	Application number	Application date	Link
Family Explorer	JP11205306 A2	19990730	JP19980006267	19980116	

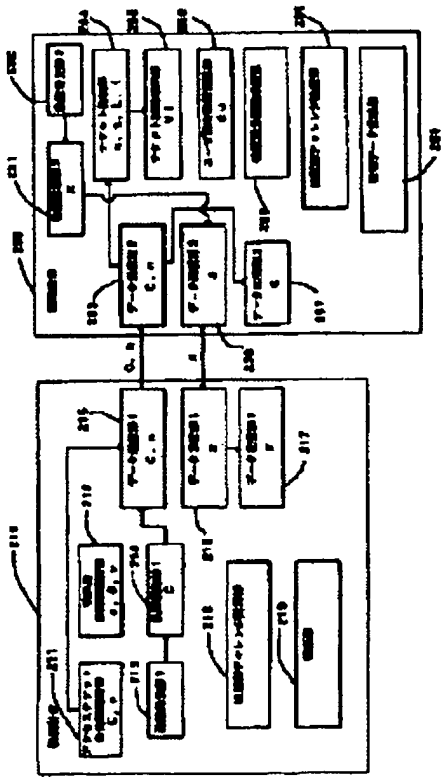
Assignee(s): FUJI XEROX CO LTD

Inventor(s): KOJIMA SHUNICHI ; KONO KENJI ; TAGUCHI MASAHIRO ; TERAO TARO

International class (IPC 8): G09C1/00 H04L9/32 (Advanced/Invention); G09C1/00 H04L9/32 (Core/Invention)

International class (IPC 1-7): G09C1/00 H04L9/32

Abstract:
Source: JP11205306A2 PROBLEM TO BE SOLVED: To provide a system and method that realize diversified services by using an access ticket generated from characteristics information not belonging to a person and information specific to the user, as for the authentification system that authenticates legality of the user. SOLUTION: The authentication device 210 sends authentication data and a ticket identifier to an authentication device 250, the authentification device 250 sends the authentication data to the authentication device 210, which calculates an authentication challenge (ρ) based on a ticket attribute revision request (μ) and an authentication device authentification data (χ). The authentification device 250 receives the ρ and the (&epsi , v) to authenticate an authentication device open key based on input data and an authentication device open key identifier (v '), to authenticate the authentification device challenge and to revise contents of a ticket attribute record (VI) depending on the ticket attribute revision request (μ). Furthermore, an authentification data generating section calculates a response (R) and the authentication device authenticates the legality of the response (R).



11) Family number: 13107360 (JP11215121 A2) | full-text | status | citations | | > |

Title: DEVICE AND METHOD FOR AUTHENTICATION

Priority: JP19980016710 19980129
Priority Map

Family:	Publication number	Publication date	Application number	Application date	Link
Family Explorer	JP11215121 A2	19990806	JP19980016710	19980129	
	JP3791169 B2	20060628	JP19980016710	19980129	

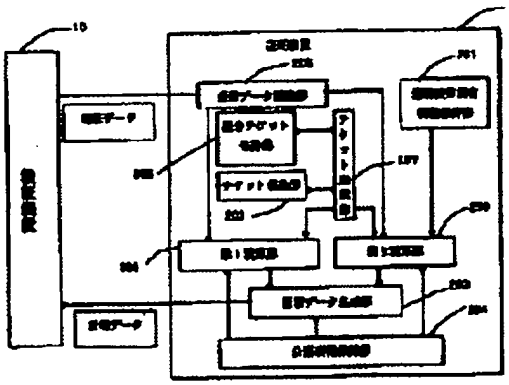
Assignee(s): FUJI XEROX CO LTD

Inventor(s): KIKO KENICHIROU

International class (IPC 8): G09C1/00 H04L9/32 (Advanced/Invention); G09C1/00 H04L9/32 (Core/Invention)

International class (IPC 1-7): G09C1/00 H04L9/32

Abstract:
Source: JP11215121A2 PROBLEM TO BE SOLVED: To perform composite authentication by using the combination of different kinds of issued tickets. SOLUTION: The ticket holding section 202 of a certifying device 20 holds a ticket indicating the specific right of a user while a composite ticket holding section 206 holds a composite ticket for certifying that the user holds a plurality of other effective tickets. A certifying data generating section 203 certifies the presence of a compositely designated right by generating certifying data through executing a prescribed operation by the use of a prescribed access ticket, a composite ticket, and inherent information of the certifying device to authentication information sent from a verifying device 10.



Search result: 1 of 1

(WO/2004/103843) PACKAGING METHOD AND DEVICE, PACKAGING BAGS

Biblio. Data | Description | Claims | National Phase | Notices | Documents

Latest bibliographic data on file with the International Bureau

Publication Number: WO/2004/103843 **International Application No.:** PCT/FR2004/001185
Publication Date: 02.12.2004 **International Filing Date:** 14.05.2004

Int. Class.: *B65D 33/25* (2006.01), *B65D 85/16* (2006.01)

Applicants: **S2F FLEXICO** [FR/FR]; 1, route de Méru, F-60119 Henonville (FR) *(All Except US)*.
BOIS, Henri, Georges [FR/FR]; 61, boulevard d'Inkermann, F-92200 Neuilly sur Seine (FR) *(US Only)*.

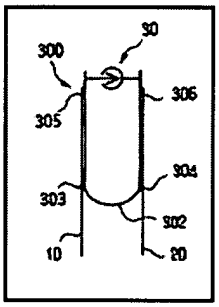
Inventor: **BOIS, Henri, Georges** [FR/FR]; 61, boulevard d'Inkermann, F-92200 Neuilly sur Seine (FR).

Agent: **MARTIN, Jean-Jacques**; Cabinet Regimbeau, 20, rue de Chazelles, F-75847 Paris Cedex 17 (FR).

Priority Data: 03/05887 16.05.2003 FR

Title: PACKAGING METHOD AND DEVICE, PACKAGING BAGS

Abstract: The invention relates to a packaging method comprising the following steps: provision of a bag whose mouth comprises opening/closing means (30) for multiple successive openings and closings and a cleavable linking veil, located at a distance therefrom inside the bag in relation to said opening/closing means (30); introduction of contents (100) to be wrapped in the bag and tightening of said bag in order to close it, tension being applied to the contents (10); the veil (40) enters into contact with the contents (100) avoiding the application of stress on the opening/closing means, guaranteeing free access to the contents (100) via said opening/closing means (30) after tearing, enabling the bag to be relaxed in a closed state as a result of the distance (D) separating the veil (40) and the opening/closing means (30). The invention also relates to a packaging device and to bags thus obtained.



Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
African Regional Intellectual Property Org. (ARIPO) (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW)
Eurasian Patent Organization (EAPO) (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM)
European Patent Office (EPO) (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR)
African Intellectual Property Organization (OAPI) (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Publication Language: French (FR)

Filing Language: French (FR)

4) Family number: 33529421 (JP2005253109 A2)
extended family

| | full-text | status | citations | < | > | ^ |

Title: CONDITIONAL ACCESS SYSTEM

Priority: US19970054575P 19970801 US19980126921 19980731
Priority Map

Family: Family Explorer

Publication number	Publication date	Application number	Application date	Link
JP2005253109 A2	20050915	JP20050120425	20050418	
WO9909743 A2	19990225	WO1998US16079	19980731	
WO9909743 A3	19990527	WO1998US16079	19980731	

Assignee(s) (std): SCIENTIFIC ATLANTA

Assignee(s): SCIENTIFIC ATLANTA INC

Inventor(s) (std): AKINS GLENDON L III ; PALGON MICHAEL S ; PINDER HOWARD G ; WASILEWSKI ANTHONY J

Inventor(s): AKINS GLENDON L

Designated states: AL AM AT AU AZ BA BB BE BF BG BJ BR BY CA CF CG CH CI CM CN CU CY CZ DE DK EE ES FI F GA GB GE GH GM GN GR GW HR HU ID IE IL IS IT JP KE KG KP KR KZ LC LK LR LS LT LU LV M MD MG MK ML MN MR MW MX NE NL NO NZ PL PT RO RU SD SE SG SI SK SL SN SZ TD TG TJ TR TT UA UG UZ VN YU ZW

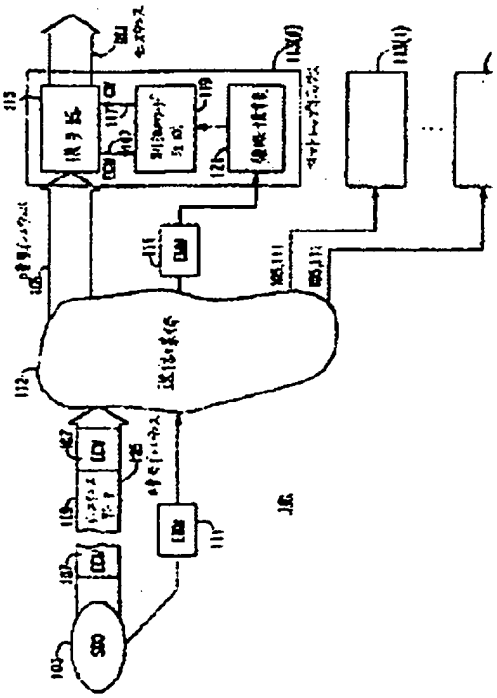
International class (IPC 8): H04H1/00 H04L9/08 H04N5/00 H04N7/16 H04N7/167 H04N7/173 (Advanced/Invention); H04H1/00 H04L9/08 H04N5/00 H04N7/16 H04N7/167 H04N7/173 (Core/Invention)

International class (IPC 1-7): H04L9/08 H04N7/167

European class: H04N5/00M4 H04N7/167D H04N7/16E2

Cited documents: WO9704553, US5381481, US5029207, US4887296, US4864615, US4736422, US4613901,

Abstract:
Source: JP2005253109A2 PROBLEM TO BE SOLVED: To provide a cable television system which provides conditional access to services. SOLUTION: This cable television system includes a headend from which service "instances" or programs are broadcast and a plurality of set top units for receiving the instances and selectively decrypting the instances for display to system subscribers. The service instances are encrypted using public keys and/or private keys provided by service providers or central authorization agents. Keys used by the set tops for a selective decryption may also be public or private in nature, and such keys may be reassigned at different times to provide a cable television system in which piracy concerns are minimized. COPYRIGHT: (C)2005,JPO&NCIPI<



1/9/1
DIALOG(R)File 347: JAPIO

08787202 **Image available**
CRYPTOGRAPHIC KEY SYSTEM

Pub. No.: 2006-180562 [**JP 2006180562** A]
Published: July 06, 2006 (20060706)
Inventor: SAITO MAKOTO
MOMIKI JUNICHI
Applicant: INTARSIA SOFTWARE LLC
Application No.: 2006-082675 [JP 200682675]
Division of 07-346095 [JP 95346095]
Filed: March 24, 2006 (20060324)
Priority: 06-309292 [JP 94309292], JP (Japan), December 13, 1994 (19941213)

International Patent Class (v8 + Attributes)
IPC + Level Value Position Status Version Action Source Office:

H04L-0009/08 A I F B 20060101 20060609 H JP

ABSTRACT

PROBLEM TO BE SOLVED: To provide a concrete structure for applying a cryptographic key system to a television system, a database system or an electronic commercial transaction system or the like.

SOLUTION: This system consists of a broadcasting station, a database, a receiving apparatus, a data communications apparatus and a user terminal. As a cryptographic key system, a secret-key system, a public-key system, and a digital signature system are used. The keys used in the system are either encrypted, or remain uncrypted to be supplied by broadcasting. The system is effective in preventing the unauthorized use of the database system, managing copyrights, and in pay-per-view systems and video-on-demand systems. Further, the system is effective in realizing an electronic market which uses an electronic data information system.

55) Family number: 10272458 (JP5168039 A2) | full-text | status | citations | < | > | ^ |

Title: RECORDING ENCODE METHOD FOR HIGH FIDELITY TELEVISION SIGNAL

Priority: JP19910352059 19911213
Priority Map

Family: Family Explorer

Publication number	Publication date	Application number	Application date	Link
JP3185806 B2	20010711	JP19910352059	19911213	
JP5168039 A2	19930702	JP19910352059	19911213	

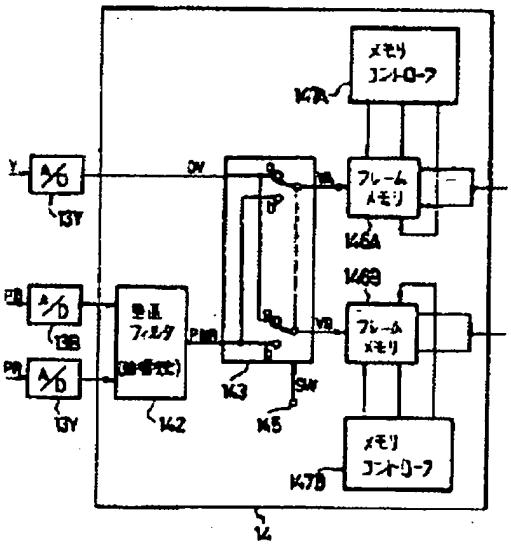
Assignee(s): SONY CORP

Inventor(s): ISHIMARU HIROYOSHI

International class (IPC 8): H04N11/22 H04N5/907 H04N9/80 H04N9/81 (Advanced/Invention); H04N11/06 H04N5/907 H04N9/80 H04N9/81 (Core/Invention)

International class (IPC 1-7): H04N11/22 H04N5/907 H04N9/80 H04N9/81

Abstract:
Source: JP5168039A2 PURPOSE: To encode a unit signal (TDM signal) for recording from a high fidelity television signal by controlling reading of plural output ports while using a serial access memory equipped with the plural output ports. CONSTITUTION: Memories 146A and 146B are serial access and two output ports are respectively provided in each memory. Then, write of input data VA and VB is controlled by memory controllers 147A and 147B, and reading of data from the respective output ports is independently controlled. Namely, TDM signals are written in memories 146A and 146B in the order of a luminance signal and a chrominance signal. In the case of reading, the same data are read from two output ports while deviating read timing, color difference signal data are extracted from the preceding output port, luminance signal data are extracted from the other output port, both data are synthesized and therefore, the required TDM signals are obtained.



PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G07F 7/08, 19/00, G06F 17/60 // 157:00

A1

(11) International Publication Number: **WO 96/13814**

(43) International Publication Date: 9 May 1996 (09.05.96)

(21) International Application Number: PCT/FI95/00591

(22) International Filing Date: 25 October 1995 (25.10.95)

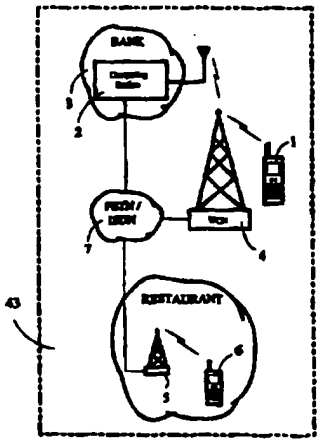
(30) Priority Data:
945075 28 October 1994 (28.10.94) FI

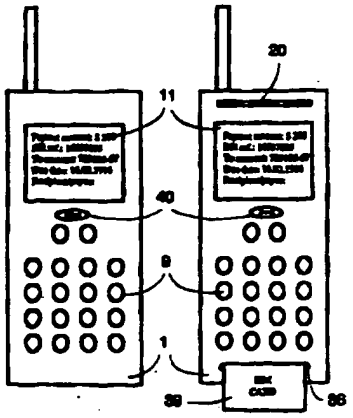
(71)(72) Applicant and Inventor: VAZVAN, Behruz [FI/FI]; Jämeräntaival 11 B 53, FIN-02150 Espoo (FI).

(81) Designated States: FI, JP, US, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
With amended claims and statement.

(54) Title: REAL TIME TELE-PAYMENT SYSTEM





(57) Abstract

This invention is a real time mobile tele-payment system that relates to payments of bills of mobile users, or providing the mobile users with the information about their bank account, the statement of account, or the movement on the account in a real time basis, by using their portable telephones under any wireless telecommunications systems. Certain features of this invention are intended as an expansion of value-added services of currently existing mobile communications systems. This invention also provides the retail and trading businessmen with the possibility to charge their customers, via wireless communications networks and in a real time basis, by using their mobile telephones. In this invention, in order to pay his/her bills, a mobile telephone subscriber enters the payment (bill) information and the payee's account number into the mobile payment part (10) which is included in his/her mobile telephone (1) or (6). After having dialled the telephone number of computing station (2) which is based in the bank (3), the payment information will be sent to the computing station (2) via a mobile communications network (4). In the computing station (2) the calling party's identity will be checked and then the payment will be transferred from the calling party's bank account to the payee's account and then both the calling party and the payee will be informed about the relevant payment. In this invention, the portable telephone is also equipped with a small charge slip printer which can print a receipt for customers of retail businesses.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

Real Time Tele-payment System

This invention is a mobile payment system that relates to payments of bills of the mobile users, or providing the mobile users with the information about their bank account balance, the statement of account, or the movement on the account in a real time basis, by using their portable telephones under any wireless telecommunications systems.

BACKGROUND OF THE INVENTION

There are several mechanical and electronical payment systems for retail business operations like, for example, what is introduced by US patent US-A-5 202 825, in which a hand-held data terminal generates a record of purchases made by a customer for charging a customer in accordance with customer-indicated payment preferences. In these systems the waiter sends by use of a portable data terminal the customer's order to a customer service station which is a typical cash register based in the restaurant. These systems reduces the time requirements for processing customers at check-out counters in comparison with those of more traditional check-out procedures of the recent past. These systems are only for sending the customer order to the cash register in the retail business.

On the other hand in the fixed telecommunications networks a user (subscriber) can be connected from his personal computer to his/her bank via telephone lines and thereby pay his/her bills. In such systems user must use a data modem between his/her computer and the telephone wire. Another disadvantage of such systems is that in order to pay his/her bills, user must have access to a personal computer connected to the fixed telephone infrastructure, therefore user mobility in such systems is completely limited. Before this invention, there was no solution that provides the portable/mobile telephone users with the possibilities to pay their bills by using their personal portable telephones. There was also no payment system, based on use of portable or mobile telephones, that could provide the retail or trading businesses with the possibility to charge their customers in a real time basis; transferring the charges from the customer's account to the account of the retail businessman. There continues to exist a need to further improve the efficiencies of payment systems.

DESCRIPTION OF THE INVENTION

In order to serve such current need, the present invention provides a new and unique mobile payment system. In the inventive system a portable telephone can be used in order to pay bills or transfer money from a bank account to other, or request the bank

for account information. Certain features of the invention are intended as an expansion of value-added services of currently existing mobile communications systems. This invention addresses needs created by users mobility. For example, suppose that you are travelling and you want to pay a certain bill or transfer some amount of money from your bank account to another person's account but you do not have time for going to the bank or the bank may be closed and you may neither have access to your personal computer (which can be connected to the bank via telephone wire). This invention provides you the possibility to pay your bills, by using your portable telephone while you are in move, regardless of are banks closed or not, regardless of if it is night or weekend etc. This invention also provides the retail businesses (for example restaurants) the possibility to charge their customers, via wireless telecommunications networks, by using only the portable telephones. For example, a waiter in a restaurant, after having entered the amount of payment and customer's information (like account number etc.) to his/her portable terminal can send the payment information to the inventive computing station, which is located in the bank. In the computing station the customer's bank account will be charged in accordance with the payment amount received from the waiter's portable telephone. The most important advantage gained by the inventive system is that all mobile telephone subscribers can pay their bills by using only their normal mobile telephones (in which the mobile payment part is included) and their subscriber identity or codes, without requiring any additional data modem, personal computer, and credit cards etc. In this invention the subscriber identity and codes function as the credit card or bank card of the portable terminal's user.

By implementing the inventive mobile payment system a mobile user (subscriber) can pay all his/her bills and handle all his/her banking issues by only using his/her mobile telephone and subscriber identity or codes, where ever under the coverage of a wireless communications network. These and other improvements and advantages are realised by providing a portable telephone (hereafter called portable terminal) including the inventive mobile payment part, and a computing station which is based in the bank. The present invention will now be described by way of examples with reference to the accompanying drawings, in which:

Fig. 1 is a schematic representation of the inventive Real Time Tele-payment System.

Fig. 2 represents, as an general example, a payment flow diagram between the portable terminal and the computing station, which is located in the bank.

Fig. 3 represents, as an general example, a payment flow diagram in which a mobile user pays his/her bills or request the statement of his/her account by using his/her own

portable telephone. In this figure also the payee is informed about the reception of a payment.

Fig. 4 is a schematic representation of two type of portable terminal: one is a normal portable telephone that includes the inventive mobile payment part, and the other is a portable telephone that includes the inventive mobile payment part, a charge slip printer and a user-friendly SIM card reader (SIM: Subscriber Identity Module).

When a mobile user wants to pay a bill or transfer money from an account to other, he/she enters all information required for payment (like his/her account number, the payee's account number, payment's due date, bill's reference number, etc.) to the mobile payment part of his/her portable terminal **1** (for example through the keypad). As it is the object of this invention, the user's own account information dose not need to be entered into the mobile payment part if the computing station **2**, based in the bank **3**, can identify the calling party. This needs that the user information (identity) should be confirmed by his/her telephone operator or service provider in a wireless communications network **4** and then be sent to the bank as a confirmation of user (subscriber) identification. More precisely, user identity can be sent by user's telephone operator or service provider to the computing station **2** when portable terminal **1** set-ups a call or a short message to the computing station **2**. Monitoring a calling party's subscriber number or information at a receiving terminal is a feature provided by today's digital telephone systems. In this invention, in order to implement such procedure, for example the switching systems at the mobile network side can be used so that only when a user set-ups a call or sends a message (by using short message services of the mobile communications systems) to the computing station **2** his/her identity can be monitored in the computing station **2** in order to identify who is the calling party. Therefore, in this invention the computing station **2** receives at least the confirmed user identity from the user's telephone operator or service provider of a wireless communications network (WCN) **4** in order to identify who is in charge for payment of bills sent by portable terminal **1**. Other required information like passwords or access codes to the user's bank account will be sent by user through his/her portable terminal **1**. In today's mobile telecommunications systems the user identity, included in his/her SIM card, is checked and confirmed by network **4** every time his/her portable terminal **1** is turned on and attached to the telephone network **4**. Since the user identity, transmitted from the portable terminal **1** to the network **4**, is completely encrypted and secured therefore the payment messages between portable terminal **1** and computing station **2** are also quite secured because of: first, the security algorithms used in the today's digital wireless telecommunications systems and mobile telephones, and secondly, because of the user's password or access codes used for payment messages in the inventive mobile payment system. All kind of wireless

communications networks can be used in order to communicate the payment messages between the portable terminal and computing station. For example if in the restaurants there is a cordless network like DECT (Digital European Cordless Telephony) **5** then the portable terminal **6** can be connected through such network and PSTN (Public Switched Telephone Network) or ISDN (Integrated Services Digital Network) **7** to the computing station **2**.

The payment question-answering procedure between the user and portable terminal **8** is entered by using the user interface **9** and received and handled by the inventive mobile payment part **10**. The payment information entering procedure **11** is an interactive procedure between the mobile payment part **10** and the user through user interface **9**. Then, the computing station's telephone number will be dialled **12** (either automatically or by user) which after the portable terminal **8** sends the required information for call set-up to the wireless communications network **15** and then payment messages **13** to the computing station **14** via the same network **15**. If the portable terminal **8** dose not send the user (telephone subscriber) identity to the computing station **14**, then the wireless communications network **15** confirms and sends the user identity to the computing station **14** either directly or through the fixed public network **16**. The computing station **14** checks the calling party's account and account number of payee (the account to which the payment should be transferred) and then transfers the required amount of payment from the payer's account to the account of payee **17**. After that the payment has been completed the computing station **14** sends a message **18** to the portable terminal **8** indicating "payment completed" or if there is not enough credit (money) in the payer's account a "No effects" message **19** will be sent to the portable terminal **8**, meaning that the payment can not be accepted. For retail businesses, portable terminal includes also a charge slip printer **20**. If the portable terminal receives a "payment completed" command **18**, the charge slip printer **20** prints a receipt for the customer. In this invention for the retail and trading businesses, the customer's SIM card **39** is entered in the SIM card reader **36** of the portable terminal **1** (of a waiter in a restaurant, for example) temporary in order to pay the bill. Then the portable terminal **8** will be connected to the wireless communications network **15**. The account number of payee (for example account number of the restaurant) can be saved in the memory of his/her portable terminal in order to reduce the information entering procedure of the mobile payment part. This means that only the payment amount should be entered to the mobile payment part. After that the payment amount has been entered to the mobile payment part **10** and the computing station's **14** telephone number has been dialled **12**, the wireless network **15** sends the customer's identity, which can be the subscriber identity or a different code,

to the computing station **14**. The computing station **14** can identify the calling party (the payer) because it has received the calling party's identity from the wireless network **15** and compared with the calling party's identity based in the computing station **14**. Therefore the calling party will be charged for the payment amount received from the portable terminal **8**. The subscriber identity sent from the wireless network **15** to the computing station **14** can be different than the payer's identity sent by the portable terminal **8** to the wireless network **15** but both of these identities belong to one user (subscriber). Alternatively the payer's identity, included in his/her SIM card **39** or entered to the portable terminal by using user interface **9**, can be sent directly from the portable terminal **8** to the computing station **14**. It should be understood that for the simplicity of the description, messages for outgoing call set-up and incoming call or short message services procedures are not explained with details since these procedures are already well known in the mobile communications systems.

Following is an example, in which a mobile user pays his/her bills or transfers money form his/her bank account to other, or ask the bank for statement of account, by using his/her own portable telephone.

First, the payer enters the bill's information **22** (for example: account number of payee, the amount of money which should be transferred, due date of the bill, reference number **11**) to the mobile payment part **21** of his/her portable terminal **41**. Then, after activating an OK function by user, the mobile payment part dials **23** the telephone number of the computing station located in the bank **24**, which after the mobile payment part **21** sends the payment information **25** to the computing station **24**, via a wireless communications network (WCN) **26** and fixed network **27** (PSTN/ISDN). Then, computing station **24** transfers the amount of payment, mentioned on the bill, from the payer's account to the payee's account **28**. Then, computing station **24** sends a "Payment Completed" message **29** to the portable terminal's mobile payment part **21**. If the payee has also a portable terminal **37**, then also his/her mobile payment part **42** would receive a "Payment Reception message" **30**, from computing station **24**, indicating the amount of payment, the payer and the payment date. However, before dialling the number of computing station, the mobile payment part may ask the payer (the user of portable terminal) "Any other payment ?" **31**. The answer can be respond by activating "Yes/No" function **32** or OK function of the mobile payment part **21**. Then the user can enter another bill information to the mobile payment part **21** and when all information required by mobile payment part has been provided, the telephone number of computing station **24** will be dialled **23**. After this, all bills information (payment messages) will be sent to the computing

station in the bank **24** as explained above. Furthermore, there is a command **33** "Send the Statement of Account" in the mobile payment part **21** for requesting the account balance, the statement of account, or the movement on the account from the computing station **24**. When a user selects such command **33**, the mobile payment part **21** sends this message **33**, either by setting up a call or by using the short message facilities of mobile communications networks **26** to the computing station **24**. Then computing station **24** sends the required account balance or the statement of account **34** to the mobile payment part **21** of the portable terminal **41**. The computing station **24** also sends a "Monthly Statement of Account" **35**, to the portable terminals **41, 42** once or twice per month. Then portable terminal's printer **38** can print it for the user to be filed as a record, if required.

Following is an example in which the payee (for example a restaurant or a retail seller) has a portable terminal by which the payer's (a customer) account can be charged.

Suppose that a customer wants to pay his/her bill in a restaurant for the service he/she has received. The customer can give his/her SIM card **39** or credit card to the waiter to be entered to the waiter's portable telephone **1, 8**. Then waiter dials the telephone number of computing station **14**, or the number will be dialled automatically after the SIM card **39** or credit card has been read by the SIM card or credit card reader **36** of the waiter's portable terminal. For example the telephone number of computing station **14** can be saved in the memory of the portable terminal of waiter, and every time a customer's SIM or credit card **39** is entered to the portable terminal **1**, the portable terminal automatically contact the computing station **14**, after having registered in the network **15**. In the bank, the computing station **14** checks the account information of payer (a customer) and then transfers the transaction amount (the sum on the bill) to the payee's (the restaurant) account **17**. If the payer's account do not have enough credit (money) the portable terminal **8** may receive a "No effects" message **19**, or the bank may pay the transaction's amount on behalf of the payer and then later charge the payer or his/her bank for the prepaid transaction. On the other hand if the payer's account information (account number, account identity) is false the computing station **14** may send a "transfer not accepted" message to the payee's portable terminal, which means that the payer (customer) should pay the amount of transaction in cash. If the portable terminal receives from the computing station **14** a "payment completed" message **18**, then the charge slip printer **20** prints a receipt for the customer, as explained in the first example.

It should be considered that in all above-mentioned examples, payment messages can be sent and received either by setting up a call between the portable terminal and computing station or by using short message services facilities of the wireless communications networks.

In the current mobile communications systems, like GSM, there is a facility called "Short Message Services, (SMS)". In SMS a mobile telephone user can send short messages to another subscriber without setting up an interactive call. In order to send the payment messages by SMS, the software of SMS installed in the portable terminal can be modified so that it can also handle the payment parameters and/or commands of the inventive mobile payment part **10**. Then by using the SMS services of the wireless communications network **15**, the bill's information **13** can be sent to the computing station **14**. When computing station **14** receives such payment message **13** sent by SMS, it also generates a message to be sent to the portable terminal in order to inform it if the payment has been completed **18** or not **19**. However, if a user wants to pay many payments (bills) at once and receive also balance or statement of his/her bank account from the computing stations, such long message can be divided to smaller parts and then be combined at the portable terminal or computing station. This means that each bill information can be sent separately using the short message services. This action is transparent to the user of portable terminal. For example several payment information can be entered to the mobile payment part **10**. Then when user selects the "Send" function **40** on the portable terminal **1**, each bill will be sent by one short message in accordance of short messages length. For example, a short message may not include more than 100 letters. If a payment message or the statement of account (sent by computing station) needs more than the assumed 100 letters, then such long information will be divided into two or several short messages and then will be sent one by one to the portable terminal or computing station.

In this invention computing station can send and receive messages either via PSTN (Public Switched Telephone Network) and ISDN (Integrated Services Digital Network) and other fixed networks or via only a wireless communications network. The computing station includes all means for transmitting and receiving payment and banking messages via the wireless networks.

It is to be understood that various changes and modifications can be made to alter the specifically described structure or methods of operation of the preferred embodiment without departing from the spirit and scope of the invention. This invention is to be defined only by the scope of the claims appended hereto.

Claims

1. A mobile payment system (43), characterised in that it is comprised of:

- at least one portable terminal (1, 6, 8), such terminal including a mobile payment part (10, 21) and other means for entering, transmitting, receiving and printing of information relating to: the payments of bills of the telephone subscriber or the user of said portable terminal; transferring of money from the bank account of the subscriber or user to the others account; sending and receiving payment messages (13, 18, 19, 25, 29) or messages including the account balance, the statement of account, or the movement on the bank account (33, 34, 35) of the telephone subscriber or the user of the portable terminal (41, 37);
- at least one computing station (2, 14, 24) which is located in the bank (3), said computing station including means for communicating with said portable terminal and for transferring the amount of payment (money) from the bank account of portable terminal's user and/or telephone subscriber to another bank account (17, 28), or from a customer's bank account, whose account information is entered into said portable terminal, to the calling party's account; and to receive and send messages about the account balance, the statement of account, or the movement on the bank account (33, 34, 35) of the portable terminal's subscriber or user;
- at least one wireless communications network (4, 15, 26) through which said portable terminal can send and receive to or from said computing station said payment messages or messages about the account balance, the statement of account, or the movement on the bank account of said portable terminal's subscriber or user.

2. A mobile payment system (43) according to claim 1, characterised in that said at least one portable terminal (1, 6) is a first plurality of portable terminals, and in which the number of said portable terminals in said first plurality of portable terminals is greater than said at least one computing station (2).

3. A mobile payment system (43) according to claim 1 and 2, characterised in that the payments or bills of a mobile telephone subscriber can be paid by entering the subscriber identity and codes into said portable terminal (1, 6, 8, 41) and the bill's information, including the payee's bank account number, the amount of payment, bill's due date and reference number into the mobile payment part (10, 21) of said portable terminal, and by setting up a call or a short message to the bank's computing station

(2, 14, 24) and sending the payment (bill's) messages (13, 25) to said computing station (2, 14, 24).

4. A mobile payment system (43) according to claim 1, 2 and 3, characterised in that said at least one portable terminal (1, 6) comprises all means for transmitting and receiving payment messages to or from said computing station (2); and that:

- said portable terminal includes a mobile payment part (10, 21) for handling the payment information (11, 22, 31, 32) entered by user to said portable terminal, and that said payment information can be saved into the memory of said portable terminal and be sent to said computing station, whenever required; and that:
- said portable terminal receives a message (18, 19, 29) from said computing station indicating that either the payment or transferring of the required amount of payment from the payer's to the payee's bank account has been accepted and/or completed or not.

5. A mobile payment system (43) according to claim 1, 2, 3 and 4 characterised in that the user of said portable terminal can enter more than one payment or bill information to the mobile payment part (10, 21) , and that after that telephone number of said computing station based in the bank (2, 14, 24) has been dialled (12, 23) either manually or automatically, all required payment information (13, 25) will be sent to said computing station; and that:

- said portable terminal can send payment (bill's) information (13, 25), handled in mobile payment part (10, 12), to the computing station (14, 24) and receive the required payment messages (18, 19, 29) from said computing station by setting up a call or using the Short Message Services (SMS) of the wireless communications network (4, 15, 26); and that
- said portable terminal's subscriber information can be sent from the user's telephone operating network (4, 15, 26) to the computing station (2, 14, 24); and that
- said portable terminal includes a charge slip printer (20, 38) that can print all payment information and the information received from said computing station for user of said portable terminal, and that,
- said mobile payment part (10, 21) can be included into any kind of digital or analogue portable telephone that is capable of operating in cellular communications systems.

6. A mobile payment system (43) according to claim 1 - 5, characterised in that said computing station (2, 14, 24) after receiving a payment message (13, 25) from said portable terminal (1, 6, 8, 41), checks and charges the payer's account (17, 28) in accordance with the payment amount received from said portable terminal and then sends a message (18, 19, 29) to said portable terminal (8, 41, 37) in order to indicate that payment has been accepted and/or completed or indicating that there is not enough credit in the payer's account; and that:

- said computing station (2, 14, 24) can receive or send payment messages (18, 19, 29, 30) or other banking messages (33, 34, 35) to said portable terminal (1, 6, 8, 41) via either fixed and wireless communications networks (4, 7, 15, 16, 26, 27) or via only wireless communications network (15, 26); and that,

- said computing station (2, 14, 24) can receive the payer's information and identity either from the payer's telephone operator or service provider through wireless communications network (4, 15, 26) when payer telephones or send messages (13, 25) to said computing station (2, 14, 24) or from the payer's portable terminal (1, 6, 8, 41); and that, the payer's information received from said payer's telephone operator or service provider or from said portable terminal may include payer's subscriber information or identity or any other required information; and that,

- said computing station can monitor the subscriber information or other identity, received from said payer's telephone operator or service provider or portable terminal, and based on said subscriber information or other identity and account number transfer the required amount of payment (money) from the payer's account to any other required account; and that,

- said subscriber information or identity will be confirmed by subscriber's telephone operator or service provider (4, 15, 26) and said confirmed information will be sent to said computing station (2, 14, 24) in which the subscriber identity will be checked (17, 28) and based on that, the received payment message (13, 25) can be accepted and a payment completed message (18, 29) will be sent to said portable terminal (1, 6, 8, 41); and that,

- said computing station (2, 14, 24) can send or receive payment messages (13, 18, 19, 25, 29, 30, 33, 34, 35) to or from the portable terminals (1, 6, 8, 41, 37) of both the payer and the payee; and that,

- said computing station (2, 14, 24) is equipped with all means for transmitting and receiving messages via any wireless communications network, to or from said portable terminal (1, 6, 8, 41, 37).

7. A mobile payment system (43) according to claim 1 - 6, characterised in that the mobile payment part (10, 21) may ask the user to enter all payment information

(11) such as payee's account number, bill's reference number, bill's due date, the amount of payment and other required information; and that:

- said mobile payment part (10, 21), after receiving all information about a payment or a bill from the user through user interface (9), may ask the user of said portable terminal "any other payment ?" (13) indicating dose user wants to pay another bill or payment; and that,

8. A mobile payment system (43) according to claim 1 - 7, characterised in that said portable terminal (1, 6, 8, 41) can be used in order to pay the bills of any mobile telephone subscriber by entering each subscriber's identities and codes into said portable terminal either by using the portable terminal's user interface (9) or the SIM card (39) and card reader (36); and that:

- said mobile telephone subscriber's codes can be different than said subscriber's identities; and that said subscriber codes can be included both in the subscriber's SIM card (39) and said computing station (2) located in the bank (3); and that:

- said portable terminal (1, 6) can be used in order to charge customers, in retail or trading businesses, by entering the customers' telephone SIM card (39) into said portable terminal (1, 6) and by using the telephone subscriber identities of each customer as an identification for payment; and that:

- after that said customer's SIM card (39) has been entered to said portable terminal (1, 6, 8), said portable terminal will be re-connected to the wireless communications network (4, 15) in order to check the subscriber identity, which after the customer's (subscriber's) bank account can be charged by sending payment messages (13) to the computing station (2, 14).

AMENDED CLAIMS

[received by the International Bureau on 25 March 1996 (25.03.96); original claims 1 and 3-8 amended; new claims 9 and 10 added; remaining claims unchanged (8 pages)]

1. A mobile payment system (43), utilizing the Short Message Services (SMS) facilities of mobile communication networks such as GSM (Global System for Mobile Communications), and subscriber identity such as SIM card (Subscriber Identity Module), and a new mobile-telephone-based functionality and mobile network architecture **characterized** in that it is comprised of:

- at least one portable terminal (1, 6, 8), such terminal utilizing the inventive Mobile Payment Part (10, 21), which provides a function and SMS-based adaptation and application part integrated into said portable terminal to provide at least an alphanumeric payment (bill) inquiry (e.g. 11), and including other means for entering, transmitting and receiving, and printing of the information mainly related to: the payments of bills of the telephone subscriber (1, 6, 8); transferring of money from the bank account of the subscriber or user to the others account; sending and receiving payment messages (13, 18, 19, 25, 29) or messages including the account balance, the statement of account, or the movement on the bank account (33, 34, 35) etc. of the telephone subscriber of the portable terminal (41, 37) without requiring to use any additional data modem to be used in conjunction with said portable terminal for transmission and reception of said payment etc. messages;

- at least one computing station (2, 14, 24) which is located in the bank (3), as it is the object of the architecture of the inventive payment system (43), said computing station includes all information about the portable telephone subscriber data which is connected to the subscriber's bank account in the same bank wherein computing station is located, and said computing station includes means for communicating with said portable terminal (4) and transferring the amount of payment (money) from the bank account of the payer (i.e. the calling subscriber) to another bank account (17, 28), and to receive and send messages about the payments, account balance, the statement of account, the movement on the bank account (33, 34, 35) or other banking messages etc. of the calling subscriber via SMS facilities of the wireless communication network (4);

- at least one wireless communication network (4, 15, 26) equipped with Short Message Services (SMS) infrastructure through which said portable terminal (1, 6, 8) can send and receive to or from said computing station said payment messages or other banking messages etc. and that said wireless communication network can confirm (i.e. authenticate) the subscriber identification for said computing station, whenever required or transfer the subscriber data received from said portable terminal directly to said computing station, in which the subscriber data can be compared with the subscriber data already recorded there.

2. A mobile payment system (43) according to claim 1, **characterized** in that said portable terminal (1, 6) is a first plurality of portable terminals, and in which the number of said portable terminals in said first plurality of portable terminals is greater than said at least one computing station (2).

3. A mobile payment system (43) according to claim 1, 2, **characterized** in that said portable terminal (1, 6, 41) comprises all means for transmitting and receiving payment etc. messages to or from said computing station (2) or other portable terminal (37); and that,

- said portable terminal includes the inventive Mobile Payment Part (10, 21) which is a short-message-based adaptation and application part for handling, dividing or connecting the payment etc. information (11, 22, 31, 32), and that said payment etc. information can be saved into the memory of said portable terminal and be sent to said computing station, whenever required; and that,

- after that portable terminal has been registered into the mobile network (4), the payments or bills of the mobile telephone subscriber (1, 6) can be paid by entering the bill's information such as the payee's bank account number, the amount of payment, bill's due date and reference number etc. into the Mobile Payment Part (10, 21), and by sending the short messages (e.g. 13, 33) to the bank's computing station (2, 14, 24) via SMS facilities of the mobile network (4) and receiving messages such as (18, 19, 30, 34, 35 etc.).

- said portable terminal receives a message (e.g. 18, 19, 29) from said computing station indicating that either the payment or transferring of the required amount of money from the payer's to the payee's bank account has been accepted and/or completed or not; and that,

- said portable terminal includes a charge slip printer (20, 38) that can print all payment information and the information received from said computing station for user of said portable terminal, whenever required.

4. A mobile payment system (43) according to claims 1, 2, 3, **characterized** in that the computing station (2, 14, 24) after receiving a payment message (13, 25) from said portable terminal (1, 6, 8, 41), checks and charges the subscriber's (payer's) account (17, 28) in accordance with the payment amount received from said portable terminal and then sends back a message (e.g. 18, 19, 29) including all information about the payment (e.g. bill reference, payer, amount etc.) to said portable terminal (8, 41, 37) in order to indicate that the payment has been accepted and/or completed or indicating that there is not enough credit in the payer's account; and that:

- said computing station (2, 14, 24) can receive or send payment messages (e.g. 18, 19, 29, 30) or other banking messages (e.g. 33, 34, 35) or any other message to said portable terminal (1, 6, 8, 41) via SMS of a mobile communication network (4, 5) through either fixed and wireless communication networks (4, 5, 7, 15, 16, 26, 27) or via only wireless communications network (4, 15, 26); and that,

- said computing station (2, 14, 24) can receive the payer's identity either from the payer's telephone operator system (4, 15, 26) when payer sends messages (e.g. 13, 25) to said computing station (2, 14, 24) or from the payer's portable terminal (1, 6, 8, 41); and that said payer's data received from said payer's telephone operator or from said portable terminal may include the payer's subscriber data or identity parameters or any other required information; and that,

- said subscriber data can be confirmed (i.e. authenticated) and secured either in the databases and infrastructure of the subscriber's telephone operator (4), or in said computing station, for example, by utilizing the algorithms used in mobile communication systems such as those of the GSM; and that,

- after that the subscriber data, communicated between said portable terminal and wireless communication network or directly between said portable terminal and computing station has been authenticated, the Mobile Payment Part (10) of the portable terminal or said computing station can send and/or receives payment etc. messages through SMS of a mobile communication network (4); and that,

- said computing station can monitor the subscriber identity, number etc., received alternatively from said payer's telephone operator or said portable terminal, and based on said subscriber identity and/or number and checking of his/her bank account number transfer the required amount of payment (money) from said payer's account to any other required account; and that,

- said subscriber data can alternatively be confirmed or sent by the subscriber's telephone operating network (4, 15, 26) to the computing station (2, 14, 24); as a confirmation of subscriber identification, enabling said computing station to compare the received subscriber data with the data already recorded in said computing station, and when subscriber data is compared and accepted by said computing station, the portable terminal can send payment messages to said computing station; and that,

- said subscriber data may include the subscriber telephone number, confirmed by mobile operator (4), or it may consist of the subscriber identity incorporated in SIM card, or any other code; and that,

- said computing station (2, 14, 24) can send or receive payment messages (e.g. 13, 18, 19, 25, 29, 30, 33, 34, 35) to or from the portable terminals (1, 6, 8, 41, 37) of both the payer and the payee; and that,

- said computing station (2, 14, 24) is equipped with all means for wired or wireless transmission and reception of messages communicated between said computing station (2), wireless communications network (4 or 5), and said portable terminal (1, 6, 8, 41, 37).

- said computing station may send e.g. a monthly report (e.g. 35) to said portable terminal (1, 6, 37, 41) to be displayed or printed (20, 38), for said subscriber, as a receipt and bank report for payments (bills, etc.) charged from the subscriber/payer account to the other subscriber/payee account, by said computing station.

5. A mobile payment system (43) according to any preceding claims, **characterized** in that the subscriber, for example, a waiter etc. in a restaurant can send the payment messages (e.g. a bill) by using the inventive portable terminal (1, 6, 8) either to the computing station (2, 14) or directly to the customer's portable terminal (e.g. a mobile telephone integrated with the inventive Mobile Payment Part), via SMS facilities of mobile communication network (5, 4), which after the payment can be accepted by said customer and be sent to the computing station (2) in which the payment procedure will be completed and then a message (including the bill's information) will be sent to both customer's and waiter's portable terminals indicating that either the payment has been completed and/or accepted (29, 30) or refused (19); and that:

- said waiter etc. or customer can enter the customer's identity code to said waiter's portable terminal, by using user interface (9), and then send the bill together with the customer's code to the computing station, which after said computing station generates a message and sends it to the customer's portable terminal to be accepted by the customer, and that after that the payment has been completed in the computing station, the computing station can send a message such as "Payment Reception" including all information about the payment (e.g. bill reference, payer, payment amount etc.) to the payee's terminal indicating that the payee has received the payment; and that,

- said portable terminal (1, 6) can be used in order to charge customers, in retail or trading businesses, by entering alternatively the customers' telephone SIM card (39) into said portable terminal's SIM card reader (36) and by using the telephone subscriber identity of each customer as an personal identification for payment; and that:

- after that said customer's SIM card (39) has been entered to said portable terminal (1, 6, 8), said portable terminal will be re-registered to the wireless communications

network (4, 15) and/or said computing station in order to check the subscriber identity, which after the customer's (subscriber's) bank account can be charged by sending payment messages (e.g. 13) to the computing station (2, 14).

6. A mobile payment system (43) according to any preceding claims, **characterized** in that whenever the subscriber turns on his/her portable terminal (1, 6) the Mobile Payment Part (10) sends the subscriber data, that can be included in the SIM card, to the computing station (2, 14) through an available wireless communication network (4, 5), and after that registration process between said computing station, said wireless communication network and said portable terminal (1, 6, 8) has been completed said portable terminal can have access to said wireless communication network through which it can send and/or receive payment, banking etc. messages to/from said computing station, and also be able to use the telecommunications services like voice etc. of said wireless communication network; and that,

- after said portable terminal has been registered in said wireless communication network (4) or computing station (2), the subscriber of said portable terminal can send and/or receive banking messages (e.g. 33, 34, 35) or can pay his/her bills by sending and receiving the payment etc. messages (e.g. 13, 18, 19, 25, 29) to the computing station (2, 14) or to another portable terminal, through the Mobile Payment Part (10) of his/her portable terminal; and that,

- said subscriber data can be a data which is recorded only in the SIM card and in said computing station that is located in the bank; and that,

- said subscriber data can be either similar to or different from that subscriber identity which is incorporated in the subscriber's telephone SIM card provided by mobile operators (4); and that,

- said subscriber data can be alternatively sent to said computer station after that registration of said portable terminal into said wireless communication network (4, 5,) has been completed, which after the subscriber can send and/or receive payment/bill messages to said computing station via SMS of said wireless communication network (4, 5,).

7. A portable terminal (1, 6, 8, 41, 37) according to any preceding claims, **characterized** in that it includes the inventive Mobile Payment Part (10, 21) which for each payment procedure may ask the subscriber (i.e. the payer) to enter all payment information (e.g. 11) such as payee's account number, bill's reference number, bill's due date, the

amount of payment and other information included in the bill or required for payment procedure: and that:

- said Mobile Payment Part (10, 21), after receiving all information about a payment or a bill from the user through user interface (9), may ask the subscriber of said portable terminal e.g. "Any other payment ?" (13) indicating dose subscriber wants to pay another bill or payment, and that after this message subscriber can enter other payment information into said Mobile Payment Part: and that,

- said portable terminal (1, 6, 8, 41) can be used in order to pay the bills of any mobile telephone subscriber by entering each subscriber's identities and codes into said portable terminal either by using the portable terminal's user interface (9) or by entering the SIM card (39) and card reader (36); and that:

- more than one payment or bill etc. data can be entered into said Mobile Payment Part (10, 21) of said portable terminal, and that after that telephone number of said computing station based in the bank (2, 14, 24) has been dialed (12, 23) either manually or automatically, all required payment information (e.g. 13, 25) will be sent to said computing station via SMS facilities of the mobile network (4, 5, 15, 26); and that,

- said portable terminal (1, 6, 8, 37, 41), includes all means of a mobile/cellular/cordless telephone for receiving and transmitting voice and data so that said portable terminal can function both as a mobile payment device and as a mobile/cellular/cordless telephone without requiring any data modem to be used in conjunction with the transmission and reception of payment etc. messages: and that,

- said Mobile Payment Part (10, 21) can be integrated into any kind of portable telephone that is capable of operating in cellular communications systems: and that,

8. A portable terminal (1, 6, 8, 41, 37) according to any preceding claims, **characterized** in that a small printing device (20, 38) is integrated into said portable terminal (1, 6, 37, 41) for printing any data received from computing station (2) or other portable terminals or any other source or the messages entered into said Mobile Payment Part (10, 11, 21) by its user or any other short messages received by said portable terminal.

9. A Mobile Payment Part (10) according to any preceding claims, **characterized** in that it is a component and function integrated into the portable terminal (1, 6), said Mobile Payment Part provides a payment (bill) inquiry (11) procedure including for example questions (such as payment amount, Bill reference, Receiver's account number, Due date, Recipient etc.) which can be displayed on the display (6) and which can be answered by the

user of the portable terminal through the user interface (9) and such payment information can be saved into the memory of the portable terminal or be sent to the computing station (2) or another portable terminal via SMS; and that

- said Mobile Payment Part (10) can be either integrated into said portable terminal as a component including the required soft-ware for providing said bill inquiry (e.g. 11), or said Mobile Payment Part can be integrated into the SIM card (i.e. Subscriber Identity Module) to provide said bill inquiry whenever subscriber wants to pay a bill or perform a payment; and that,

- said Mobile Payment Part is a function and SMS-based adaptation, integrated into said portable terminal or alternatively into said SIM card to provide an alphanumeric payment (bill) inquiry (11) procedure; and that,

- said Mobile Payment Part (10) can divide and split any long data of any length, for example e-mails done in a personal computer etc. which can be connected to said portable terminal into several short messages and send them to other portable terminals/telephones (e.g. 1 or 6) or to said computing station (2) via SMS facilities of a wireless communication network (e.g. 4 or 5) without requiring any data modem to be connected between the portable terminal and said personal computer, so that said Mobile Payment Part divides such e-mail to several short messages in a numbering sequence, for example, first short message, second short message etc.; and that,

- said Mobile Payment Part (10) is able to connect several short messages originated from a long data of any length e.g. an e-mail according to said short messages' numbers defined in the sender's portable terminal (e.g. 1) and their sender's identity (e.g. subscriber number), and put them into the original order and configure said original long data, which can be a long information sent by computing station or another portable terminal (e.g. 6) or any other source equipped with the inventive Mobile Payment Part (10) via SMS facilities of a mobile communication system (4), and then display said original data (e.g. the e-mail) on the display of the portable terminal (1, 6) or forward it to a separate monitor or personal computer without requiring any data modem to be used between said portable terminal and said personal computer; and that,

- all short messages which are resulted from a longer data and received by said portable terminal (1 or 6) or computing station (2) may contain a short message number which is unique for each message and is defined according to their dividing sequence; and that,

- all short messages which are divided from a longer data and received by said portable terminal (1 or 6) or computing station (2) , through SMS infrastructure (4 or 5).

may contain both the sender's and receiver's identity number (e.g. payer's and payee's subscriber numbers), which can be added to each short message either at said message sending portable terminal (e.g. 1) or at the wireless communication network's SMS facilities (4); and that,

- all short messages which are divided from a longer data and received by said portable terminal (1 or 6) or computing station (2), through SMS infrastructure (4 or 5), may be connected according to their sender's identity and their arrival time to the SMS facilities of a mobile communication system (4) or their sending time from the portable terminal or computing station or any other source; and that such sending or arrival time can be defined either at said portable terminal (e.g. 1), which sends the messages, or at the SMS facilities of the wireless communication network (4).

10. A mobile payment system (43) according to any preceding claims, **characterized** in that the telephone calls made by portable terminal (1, 6) can be charged simultaneously after each or several calls, from the subscriber's bank account (i.e. payer's account) to the wireless communication operator's (4 or 5) account, so that said operator can send the bills relevant to the telecommunications services used by said subscriber, directly to said computing station (2); and that,

- said computing station can include either the subscribers' data and bank account information or both the subscribers' and said wireless communication operator's data and bank account information so that the subscribers' all telephone calls can be charged directly from the subscriber's account to said wireless communication operator's bank account, by said computing station; and that,

- said computing station may send e.g. a monthly report (e.g. 35) to said portable terminal (1, 6, 37, 41) to be displayed or printed (20, 38), for said subscriber, as a receipt against charged calls or any telecommunications services used by said subscriber and charged from said subscriber bank account to the wireless communication operator's (4 or 5) bank account, by said computing station.

STATEMENT UNDER ARTICLE 19

Hereby we would like to file and publish the attached Amendment together with the above application. The claims filed are amended in order to better define the scope of the claims for the purposes of provisional protection. All claims are amended after that International Searching Report was received by the applicant so that the amended claims define the scope of the claims mainly based on using the second alternative (i.e. Short Message Services facilities, see page 7 of description). Moreover, it was noticed that the filed claims could not cover all objects of the above-mentioned application without applying for amendment. All claims amended here fall into the description of the invention, and go not beyond the disclosure in the above international application as filed. The differences between the claims as filed and as amended are indicated in the next page.

FIG 2

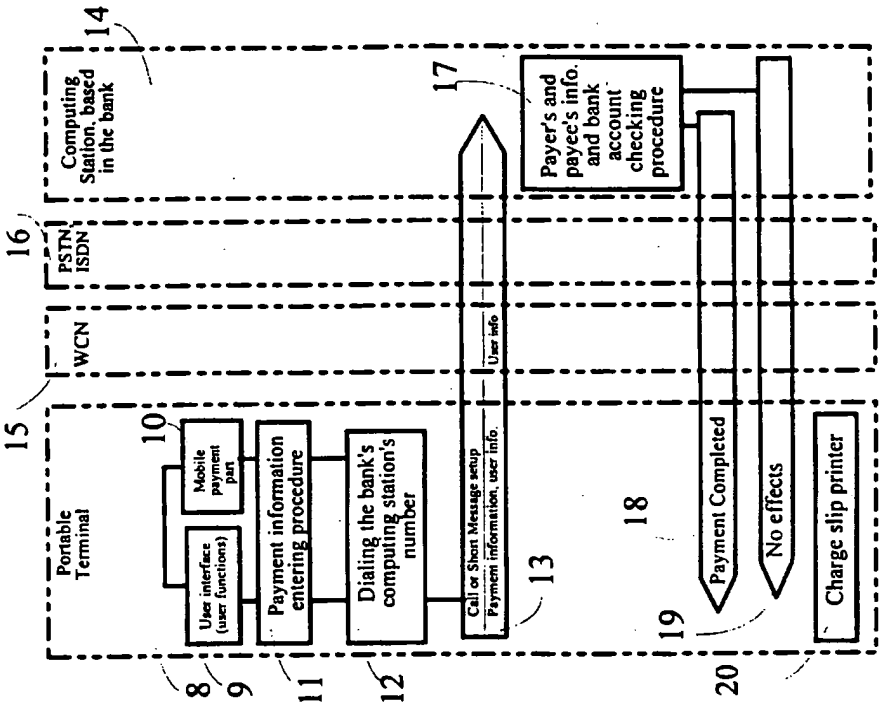

Portable Terminal
WCN
PSTN/ ISDN
Computing Station, based in the bank
Mobile payment part
User interface (user functions)
Payment information entering procedure
Dialing the bank's computing station's number
Call or Short Message setup
Payment information, user info.
User info
Payer's and payee's info. and bank account checking procedure
Payment Completed
No effects
Charge slip printer
8
9
10
11
12
13
14
15
16
17
18
19
20

FIG 1

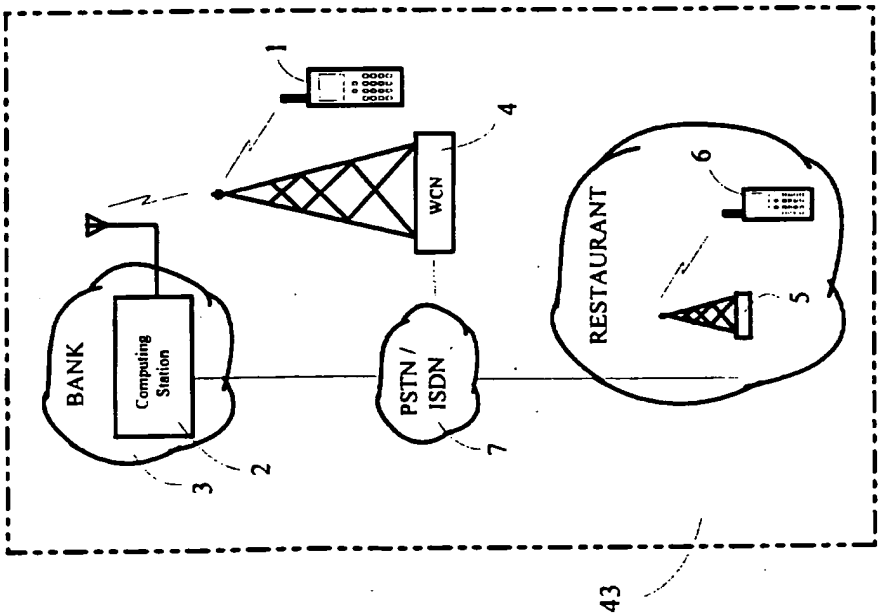

BANK
Computing Station
WCN
PSTN / ISDN
RESTAURANT
1
2
3
4
5
6
7
43

FIG 4

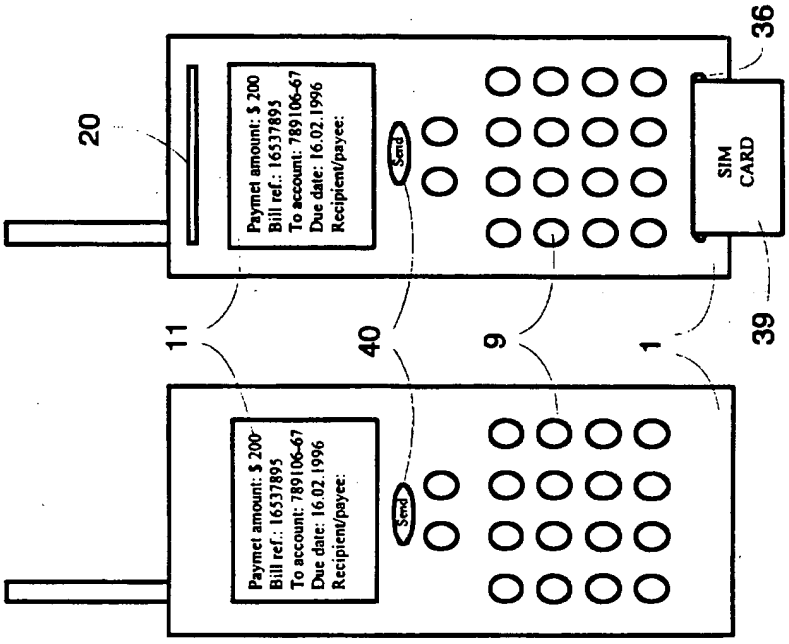
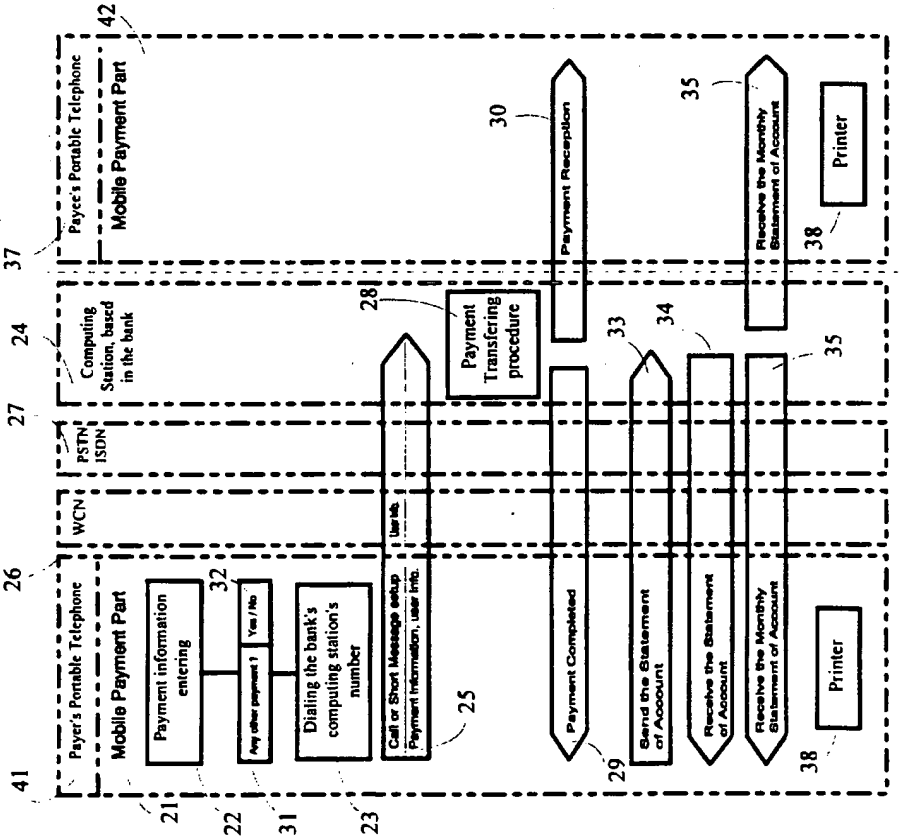

Paymet amount: $ 200
Bill ref.: 16537895
To account: 789106-67
Due date: 16.02.1996
Recipient/payee:
Send
SIM CARD
20
36
11
40
9
1
39

FIG 3


Payer's Portable Telephone
Mobile Payment Part
Payment information entering
Any other payment ?
Yes / No
Dialing the bank's computing station's number
Call or Short Message setup
Payment information, user info.
Payer's Portable Telephone
Mobile Payment Part
WCN
PSTN ISDN
Computing Station, based in the bank
Payment Transfering procedure
Payment Reception
Payment Completed
Send the Statement of Account
Receive the Statement of Account
Receive the Monthly Statement of Account
Printer
Receive the Monthly Statement of Account
Printer
21
22
31
23
25
26
27
24
28
30
29
33
34
35
35
37
42
41
32
38
38

INTERNATIONAL SEARCH REPORT

International application No.
PCT/FI 95/00591

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G07F 7/08, G07F 19/00, G06F 17/60 // G06F 157:00
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G07F, H04M, H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

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.
X	WO 9411849 A1 (VATANEN, H.T.), 26 May 1994 (26.05.94) -- --------	1-8

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

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

Date of the actual completion of the international search	Date of mailing of the international search report
1 March 1996	04-03-1996
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Jan Silfverling Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

05/02/96

International application No.
PCT/FI 95/00591

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO-A1- 9411849	26/05/94	NONE	

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04N 7/16, 7/167 — A1

(11) International Publication Number: **WO 00/46994**

(43) International Publication Date: 10 August 2000 (10.08.00)

(21) International Application Number: PCT/IB00/00163

(22) International Filing Date: 4 February 2000 (04.02.00)

(30) Priority Data:
99400261.6 4 February 1999 (04.02.99) EP

(71) Applicant *(for all designated States except US)*: CANAL+ SOCIETE ANONYME [FR/FR]; 85/89, quai André Citroën, F-75711 Paris (FR).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: MAILLARD, Michel [FR/FR]; 42, avenue du Maréchal Leclerc, F-28130 Maintenon (FR).

(74) Agents: COZENS, Paul, Dennis et al.; Mathys & Squire, 100 Gray's Inn Road, London WC1X 8AL (GB).

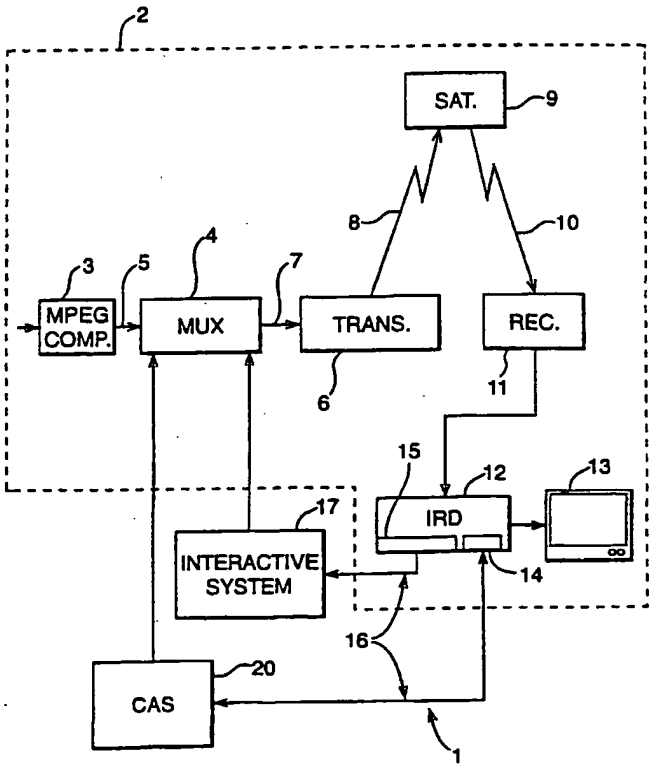
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR ENCRYPTED TRANSMISSION

(57) Abstract

A method and apparatus for encryption of data between a first device (12) and a second device (30), in which one or more precalculated key pairs (41) are stored in a memory of the first device (12), the or each key pair comprising a session key and an encrypted version of the session key. The encrypted version is passed to the second device (30), which decrypts (42) the session key, this session key being thereafter used to encrypt data communicated from the second device (30) to the first device (12) and/or vice versa. The invention is particularly applicable to a digital television system in which data, notably control word data, is to be communicated in encrypted form between a decoder and an associated portable security module.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND APPARATUS FOR ENCRYPTED TRANSMISSION

The present invention relates to a method and apparatus for encryption of messages between two devices, for example a decoder and a portable security module in a digital television system.

Transmission of encrypted data is well-known in the field of pay TV systems, where scrambled audiovisual information is usually broadcast by satellite to a number of subscribers, each subscriber possessing a decoder capable of descrambling the transmitted program for subsequent viewing.

In a typical system, scrambled data is transmitted together with a control word for descrambling of the data, the control word itself being encrypted by a so-called exploitation key and transmitted in encrypted form. The scrambled data and encrypted control word are then received by a decoder having access to an equivalent of the exploitation key stored on a portable security module such as a smart card inserted in the decoder. The encrypted control word is then decrypted on the smart card and subsequently communicated to the decoder for use in descrambling the transmitted data.

In order to try to improve the security of the system, the control word is usually changed every ten seconds or so. This avoids the situation with a static or slowly changing control word where the control word may become publicly known. In such circumstances, it would be relatively simple for a fraudulent user to feed the know control word to the descrambling unit on his decoder to descramble the transmission.

Notwithstanding this security measure, a problem has arisen in recent years where the stream of control words sent during a broadcast becomes known through monitoring of data communicated at the interface between the smart card and decoder. This information may be used by any unauthorised user who has recorded the still-scrambled broadcast on a video recorder. If the film is replayed at the same time as the stream of control words is fed to the decoder, visualisation of the broadcast

becomes possible. This problem has further been exacerbated with the rise of the internet and it is now common to find any number of internet sites that list the stream of control words emitted during a given transmission.

The European patent application PCT WO 97/3530 in the name of Digco addresses this problem by proposing a solution in which the control word stream passed across the interface between the smart card and decoder is itself encrypted with a session key. The session key is generated randomly by the decoder and encrypted with a second key held in the decoder and corresponding to a public key used with a private/public encryption algorithm. The associated smart card possesses the necessary private key to decrypt the session key, which is thereafter used by the smart card to encrypt the control word stream sent from the smart card to the decoder.

As will be appreciated, the use of a locally generated session key to encrypt the control word stream means that the encrypted stream cannot thereafter be fed into another decoder for use in descrambling the data since each decoder will possess a different session key for use in decrypting the control word stream sent from the smart card.

Whilst this solution provides a higher level of security than conventional systems there are nevertheless a number of disadvantages associated with this system.

Notably, the use of a public/private key algorithm is effectively obligatory in such a system since it is not desirable for security reasons to store both a symmetric key and the associated algorithm in the decoder, due to the ease in which this information may be extracted from a decoder memory. This problem does not arise in the case of a public key, since possession of this key does not enable decryption of private key encrypted messages.

It is one object of the present invention to provide a more adaptable alternative to the above known system. However, the invention is not limited to the field of decoder security and, as will be described below, may be applied to a number of other

situations in which secure communication of data is required.

A first aspect of the present invention provides a method of encryption of data communicated between a first and second device, wherein at least one precalculated key pair is stored in a memory of the first device, said at least one key pair comprising a session key and an encrypted version of the session key prepared using a transport key, the encrypted version of the session key being subsequently communicated to the second device which decrypts the encrypted version using an equivalent transport key stored in its memory such that data communicated from at least the second to the first device may thereafter be encrypted and decrypted by the session key in the respective devices.

A preferred embodiment provides a method of encryption of data communicated between a first and second device, characterised in that one or more precalculated key pairs are stored in a memory of the first device, the or each key pair comprising a session key and an encrypted version of this session key prepared using a transport key, the encrypted value of the session key being subsequently communicated to the second device which decrypts this value using an equivalent transport key stored in its memory such that data communicated from at least the second to the first device may thereafter be encrypted and decrypted by the session key in the respective devices.

Unlike the Digco system described above, the use of a precalculated stored pair of values avoids the necessity of having to provide an encryption algorithm within the first device (e.g. the decoder) to encrypt an internally generated session key. As a consequence, the algorithm chosen to encrypt the session key need not be limited to a public/private key algorithm but may correspond to a symmetric type algorithm if desired. Nevertheless, as will be understood, the present invention may also be implemented using public/private key algorithms to encrypt the session key, as will be discussed in further detail below.

Advantageously, a plurality of key pairs are stored in the memory of the first device,

the first device selecting and processing one or more session keys to generate a definitive session key and communicating the associated encrypted value or values to the second device for decryption and processing by the second device to generate the definitive session key.

The provision of a plurality of key pairs within the first device enables the first device to choose and define a different definitive session key for each communication session. In one embodiment, a subset of a plurality of stored session keys is chosen by the first device to generate the definitive session key, the associated encrypted values of these subset session keys being communicated to the second device for decryption and processing.

Depending on the type of operation used, the resulting definitive session key may be dependent on the order of combination of the chosen session keys. In such an embodiment, this order information is communicated to the second device to enable the second device to correctly generate the definitive session key using the associated encrypted values.

For example, an initial session key value known to both the first and second devices may be repeatedly encrypted in both devices by an ordered sequence of session keys using an encryption algorithm sensitive to the order of encryption, such as the DES symmetric algorithm.

Of course, where the first device is using a selected subset of keys to generate the definitive session key, it may not be necessary to also use an order dependent algorithm to generate a changeable definitive session key and the keys may be combined, for example, using a simple arithmetical operation.

In one advantageous embodiment, the one or more precalculated key pair values may be selected from a larger set of precalculated key pairs prior to storage in the first device. For example, the operator or system manager may communicate a large number of precalculated key pairs to the manufacturer of the first device, the device

manufacturer thereafter selecting at random the key pairs to be stored in a given device.

In this way, the key pair or pairs embedded in the first device will be unique to that device, or at least quasi-unique, thereby increasing the level of security for the system. Furthermore, the entity responsible for manufacture of the device need not possess the algorithm or keys used to prepare the encrypted session key values but may be simply supplied with a table of key pairs.

Preferably, the encrypted key value or values communicated to the second device also include a signature value that may be read by the second device to verify the authenticity of the communicated value.

Such a signature value can be generated and verified in accordance with a conventional signature system, for example using combination of hash and public/private key algorithms such as MD5 and RSA, this signature being appended to the key pair values stored in the first device.

Conveniently, the signature value can also be precalculated at the time of calculation of the encrypted key value and thereafter stored in the first device.

In a particularly preferred embodiment, the algorithm and transport key used to encrypt and decrypt the session key or keys correspond to a symmetric algorithm and associated symmetric key. The use of a symmetric algorithm enables an increase in the processing time necessary for the second device to decrypt the session key in comparison with an operation using a public/private key algorithm.

Whilst one of the advantages of the present invention lies in the adaptability of the present system to use a symmetric algorithm, it will be appreciated that this is not obligatory. For example, in an alternative embodiment, the session key or keys may be encrypted by a public key prior to storage in the first device and decrypted by an equivalent private key within the second device.

Further preferably, the encryption algorithm used with the session key to encrypt and decrypt data communicated between the first and second device (or vice versa) corresponds to a symmetric algorithm. The choice of algorithm used may depend on the system requirements such as the need to have bidirectional communication between the devices.

Suitable symmetric algorithms may include DES or even an appropriate proprietary algorithm. Suitable public/private key algorithms may comprise RSA or other similar algorithms.

As mentioned above, the present invention is particularly applicable to the field of digital television and, in one preferred embodiment, the first device corresponds to a decoder and the second device to a portable security module (or vice versa).

The portable security module may conveniently comprise a smart card. If so, the data encrypted with the session key may correspond to simple control word information used by the decoder to descramble broadcast data.

The same principle may also be applied to the case where the descrambling unit in the decoder is implemented as a detachable conditional access module or CAM, broadcast data being descrambled in the conditional access module and communicated to the decoder.

In this embodiment, the first device may thus correspond to a decoder and the second device to a detachable conditional access module. If so, the data encrypted with the session key will normally correspond to the data descrambled by the conditional access module e.g. the broadcast programme itself.

In a conditional access module implementation, a smart card may also form part of the system, this card being inserted in the conditional access module to decrypt the control word, which is then passed to the conditional access module to permit descrambling of the broadcast programme. If so, the first device may then correspond to a

conditional access module, the second device to a smart card and the data encrypted with the session key to control word data.

Within the field of digital television, the invention may also be applied to the communication of data between a decoder and other devices, such as a television or video recorder. In particular, in one embodiment, the first device corresponds to a first decoder and the second device to a second decoder.

In households possessing a first and second decoder, there are often a number of problems associated with maintaining communication between a first or "master" decoder and a second "slave" decoder. The use of a secure encrypted link to communicate audiovisual data, control word data, or even data relating to current subscription rights and exploitation keys, may prove useful in this context.

In yet a further realisation, the present invention may be applied to home network system, the first and second devices corresponding to first and second consumer electronic devices adapted to transfer data via a communication link (e.g. radio, PLC, infra-red etc.).

The above embodiments have been described in relation to a method of encryption of data. Viewed from another aspect, the invention may equally be applied to first and second devices adapted to carry out such a method.

Another aspect of the present invention provides a system for providing secure communication of data between first and second devices, said first device comprising a memory for storing at least one precalculated key pair comprising a session key and an encrypted version of the session key prepared using a transport key, and communication means, such as a communication link, for communicating the encrypted version of the session key to said second device, said second device comprising a memory for storing an equivalent transport key, decryption means, such as a processor, for decrypting said encrypted version of the session key using said equivalent transport key, and means, such as the processor, for encrypting data to be

communicated to said first device using said session key.

Features described above relating to method aspects of the present invention can also be applied to device or system aspects, and vice versa.

As used above, the terms "portable security module", "smart card" and "conditional access module" may be interpreted in their broadest sense as applying to any portable microprocessor and/or memory based card capable of carrying out the described functions.

As particular examples of such devices, a smart card may correspond to a card device constructed in accordance with the known international standards ISO 7816-1, 7816-2 and 7816-3 whilst the conditional access module may be implemented as a PCMCIA or PC card corresponding to the standards fixed by the PCMCIA group. Other physical shapes and forms are of course possible.

The terms "scrambled" and "encrypted" and "control word" and "key" have been used at various parts in the text for the purpose of clarity of language. However, it will be understood that no fundamental distinction is to be made between "scrambled data" and "encrypted data" or between a "control word" and a "key".

Similarly, unless obligatory in view of the context stated or unless otherwise specified, no limitation to either symmetric or public/private algorithms is to be inferred for a given encryption and/or decryption process. In the same way, whilst the matching keys used in encrypting and decrypting information may be referred to by the same name (e.g. "transport key", "session key") it is to be understood that these need not be numerically identical keys as long as they fulfil their functions. For example, the corresponding public and private keys used to encrypt and decrypt data will normally possess numerically different values.

The term "receiver/decoder" or "decoder" as used herein may connote a receiver for receiving either encoded or non-encoded signals, for example, television and/or radio

signals, which may be broadcast or transmitted by any appropriate means. Embodiments of such decoders may also include a decoder integral with the receiver for decoding the received signals, for example, in a "set-top box", a decoder functioning in combination with a physically separate receiver, or such a decoder including additional functions, such as a web browser, integrated with other devices such as a video recorder or a television.

As used herein, the term "digital transmission system" includes any transmission system for transmitting or broadcasting for example primarily audiovisual or multimedia digital data. Whilst the present invention is particularly applicable to a broadcast digital television system, the invention may also be applicable to a fixed telecommunications network for multimedia internet applications, to a closed circuit television, and so on.

As used herein, the term "digital television system" includes for example any satellite, terrestrial, cable and other system.

There will now be described, by way of example only, a number of embodiments of the invention, with reference to the following figures, in which:

Figure 1 shows by way of background the overall architecture of a digital TV system;

Figure 2 shows the architecture of the conditional access system of Figure 1;

Figure 3 shows a method of encryption of data between a smart card and a decoder according to this embodiment of the invention;

Figure 4 shows the generation of a session key in a decoder operating according to the embodiment of Figure 3; and

Figure 5 shows the steps in the preparation of a session key in a smart card interfacing with the decoder of Figure 4.

The present invention describes a method of encryption of data, in particular but not exclusively applicable to the encryption of data across the interface between a portable security module and decoder in a digital television system. By way of background, the architecture of a known digital television system will now be described.

Digital Television System

An overview of a digital television system 1 is shown in Figure 1 comprising a broadcast system 2 which uses the MPEG-2 compression system to transmit compressed digital signals. In more detail, an MPEG-2 compressor 3 in a broadcast centre receives a digital signal stream (for example a stream of audio or video signals). The compressor 3 is connected to a multiplexer and scrambler 4 by linkage 5. The multiplexer 4 receives a plurality of further input signals, assembles one or more transport streams and transmits compressed digital signals to a transmitter 6 of the broadcast centre via linkage 7, which can of course take a wide variety of forms including telecom links.

The transmitter 6 transmits electromagnetic signals via uplink 8 towards a satellite transponder 9, where they are electronically processed and broadcast via a notional downlink 10 to earth receiver 11, conventionally in the form of a dish owned or rented by the end user. The signals received by receiver 11 are transmitted to an integrated receiver/decoder 12 owned or rented by the end user and connected to the end user's television set 13. The receiver/decoder 12 decodes the compressed MPEG-2 signal into a television signal for the television set 13.

A conditional access system 20 is connected to the multiplexer 4 and the receiver/decoder 12, and is located partly in the broadcast centre and partly in the decoder. It enables the end user to access digital television broadcasts from one or more broadcast suppliers. A portable security module in the form of a smartcard capable of decrypting messages relating to broadcast programmes or data can be inserted into the receiver/decoder 12.

An interactive system 17, also connected to the multiplexer 4 and the receiver/decoder 12 and again located partly in the broadcast centre and partly in the decoder, may be provided to enable the end user to interact with various applications via a modemmed back channel 16.

The conditional access system 20 will now be described in more detail. With reference to Figure 2, in overview the conditional access system 20 includes a Subscriber Authorization System (SAS) 21. The SAS 21 is connected to one or more Subscriber Management Systems (SMS) 22, one SMS for each broadcast supplier, for example by a respective TCP-IP linkage 23 (although other types of linkage could alternatively be used). Alternatively, one SMS could be shared between two broadcast suppliers, or one supplier could use two SMSs, and so on.

First encrypting units in the form of ciphering units 24 utilising "mother" smartcards 25 are connected to the SAS by linkage 26. Second encrypting units again in the form of ciphering units 27 utilising mother smartcards 28 are connected to the multiplexer 4 by linkage 29. The receiver/decoder 12 receives a portable security module, for example in the form of "daughter" smartcard 30. It is connected directly to the SAS 21 by Communications Servers 31 via the modemmed back channel 16. The SAS sends, amongst other things, subscription rights to the daughter smartcard on request.

The smartcards contain the secrets of one or more commercial operators. The "mother" smartcard encrypts different kinds of messages and the "daughter" smartcards decrypt the messages, if they have the rights to do so.

The first and second ciphering units 24 and 27 comprise a rack, an electronic VME card with software stored on an EEPROM, up to 20 electronic cards and one smartcard 25 and 28 respectively, for each electronic card, one card 28 for encrypting the ECMs and one card 25 for encrypting the EMMs.

The operation of the conditional access system 20 of the digital television system will now be described in more detail with reference to the various components of the

television system 2 and the conditional access system 20.

Multiplexer and Scrambler

With reference to Figures 1 and 2, in the broadcast centre, the digital audio or video signal is first compressed (or bit rate reduced), using the MPEG-2 compressor 3. This compressed signal is then transmitted to the multiplexer and scrambler 4 via the linkage 5 in order to be multiplexed with other data, such as other compressed data.

The scrambler generates a control word used in the scrambling process and included in the MPEG-2 stream in the multiplexer. The control word is generated internally and enables the end user's integrated receiver/decoder 12 to descramble the programme.

Access criteria, indicating how the programme is commercialised, are also added to the MPEG-2 stream. The programme may be commercialised in either one of a number of "subscription" modes and/or one of a number of "Pay Per View" (PPV) modes or events. In the subscription mode, the end user subscribes to one or more commercial offers, or "bouquets", thus getting the rights to watch every channel inside those bouquets. In the preferred embodiment, up to 960 commercial offers may be selected from a bouquet of channels.

In the Pay Per View mode, the end user is provided with the capability to purchase events as he wishes. This can be achieved by either pre-booking the event in advance ("pre-book mode"), or by purchasing the event as soon as it is broadcast ("impulse mode"). In the preferred embodiment, all users are subscribers, whether or not they watch in subscription or PPV mode, but of course PPV viewers need not necessarily be subscribers.

Entitlement Control Messages

Both the control word and the access criteria are used to build an Entitlement Control

Message (ECM). This is a message sent in relation with a scrambled program; the message contains a control word (which allows for the descrambling of the program) and the access criteria of the broadcast program. The access criteria and control word are transmitted to the second encrypting unit 27 via the linkage 29. In this unit, an ECM is generated, encrypted and transmitted on to the multiplexer and scrambler 4. During a broadcast transmission, the control word typically changes every few seconds, and so ECMs are also periodically transmitted to enable the changing control word to be descrambled. For redundancy purposes, each ECM typically includes two control words; the present control word and the next control word.

Each service broadcast by a broadcast supplier in a data stream comprises a number of distinct components; for example a television programme includes a video component, an audio component, a sub-title component and so on. Each of these components of a service is individually scrambled and encrypted for subsequent broadcast to the transponder 9. In respect of each scrambled component of the service, a separate ECM is required. Alternatively, a single ECM may be required for all of the scrambled components of a service. Multiple ECMs are also generated in the case where multiple conditional access systems control access to the same transmitted program.

Entitlement Management Messages (EMMs)

The EMM is a message dedicated to an individual end user (subscriber), or a group of end users. Each group may contain a given number of end users. This organisation as a group aims at optimising the bandwidth; that is, access to one group can permit the reaching of a great number of end users.

Various specific types of EMM can be used. Individual EMMs are dedicated to individual subscribers, and are typically used in the provision of Pay Per View services; these contain the group identifier and the position of the subscriber in that group.

Group subscription EMMs are dedicated to groups of, say, 256 individual users, and are typically used in the administration of some subscription services. This EMM has a group identifier and a subscribers' group bitmap.

Audience EMMs are dedicated to entire audiences, and might for example be used by a particular operator to provide certain free services. An "audience" is the totality of subscribers having smartcards which bear the same conditional access system identifier (CA ID). Finally, a "unique" EMM is addressed to the unique identifier of the smartcard.

EMMs may be generated by the various operators to control access to rights associated with the programs transmitted by the operators as outlined above. EMMs may also be generated by the conditional access system manager to configure aspects of the conditional access system in general.

The term EMM is also often used to describe specific configuration type messages communicated between the decoder and other elements of the system and, for example, will be used later in this application to refer to a specific message passed from the decoder to a smart card.

Subscriber Management System (SMS)

A Subscriber Management System (SMS) 22 includes a database 32 which manages, amongst others, all of the end user files, commercial offers, subscriptions, PPV details, and data regarding end user consumption and authorization. The SMS may be physically remote from the SAS.

Each SMS 22 transmits messages to the SAS 21 via respective linkage 23 which imply modifications to or creations of Entitlement Management Messages (EMMs) to be transmitted to end users.

The SMS 22 also transmits messages to the SAS 21 which imply no modifications or

creations of EMMs but imply only a change in an end user's state (relating to the authorization granted to the end user when ordering products or to the amount that the end user will be charged).

The SAS 21 sends messages (typically requesting information such as call-back information or billing information) to the SMS 22, so that it will be apparent that communication between the two is two-way.

Subscriber Authorization System (SAS)

The messages generated by the SMS 22 are passed via linkage 23 to the Subscriber Authorization System (SAS) 21, which in turn generates messages acknowledging receipt of the messages generated by the SMS 21 and passes these acknowledgements to the SMS 22.

In overview the SAS comprises a Subscription Chain area to give rights for subscription mode and to renew the rights automatically each month, a Pay Per View Chain area to give rights for PPV events, and an EMM Injector for passing EMMs created by the Subscription and PPV chain areas to the multiplexer and scrambler 4, and hence to feed the MPEG stream with EMMs. If other rights are to be granted, such as Pay Per File (PPF) rights in the case of downloading computer software to a user's Personal Computer, other similar areas are also provided.

One function of the SAS 21 is to manage the access rights to television programmes, available as commercial offers in subscription mode or sold as PPV events according to different modes of commercialisation (pre-book mode, impulse mode). The SAS 21, according to those rights and to information received from the SMS 22, generates EMMs for the subscriber.

The EMMs are passed to the Ciphering Unit (CU) 24 for ciphering with respect to the management and exploitation keys. The CU completes the signature on the EMM and passes the EMM back to a Message Generator (MG) in the SAS 21, where a header

is added. The EMMs are passed to a Message Emitter (ME) as complete EMMs. The Message Generator determines the broadcast start and stop time and the rate of emission of the EMMs, and passes these as appropriate directions along with the EMMs to the Message Emitter. The MG only generates a given EMM once; it is the ME which performs cyclic transmission of the EMMs.

On generation of an EMM, the MG assigns a unique identifier to the EMM. When the MG passes the EMM to the ME, it also passes the EMM ID. This enables identification of a particular EMM at both the MG and the ME.

Programme Transmission

The multiplexer 4 receives electrical signals comprising encrypted EMMs from the SAS 21, encrypted ECMs from the second encrypting unit 27 and compressed programmes from the compressor 3. The multiplexer 4 scrambles the programmes and sends the scrambled programmes, the encrypted EMMs and the encrypted ECMs to a transmitter 6 of the broadcast centre via the linkage 7. The transmitter 6 transmits electromagnetic signals towards the satellite transponder 9 via uplink 8.

Programme Reception

The satellite transponder 9 receives and processes the electromagnetic signals transmitted by the transmitter 6 and transmits the signals on to the earth receiver 11, conventionally in the form of a dish owned or rented by the end user, via downlink 10. The signals received by receiver 11 are transmitted to the integrated receiver/decoder 12 owned or rented by the end user and connected to the end user's television set 13. The receiver/decoder 12 demultiplexes the signals to obtain scrambled programmes with encrypted EMMs and encrypted ECMs.

If the programme is not scrambled, that is, no ECM has been transmitted with the MPEG-2 stream, the receiver/decoder 12 decompresses the data and transforms the signal into a video signal for transmission to television set 13.

If the programme is scrambled, the receiver/decoder 12 extracts the corresponding ECM from the MPEG-2 stream and passes the ECM to the "daughter" smartcard 30 of the end user. This slots into a housing in the receiver/decoder 12. The daughter smartcard 30 controls whether the end user has the right to decrypt the ECM and to access the programme. If the end user does have the rights, the ECM is decrypted within the smart card and the control word extracted.

Thereafter the smart card then communicates the control word to the decoder 12 which then descrambles the programme using this control word. In most conventional systems, the control word is communicated across the smart card interface in a clear or non-encrypted form, leading to the problems of security described in the introduction of the present application. After descrambling by the decoder, the MPEG-2 stream is decompressed and translated into a video signal for onward transmission to television set 13.

In the system described above, the descrambling of the MPEG data is carried out within the decoder using the control word information communicated to the decoder from the smart card. In other systems, the descrambling circuitry may be implemented in a detachable conditional access module or CAM, commonly embodied in the form of a PCMCIA or PC card insertable in a socket in the decoder.

The CAM module may itself further include a slot to receive a smart card. In such systems, control word data is decrypted in the smart card communicated to the CAM module which then descrambles the scrambled MPEG data stream to supply the decoder with a clear MPEG stream for decompression and subsequent display.

In this type of system, sensitive data may be passed between the smart card and CAM (control word data) and/or between the CAM and decoder (descrambled MPEG data) and problems of security may arise at either of these interfaces.

Data Encryption across an Interface

Referring to Figure 3, there will now be described a method of data encryption as applied to the control word data communicated between a smart card and a decoder in one of the simplest embodiments of this invention. However, the same principles may be applied to the encryption of control word data between a smart card and a CAM, audiovisual MPEG data between a CAM and a decoder, or indeed any type of data between two such devices.

In accordance with the present invention, a set of key pairs is stored in a non-volatile memory of the decoder e.g. a FLASH memory. Each key pair corresponds to a key value in clear form and an encrypted version of the key. As will be described, the encrypted version of the key will be eventually communicated in an EMM message sent to a smart card inserted in the decoder.

Thus, within the decoder a set of EMM message/key pairs are stored as follows:

n	EMM (19 octets)	Key (8 octets)
1	EMM(1)	Key(1)
2	EMM(2)	Key(2)
3	EMM(3)	Key(3)
.	.	.
.	.	.
.	.	.
.	.	
16	EMM(16)	Key(16)

The encrypted value of the key stored in the EMM is calculated external of the decoder using an encryption algorithm not present in the decoder. In the present example the key values Key(1), Key(2) etc. correspond to symmetric keys to be used with a symmetric encryption algorithm such as DES.

The encryption algorithm used to prepare the encrypted DES key values contained with the stored EMM messages may also correspond to a symmetric encryption algorithm. For increased security, a proprietary symmetric algorithm (PSA) different from DES will be used to prepare the encrypted values, although in another

embodiment DES may also be used to encrypt the key values.

In addition to the encrypted value of the associated key, the EMM message may also include a signature value associated with the message and prepared as per any conventional signature preparation method. For example, a message may be subject to a hash function such as MD5 followed by encryption of the hash value by a private key of private/public key algorithm such as RSA. Verification of the signature may then be carried out at the point of reception using a MD5 algorithm and the corresponding public key of the private/public key pair.

The EMM message will additionally include a standard smart card header element (as defined by the international standard ISO 7816-3) to place the message in a format necessary to permit it to be read by a smart card. An EMM associated with an 8 byte key will therefore typically have the following structure:

Header	5 bytes
Encrypted key	10 bytes
Signature	9 bytes

In the present embodiment a set of 16 key/message pairs are implanted in the memory of the decoder. Alternative embodiments are equally possible using more or less key/message pairs and the invention may even be implemented using a single key/message pair. Whilst it may be envisaged that all decoders are equipped with the same key/message pairs it is preferred for security reasons that each decoder has a unique set of key/message pairs. In implementing this embodiment, an operator may supply to a decoder manufacturer a set of ten thousand or more key/message pairs, the decoder manufacturer taking a random selection of 16 pairs during the personalisation of each decoder.

In order to increase the security, a different subset of the message/key pairs stored in the decoder will be used during each session. A session may be defined as corresponding to each time the decoder is switched on and off, or each time the

decoder changes channel, for example.

Referring to Figure 3, a random number generator 40 within the decoder selects 8 out of the 16 message/key pairs to be used in that session. The 8 selected EMM messages 41 of the pairs are then communicated to the smart card 30 to be verified and decrypted and processed as shown at 42 and 43 to obtain the appropriate session key (see below). The same key generation operation is carried out within the decoder at 43 using the corresponding key values of the pairs so as to obtain the same session key value.

The generation of the session key within the decoder will now be described with reference to Figure 4.

A base session key value KeyS Initial shown at 44 and constant for all decoders is encrypted at 45 by the first key 46 of the subset chosen by the random generator 40. The resulting value is then encrypted at 47 using the second key 48 of the session subset and the operation repeated just until the last encryption operation 49 carried out with the last key 50 of the subset so as to obtain the final session key value shown at 51.

The initial session key value KeyS Initial can be a universal value present in all decoders and smart cards, a value linked to a specific decoder/smart card pair or even a value generated at the start of each session in the decoder and thereafter communicated to the smart card.

In the example given above, the session key is prepared by a sequence of repeated operations on the KeyS Initial using the DES algorithm and the selected keys 46, 48, 50 etc. In the case of the DES algorithm, the order in which the keys are applied is important and must be respected to produce the same key each time.

However, whilst the session key S is itself a numerical value that will be used as a DES key in the subsequent decryption operation (see below), the steps used to

generate this key value need not correspond to DES encryption steps. Instead, the subset of keys chosen by the random number generator may be combined together in any number of ways to arise at a suitable session key value KeyS Final. For example, the keys may be combined using a sequence of simple arithmetic operations. Depending on the method chosen, it may not be necessary that the order of the steps in the preparation of the KeyS be respected in order to regenerate the same key.

Referring now to Figure 5, the decryption and processing operations 42 and 43 carried out in the smart card 30 to generate the session key used by the smart card will now be described.

Upon insertion of the smart card in the decoder, the subset of EMM messages matching the selected key values are sent to the smart card. Authentification of each EMM messages is first carried out with reference to the attached signature value, using for example an MD5/RSA type process as described above. For simplicity, this step has been omitted from Figure 5.

The first EMM message 60 is then decrypted at 61 using a transport key 59 embedded in a secure and non-readable manner within the smart card. As mentioned above, for security reasons the algorithm used in the decryption 61 of the EMM message may correspond to a proprietary security algorithm PSA known only to the operator responsible for preparation of the message/key pairs used in the decoder and the personalisation of the smart card.

The transport key KeyT shown at 59 may be a key value common to all smart cards in the system or unique to one such card. The use of a unique key value KeyT requires that the message/key table stored in the decoder be prepared with the same key as that in the card, such that a decoder and card will be irreversibly linked together. In practice, this may not be desirable.

A similar decryption operation using the transport key 59 is then carried out at 62 on the next EMM message 63 in the series and 50 on until the last decryption operation

64 on the final EMM message 65.

In the present embodiment, encryption of each of the EMM messages 60, 63, 65 produces keys 46, 48, 50 identical to those associated in the message/key table present in the decoder and used for generation of the session key as described previously. For this reason, the same reference numbers have been used for these keys and for the key generation operation 43 also carried out in the decoder. Similarly, the same initial session key 44 present in the decoder is also stored in the smart card.

The initial session key KeyS Initial shown at 44 is then encrypted at 45 by the first key 46, the result re-encrypted at 47 by the second key 48 and so on until the final encryption step carried out at 49 using the last key 50 in the series so as to obtain the final session key at 51.

Both the decoder and smart card now possess the same session key KeyS which may thereafter be used in encrypting and decrypting data passed in either direction between the two devices.

Referring back to Figure 3, the smart card 30 receives an encrypted ECM message containing the control word necessary for descrambling an associated segment of MPEG audiovisual or other data. The smart card decrypts the ECM at 71 to obtain the control word value CW.

In passing, we note that the algorithm used to encrypt ECM messages for a user may conveniently correspond to the Proprietary Security Algorithm used for decryption of the EMM messages received from the smart card as described above.

The decrypted control word is then re-encrypted at 72 using the session key KeyS and the encrypted control word value f(CW) transmitted over the decoder/smart card interface as shown. The encrypted value f(CW) is then decrypted at 73 using the session key KeyS held in the decoder and the clear value of the control word CW obtained at 74.

As the session key is symmetric, it may equally be used in the encryption of data transmitted from the decoder to the smart card. Furthermore, the data transmitted from the smart card to the decoder may be data other than simple control word data.

As mentioned above, the same principle may be applied across all interfaces in a system comprising a decoder in which a detachable CAM module is inserted (decoder/CAM interface, CAM/smart card interface etc.). Similarly, the same principle may be applied in the case of a portable module (either a CAM type module or a smart card) inserted in other devices such as a television or video recorder.

In fact, the above method of setting up an encrypted communication channel may be applied to any pair of devices where security of data communication is required. In particular, the same principle may be applied in a home network system where multiple consumer devices (television, video, PC, decoder etc.) transfer data such as audiovisual data or computer files via a communication link. This may be an RF link, an infrared link, a dedicated bus, a power line connection etc. For example, it may be desired to transmit control word in other data in an encrypted form between a decoder and a television or between a master decoder and a slave decoder in the same household.

Other examples of systems of this type where a secure communication link would be desirable will also be apparent to the reader.

CLAIMS

1. A method of encryption of data communicated between a first and second device, wherein at least one precalculated key pair is stored in a memory of the first device, said at least one key pair comprising a session key and an encrypted version of the session key prepared using a transport key, the encrypted version of the session key being subsequently communicated to the second device which decrypts the encrypted version using an equivalent transport key stored in its memory such that data communicated from at least the second to the first device may thereafter be encrypted and decrypted by the session key in the respective devices.

2. A method as claimed in claim 1, in which a plurality of key pairs are stored in the memory of the first device, the first device selecting and processing at least one session key to generate a definitive session key and communicating the associated encrypted version of said at least one session key to the second device for decryption and processing by the second device to generate the definitive session key.

3. A method as claimed in claim 2 in which a subset of a plurality of stored session keys is chosen by the first device to generate the definitive session key, the associated encrypted versions of the subset of session keys being communicated to the second device for decryption and processing.

4. A method as claimed in claim 2 or 3, in which the order of combination of a plurality of session keys used to generate the definitive session key is communicated from the first to the second device.

5. A method as claimed in claim 4 in which an initial session key value known to both the first and second devices is repeatedly encrypted in both devices by an ordered sequence of session keys using an encryption algorithm sensitive to the order of encryption.

6. A method as claimed in any preceding claim in which said at least one

precalculated key pair is selected from a larger set of precalculated key pairs prior to being stored in the first device.

7. A method as claimed in any preceding claim in which the encrypted version of a session key communicated to the second device also includes a signature value readable by the second device to verify the authenticity of the encrypted version of the session key.

8. A method as claimed in any preceding claim in which an algorithm and transport key used to encrypt and decrypt a session key correspond to a symmetric algorithm and associated symmetric key.

9. A method as claimed in any preceding claim in which an encryption algorithm used with a session key to encrypt and decrypt data communicated between the first and second device corresponds to a symmetric algorithm.

10. A method as claimed in any preceding claim, in which the first device is a decoder.

11. A method as claimed in any preceding claim, in which the second device is a portable security module.

12. A method as claimed in claim 11, in which the portable security module corresponds to one of a smart card and a conditional access module.

13. A method as claimed in any of claims 1 to 9, in which the first device corresponds to a conditional access module and the second device corresponds to a smart card.

14. A method as claimed in any of claims 10 to 13, in which data encrypted and decrypted with a session key corresponds to control word data.

15. A method as claimed in any of claims 10 to 13, in which data encrypted and decrypted with a session key corresponds to descrambled broadcast data.

16. A method as claimed in any of claims 1 to 9 in which the first and second device correspond to a first and second decoder respectively.

17. A method as claimed in any of claims 1 to 9 as applied to a home network system, the first and second devices corresponding to first and second consumer electronic devices adapted to transfer data via a communication link.

18. A first device adapted to be used in a method as claimed in any of claims 1 to 17, the first device including a memory in which at least one precalculated key pair is stored, said at least one precalculated key pair comprising a session key and an encrypted version of this session key.

19. A second device adapted to be used in a method as claimed in any of claims 1 to 18 and with a first device as claimed in claim 18, the second device comprising a memory in which is stored a key and algorithm that are needed to decrypt the encrypted session key value stored in the memory of the first device.

20. A first and second device as claimed in claims 18 and 19, in which the first device corresponds to a decoder and the second device to a portable security module.

21. A system for providing secure communication of data between first and second devices, said first device comprising a memory for storing at least one precalculated key pair comprising a session key and an encrypted version of the session key prepared using a transport key, and communication means for communicating the encrypted version of the session key to said second device, said second device comprising a memory for storing an equivalent transport key, decryption means for decrypting said encrypted version of the session key using said equivalent transport key, and means for encrypting data to be communicated to said first device using said session key.

22. A system as claimed in claim 21, wherein the memory of the first device is adapted to store a plurality of key pairs, the first device comprising means for selecting and processing at least one session key to generate a definitive session key said communication means being adapted to communicate the associated encrypted version of said at least one session key to the second device, said second device comprising means for processing said at least one session key to generate the definitive session key.

23. A system as claimed in claim 21 or 22, in which the encrypted version of a session key includes a signature value readable by the second device to verify the authenticity of the encrypted version of the session key.

24. A system as claimed in any of claims 21 to 23, in which an algorithm and transport key used to encrypt and decrypt a session key correspond to a symmetric algorithm and associated symmetric key.

25. A system as claimed in any of claims 21 to 24, in which an encryption algorithm used with a session key to encrypt and decrypt data communicated between the first and second device corresponds to a symmetric algorithm.

26. A system as claimed in any of claims 21 to 25, in which the first device is a decoder.

27. A system as claimed in any of claims 21 to 26, in which the second device is a portable security module.

28. A system as claimed in claim 27, in which the portable security module corresponds to one of a smart card and a conditional access module.

29. A system as claimed in any of claims 21 to 25, in which the first device corresponds to a conditional access module and the second device corresponds to a

smart card.

30. A system as claimed in any of claims 21 to 25 in which the first and second device correspond to a first and second decoder respectively.

31. A system as claimed in any of claims 21 to 25 as applied to a home network system, the first and second devices corresponding to first and second consumer electronic devices adapted to transfer data via a communication link.

32. A method of encryption of data communicated between a first and second device substantially as herein described.

33. A system for providing secure communication of data between first and second devices substantially as herein described.

FIG. 1

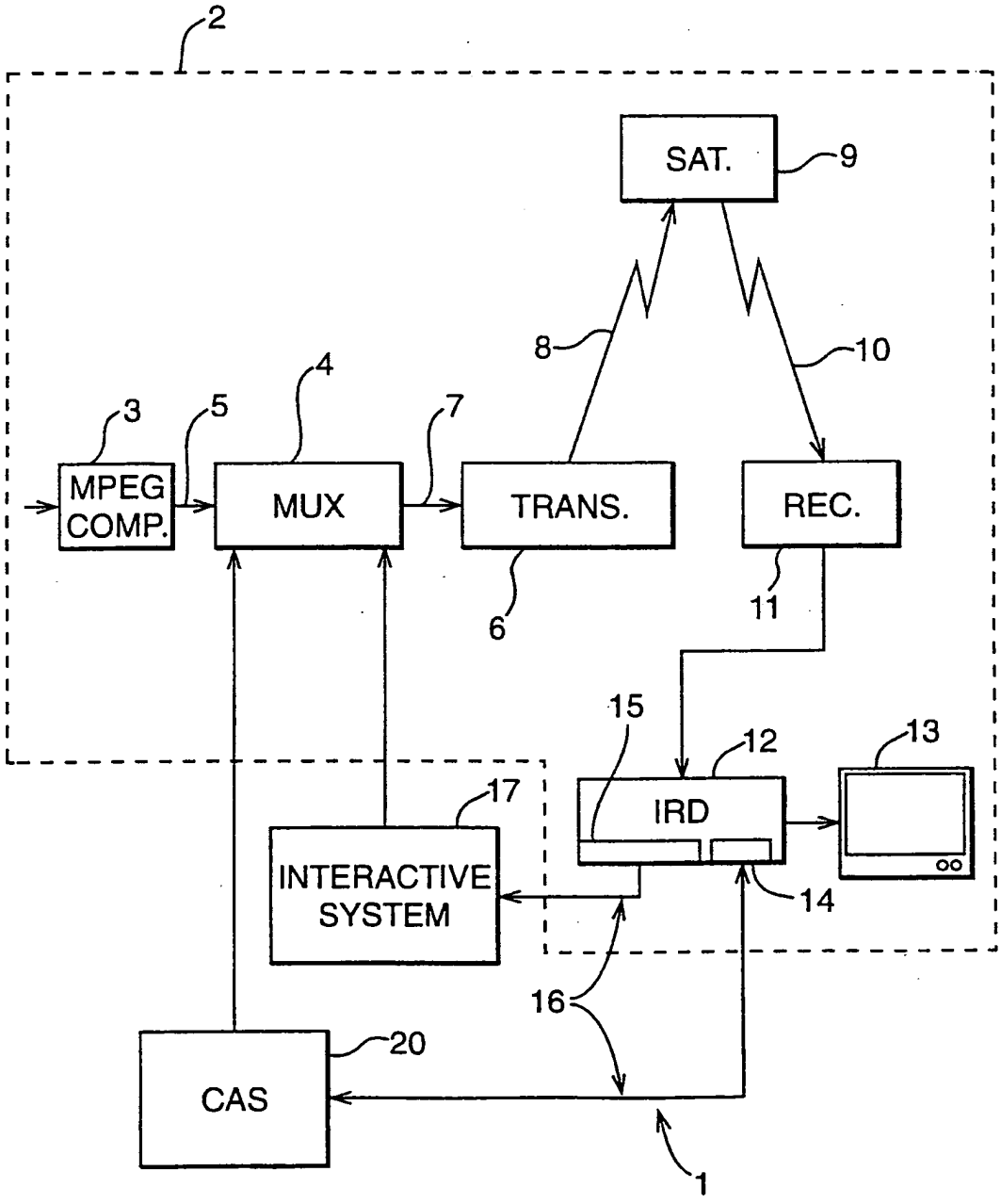

2
SAT.
9
8
10
4
3
5
7
MPEG COMP.
MUX
TRANS.
REC.
6
11
15
12
13
17
IRD
INTERACTIVE SYSTEM
14
16
20
CAS
1

FIG. 2

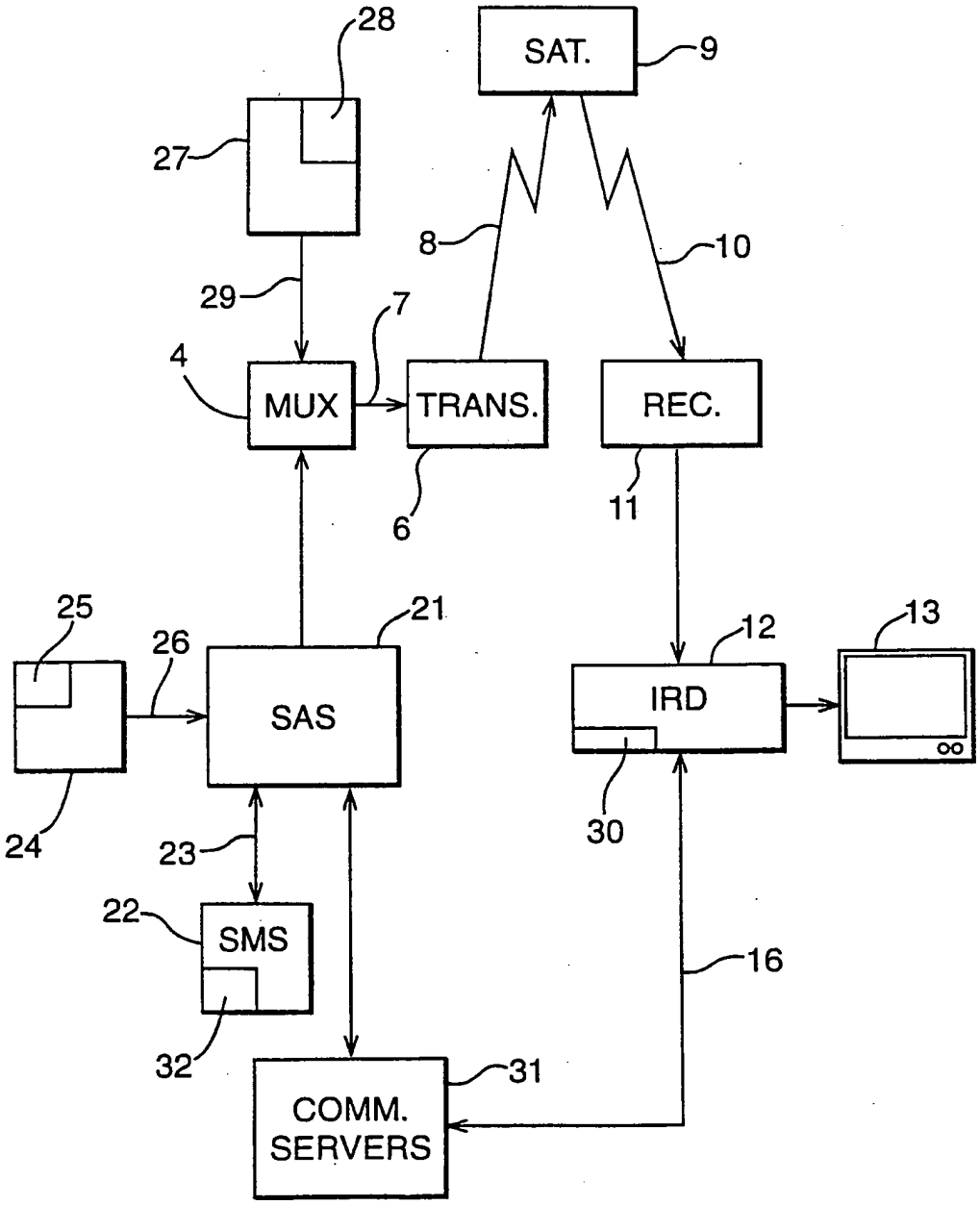
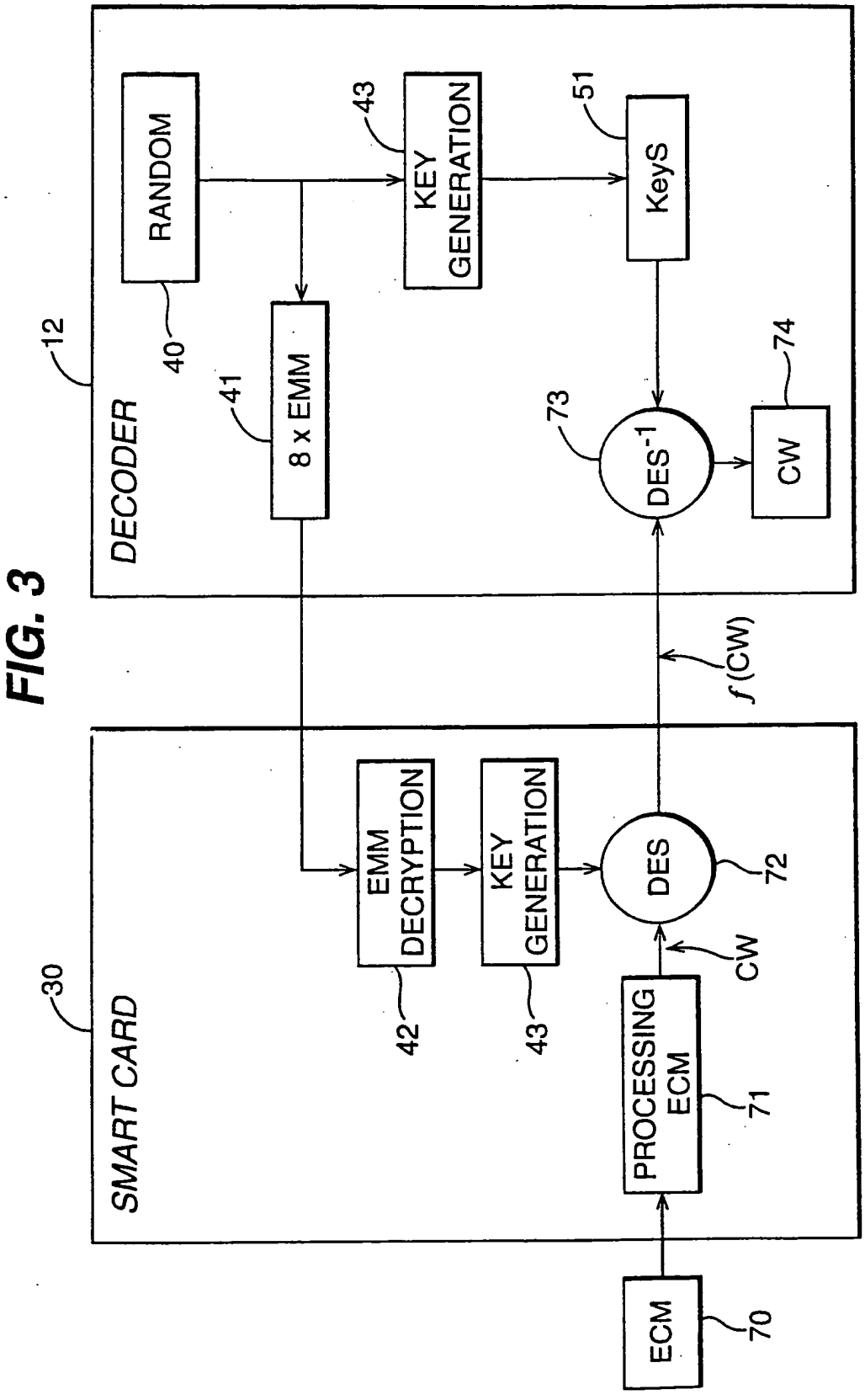

28
SAT.
9
27
8
10
29
7
4
MUX
TRANS.
REC.
6
11
25
26
21
12
13
SAS
IRD
24
23
30
22
SMS
16
32
31
COMM.
SERVERS

FIG. 3


DECODER
12
RANDOM
40
8 x EMM
41
KEY GENERATION
43
KeyS
51
DES-1
73
CW
74
f(CW)
SMART CARD
30
EMM DECRYPTION
42
KEY GENERATION
43
DES
72
CW
PROCESSING ECM
71
ECM
70

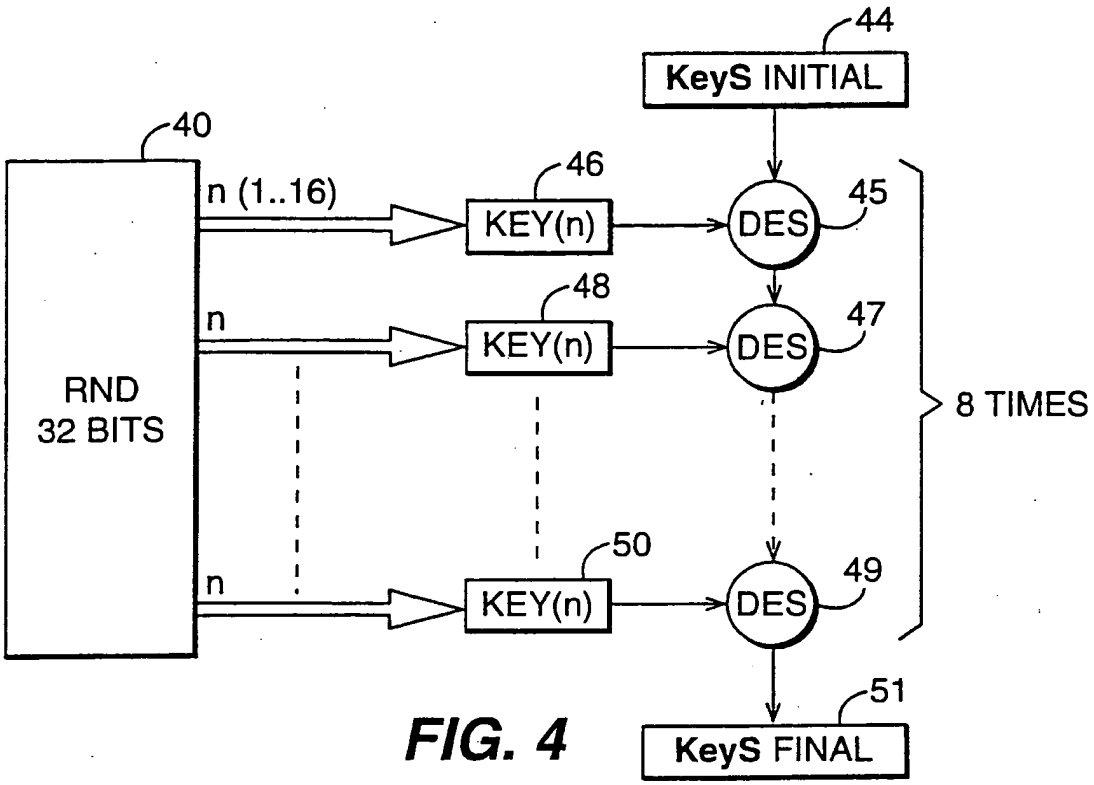

44
KeyS INITIAL
40
n (1..16)
46
KEY(n)
DES
45
48
n
KEY(n)
DES
47
RND
32 BITS
8 TIMES
50
n
KEY(n)
DES
49
51
KeyS FINAL

FIG. 4

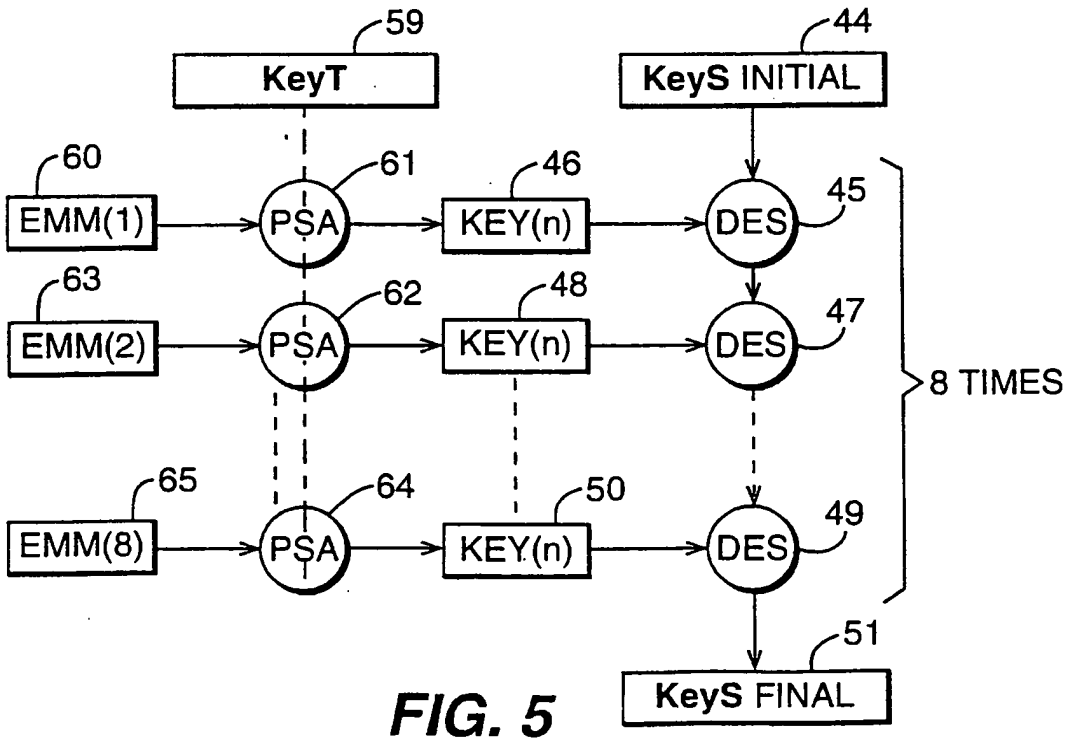

59
KeyT
44
KeyS INITIAL
60
61
46
EMM(1)
PSA
KEY(n)
DES
45
63
62
48
EMM(2)
PSA
KEY(n)
DES
47
8 TIMES
65
64
50
EMM(8)
PSA
KEY(n)
DES
49
51
KeyS FINAL

FIG. 5

INTERNATIONAL SEARCH REPORT

International Application No
PCT/IB 00/00163

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04N7/16 H04N7/167

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 817 485 A (THOMSON MULTIMEDIA SA) 7 January 1998 (1998-01-07) page 3, column 3, line 54 -page 5, column 8, line 11 figures 1-5	1,2,4, 10-15, 17, 19-22, 26-29
X	EP 0 723 371 A (THOMSON MULTIMEDIA SA) 24 July 1996 (1996-07-24) page 3, column 3, line 57 -page 5, column 7, line 8 figures 1-4	1,2,4, 10-15, 19-22, 26-29

-/--

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

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 May 2000	07/06/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Van der Zaal, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int tional Application No
PCT/IB 00/00163

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	EBU PROJECT GROUP B/CA: "FUNCTIONAL MODEL OF A CONDITIONAL ACCESS SYSTEM" EBU REVIEW- TECHNICAL, no. 266, 21 December 1995 (1995-12-21), pages 64-77, XP000559450 Grand Saconnex, CH page 64, left-hand column, line 1 -page 72, right-hand column, line 29 figures 1-8	1-33

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int ional Application No
PCT/IB 00/00163

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0817485 A	07-01-1998	FR 2750554 A	02-01-1998
		CN 1171015 A	21-01-1998
		JP 10164052 A	19-06-1998
		US 6035038 A	07-03-2000
EP 0723371 A	24-07-1996	FR 2729521 A	19-07-1996
		JP 8307850 A	22-11-1996

Form PCT/ISA/210 (patent family annex) (July 1992)

(WO/2000/062260) METHOD AND SYSTEM FOR ORDERING, LOADING AND USING ACCESS TICKETS

Biblio. Data | Description | Claims | National Phase | Notices | Documents

Latest bibliographic data on file with the International Bureau

Publication Number: WO/2000/062260
Publication Date: 19.10.2000
Chapter 2 Demand Filed: 22.04.2000
International Application No.: PCT/CH1999/000142
International Filing Date: 07.04.1999

Int. Class.: ***G06Q 20/00*** (2006.01), ***G07B 15/00*** (2006.01), ***G07F 17/42*** (2006.01), ***G07F 7/00*** (2006.01), ***G07F 7/08*** (2006.01)

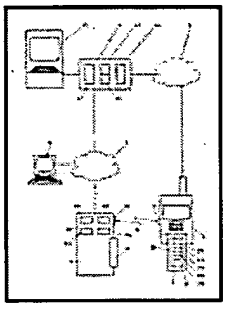
Applicants: **SWISSCOM MOBILE AG** [CH/CH]; Schwarztorstrasse 61 CH-3050 Bern (CH) *(All Except US)*.
RITTER, Rudolf [CH/CH]; Rossweidweg 8 CH-3052 Zollikofen (CH) *(US Only)*.
LAUPER, Eric [CH/CH]; Hochfeldstrasse 96 CH-3012 Bern (CH) *(US Only)*.

Inventors: **RITTER, Rudolf** [CH/CH]; Rossweidweg 8 CH-3052 Zollikofen (CH).
LAUPER, Eric [CH/CH]; Hochfeldstrasse 96 CH-3012 Bern (CH).

Agent: **BOVARD AG**; Optingenstrasse 16 CH-3000 Bern 25 (CH).

Title: METHOD AND SYSTEM FOR ORDERING, LOADING AND USING ACCESS TICKETS

Abstract: The invention relates to a method and a system for ordering, loading and using access tickets for the access to access-controlled service devices (3). Access tickets are ordered by a reservation centre (4) in said service device (3) by transmitting order information via an order channel. The order information comprises the telephone number of a mobile communications terminal (1). The ordered access tickets are transmitted to said terminal (1) via a mobile network (6) and are stored in a storage module (21) of the communications terminal (1). Data is exchanged between the storage module (21) and a reading device (31) of a service device (3) via a contactless interface (13). Decisions on the access permission for the user of said communications terminal (1) are made, e.g. in the reading device (31) or in the communications terminal (1), considering ticket information contained in said access ticket. Said information can be limited to a digitally signed ticket number or can contain data on the relevant service device. Access for the user to the service device (3) is given or denied according to the decision and by means of an access device (32) that is connected to the reading device.



Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.
African Regional Intellectual Property Org. (ARIPO) (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW)
Eurasian Patent Organization (EAPO) (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM)
European Patent Office (EPO) (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE)
African Intellectual Property Organization (OAPI) (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Publication Language: German (DE)

Filing Language: German (DE)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 January 2001 (11.01.2001)

PCT

(10) International Publication Number
WO 01/03044 A1

(51) International Patent Classification[7]: G06F 17/60

(21) International Application Number: PCT/US00/18510

(22) International Filing Date: 6 July 2000 (06.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/347,391 6 July 1999 (06.07.1999) US

(71) Applicant *(for all designated States except US)*: TRANSCAST INTERNATIONAL, INC. [US/US]; Regency Plaza, 2350 Mission College Blvd., Suite 190, Santa Clara, CA 95054 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: NARAYAN, Kris [US/US]; 983 Sandalridge Court, Milpitas, CA 95035 (US).

(74) Agent: THAPPETA, Narendra, Reddy; Law Firm of Naren Thappeta, 39899 Balentine Drive #119, Newark, CA 94560 (US).

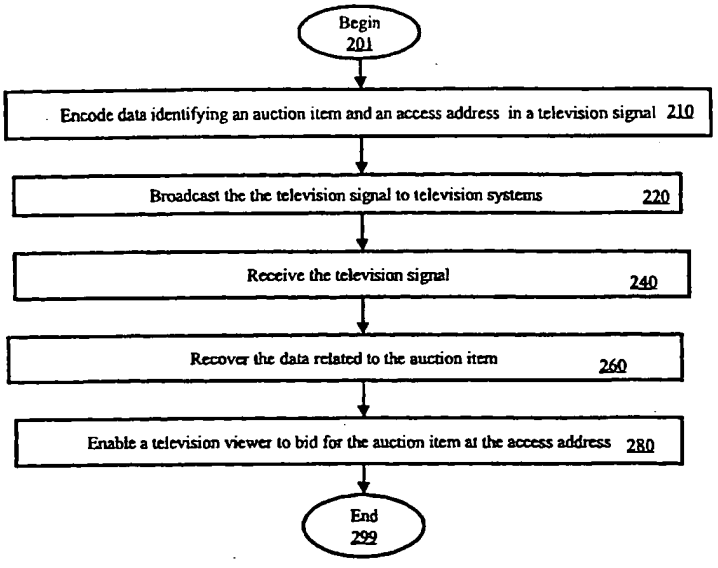
(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: ENABLING VIEWERS OF TELEVISION SYSTEMS TO PARTICIPATE IN AUCTIONS



(57) Abstract: Enabling the viewers of television systems to participate in auctions. Data identifying an item (e.g., description of the auction item and a unique code) offered for sale in an auction and an access address (e.g., universal resource locator of a web site) may be encoded (210) in a television signal and broadcast (220) to various television systems. The data may be recovered (240, 260) by a transaction enabler which enables a viewer to bid for the auction item (280). Other information such as highest present bid price may also be encoded in the television signal and displayed for the viewer.

WO 01/03044 A1

ENABLING VIEWERS OF TELEVISION SYSTEMS TO PARTICIPATE IN AUCTIONS

Related Application

The present invention is related to co-pending U.S. Patent Application Entitled, "Encoding Hot Spots in Television Signals", Serial Number: 09/276,266, Filing Date: March 25, 1999, which is incorporated in its entirety into the present application.

Background of the Invention

Field of the Invention

The present invention relates to television systems, and more specifically to a method and apparatus for using television signals to enable viewers of television systems to participate in auctions.

Related Art

An auction generally refers to a process in which multiple parties are provided the opportunity to bid for an offered item. The offered item can be a process or a service. In a typical bidding process, an a seller offers an item, and a party ("bidder") bids for the offered item usually by specifying a price the party is willing to pay. The seller may specify the minimum acceptable price and a time at which the auction closes.

Typically, an offered item is sold to the highest bidder (i.e., party specifying highest price) in return for the specified highest price. However, criteria other than price (e.g., credit worthiness) of the bidder may also be taken into consideration in determining the bidder to whom to sell an offered item.

Central servers are known in the relevant arts which coordinate the bidding process. For example, web site at URL of http://www.ebay.com enable sellers to offer products

according to various categories (e.g., sports memorabilia, computers), and a bidder may bid on the offered products by using a browser on the world-wide web as is well known in the relevant arts.

Organizations such as those providing the web sites to enable auctions are hereafter referred to as "service providers". Service providers often advertize on various other web sites so that users accessing ("surfing") these web sites may know about the general service. Typically, a user (viewer of the advertisement) can click on an advertisement to access the web sites providing the auction service.

However, these advertisements are typically targeted to the users surfing the world wide web, and may not target at least some of the viewers ("television viewers") of television systems. The television viewers constitutes a big segment of the auction market, and it is therefore desirable to enable television viewers to participate in the auctions.

Such participation may be particularly important as the viewers of a specific television program may be expected to be of certain 'profile', and certain items may be suitable for people of that profile. For example, a person watching Mr. Mark McGuire (a baseball player in United States baseball) hit a record breaking home run may be interested in purchasing a baseball bat signed personally by Mr. Mark McGuire. That is, the auction items can be targeted to the viewers of television programs.

At least for the above-stated reasons, what is needed is a method and apparatus for enabling viewers of television systems to participate in auctions.

Summary of the Invention

The present invention enables viewers ("television viewers") of television systems to participate in auctions. The auctions may be occurring on web sites on the Internet also. In

an embodiment of the invention, data describing an item ("auction item") available for bidding and an access address of a system at which a television viewer may bid are encoded in a television signal.

The user may submit a bid at a system (e.g., a web site) identified by the access address. In case the system is a web server, users ('surfers') of world-wide-web may also submit bids by accessing the web server on the world-wide web. Accordingly, the present invention may be used to draw television viewers to web-sites (e.g., www.ebay.com) dedicated to auctions also.

In an embodiment, the data is encoded in the non-display portion (e.g., vertical blanking interval) of the television signal. However, other portions of a television signal may also be used for encoding the data. Other information of interest to the viewer such as a minimum bid amount specified by a seller and the present maximum may also be encoded in the television signal, and displayed for viewer convenience.

A transaction enabler may recover the data encoded in the television signals, and display the information to the viewer. The viewer may conveniently bid on the auction items, for example, by specifying the bid price (offer) and clicking on a pre-specified portion of a displayed image.

The bid may be automatically sent to a server identified by the access address. In the alternative, the viewer may be first navigated to a web server specified by the access address, and the user may specify the bid price then. A unique code identifying the auction item may also be encoded in the television signal, and the code may be used to identify that the bid price relates to the auction item. In the alternative, the URL itself may contain such identification codes.

The transaction server may also provide updated information on a present highest bid. For example, an end time associated with the auction may be provided to the television viewer, and the viewer may check the present highest bid at a later time before the end time, and then decide whether to submit a bid. In addition, the transaction server may interact with the system providing the auction service, and provide periodic updates at viewer's option. As a result, a viewer may make an informed decision on whether to bid.

Therefore, the present invention enables a television viewer to participate in an auction by encoding in a television signal the data identifying an auction item and an access address.

The present invention enables television viewers to be drawn to web sites providing auction service by specifying the URL of the web site as the access address.

The present invention is useful for broadcasters as the broadcasters may facilitate the joining of additional bidders to a bidding process, and be compensated for such additions.

The present invention is useful for service providers providing auction service as the television viewers are drawn to bid for on-going auctions.

The present invention is useful for service providers providing auction service also because higher commissions may be charged for the auction items sold in accordance with the present invention.

The present invention is useful for television viewers as a television viewer may have non-intrusive access to information on auctions, and purchase the auction items by a convenient user interface.

The present invention is useful for sellers participating in auctions as the sellers may attain greater return for the auction items due to additional pool of bidders participating in accordance with the present invention.

Further features and advantages of the invention, as well as the structure and operation of various embodiments of the invention, are described in detail below with reference to the accompanying drawings. In the drawings, like reference numbers generally indicate identical, functionally similar, and/or structurally similar elements. The drawing in which an element first appears is indicated by the leftmost digit(s) in the corresponding reference number.

Brief Description of the Drawings

The present invention will be described with reference to the accompanying drawings, wherein:.

Figure 1 is a block diagram illustrating an example environment in which the present invention can be implemented;

Figure 2 is a flow-chart illustrating a method in accordance with the present invention;

Figure 3 is a block diagram illustrating an example broadcast system which encodes data related to an auction item in a television signal;

Figure 4 is a block diagram illustrating the details of a transaction enabler in an embodiment of the present invention;

Figure 5 depicts a display screen using which a user may participate in auctions in accordance with the present invention.

Detailed Description of the Preferred Embodiments

1. Overview and Discussion of the Invention

The present invention allows viewers ("television viewers") of television systems to participate in auctions. Typically, the data relating to an item ("auction item") offered for sale in an on-going auction is encoded in a television signal. The encoded information may be displayed while the television viewers watch the images encoded in the television signal. The

viewers may be provided a convenient interface to bid on the auction item.

Auction items consistent with expected viewer profiles may be sold using the present invention. A seller may be able to sell at higher prices as many viewers are likely to bid. For example, a diamond ring may be auctioned towards the end of a romantic movie. The invention is described below with respect to several examples for illustration.

2. Example Environment

Figure 1 is a block diagram illustrating an example environment in which the present invention can be implemented. The environment may include bidding systems 110-A and 110-B, Internet 120, web site 130, broadcast system 150, and television 170. A viewer of television 170 may participate in auctions as described below in further detail.

Web site 130 may provide an auction service. As an illustration, web site 130 may implement the interface of www.ebay.com, well known in the relevant arts. Bidder systems 110-A and 110-B may access Internet 120 to bid on the items offered for sale on web site 130. Bidding systems 110-A and 110-B, Internet 120, and web site 130 may be implemented in a well-known way. Even though the auction service is shown as being provided from web site 130, it should be understood that different other servers using different access technologies (e.g., dial-up) may be used in providing the service.

Broadcast system 150 includes information related to an auction item in a television signal and transmits the television signal on broadcast medium 146 (airwaves, cable, etc.). The data may specify the item offered for sale, the present highest bid, and an access address for enabling the viewer to bid. For example, the access may contain a URL of web site 130. An example embodiment of broadcast system 150 is described below.

The auction may be in progress (on-going) on web site 130, and accordingly broadcast system 150 may access web site 130 to access any data (e.g., present highest bid) for inclusion

in the television signal. Link 134 may be provided on Internet 120 even though a dedicated line is shown in Figure 1.

Viewer bidding system 150 receives the television signal, and enables a viewer to participate in auctions. Viewer bidding system 150 may display the images encoded in the received television signal. In addition, viewer bidding system 150 may recover the data related to the auction item, and display the corresponding information. By appropriate action, the user may indicate a higher bid and transmit the higher bid on virtual link 163 on Internet 120.

In an embodiment, viewer bidding system 150 may include television 170, transaction enabler 160, and remote control 180. Transaction enabler 160 may overlay any images necessary for providing an user interface on top of the images encoded in the television signal ("television signal images"). For example, information identifying the auction item (e.g. Mark McGuire's bat) and the highest bid price may be overlaid on television signal images.

Transaction enabler 160 may encode the overlaid image in a form consistent with conventional television signals for display on television 170. In other words, transaction enabler 160 operates as a 'set-top' box. However, transaction enabler 160 may be integrated into television 170, for example, using embedded chip-sets provided by TeleCruz Technology, Inc. In either case, remote control 180 enables the user to specify the bid price and to transmit the new bid. An example embodiment of transaction enabler 160 is described below in further detail. However, first a method in accordance with the present invention is described first below.

3. Method

Figure 2 is a flow-chart illustrating a method in accordance with the present invention. The method begins in step 201, in which control passes to step 210. In step 210, data identifying an auction item and an access address may be encoded in a television signal. The data identifying an auction item may include both a descriptive component (e.g., "baseball bat signed by Mark McGuire") and a unique code specifying the auction item (or group in case multiple items of the same type are available).

The data may be encoded in one of different formats depending on different criteria, but consistent with an interface at viewer bidding system 150. For example, a unique code identifying an auction item may be encoded as a parameter of a URL (access address) since the web browser's based technology lends well to such encoding and later submission of a bid. The television signal may also be encoded with image frames for display on television signals. Both (images and data related to auction items) encoding may be performed in a known way.

In step 220, the television signal may be broadcasted to television systems covering a large geographic area. In step 240, the television signal may be received at a viewer end (e.g., by transaction enabler 160 of Figure 1). In step 260, the data related to the auction item (encoded in step 210) may be recovered. The recovery generally needs to be consistent with the encoding scheme used by broadcast system. In general, any compatible encoding scheme may be used.

In step 280, the user is provided a convenient user interface to bid on the auction item. Typically, the description of the auction item is displayed, and the user may be provided the option to bid, in which case the bid is submitted to a system identified by the access address. While submitting the bid, the unique code identifying the auction item may be used to specify to the system that the bid relates to that particular item. The access address is used to connect

the user to a central machine (e.g., web site or any server) or person. The user may then submit the bid. The highest bidder is generally entitled to the offered auction item for the submitted bid.

The method and environment described above may be applied in several ways as will be apparent to one skilled in the relevant arts based on the disclosure herein. All such implementations are contemplated to be within the scope and spirit of the present invention. However, it may be desirable to have bidders (television viewers) participation at different points of a broadcast. The manner in which the point can be controlled is described below with respect to broadcast system 140.

4. Broadcast System

Figure 3 is a block diagram illustrating an example embodiment of broadcast system 140. Even though the description of broadcast system is provided substantially with respect to broadcasters producing a television signal, the present invention can be practiced by intermediate broadcasters also. Such advertisements are generally more targeted to the specific geographic profile. Broadcast system 140 may contain production block 310, authoring block 320, broadcast block 330, timing determination block 340, auction data interface 360, and storage 350. Each block is described in further detail below.

Timing determination block 340 may determine the specific time at which to encode data related to an auction item. For example, it may be desirable to broadcast data related to a baseball bat (auction item) when a home run is hit. Timing determination block 340 may be implemented to monitor the scores of the baseball game and generate an indication to auction data interface 360. Several other criteria can be used in determining when to send data related to an auction item.

Timing determination block 340 may also determine when to send updates

corresponding to various auctions. When timing determination block 340 determines to cause update corresponding to an auction to be sent, auction data interface 360 may interact with web site 130 to retrieve a present highest bid from web site 130. The present highest bid may be provided to authoring block 320 for encoding in a broadcast television signal.

Auction data interface 360 receives data on line 134 if a web based auction is on-going for the auction item of interest on web site 130. The data may indicate the present highest bid, bid history, the seller, any comments about the seller. As noted above, auction data interface 360 may provide the data to be encoded in the television signals. The data may contain, in addition to the data retrieved from web site 130, data identifying the auction item (descriptive component and unique code).

Some of the data may be pre-stored in storage 350 also. For example, it may be desirable to display graphic icons on television systems to represent different auction items. Bit maps representing the graphics icons may be stored in storage 350. In general, auction data interface 360 may gather any data which may be of interest to bidders, and pass the data to authoring block 320.

Production block 310 may contain different components such as cameras which are used to film a show/program. The display signal is preferably in a form suitable for eventual transmission as a television signal. In general, production block 310, may encode images in a display data portion of a television signal. The images may be displayed later on a television system for viewing a broadcast program. Production block 310 may be implemented in a known way.

Authoring block 320 encodes data received from auction data interface 360 into television signals. The data may be encoded according to any convention, and transaction enabler 160 may need to be accordingly designed. Several such conventions can be designed

in known way. Authoring block may either store the resulting signal in storage 350 or forward to broadcasting block 330.

In one embodiment, authoring block 320 encodes the data in non-display portion (e.g., vertical blanking interval) of the display signal. Such encoding may be performed in a known way. In an alternative embodiment, the data may be encoded in other portions (e..g, least significant bits of pixel data elements representing an image) as well. This alternative embodiment is described in further detail in co-pending U.S. Patent Application Entitled, "Encoding Hot Spots in Television Signals", Serial Number: 09/276,266, Filing Date: March 25, 1999, which is incorporated in its entirety into the present application.

Even though the encoding is described with reference to analog television signals, it should be understood that the present invention may be practiced in conjunction with digital television signals (e.g., those suitable for HDTV) also. Some of the techniques described in this application may be employed for such encoding in the digital television signals. Many other techniques will be apparent to one skilled in the relevant arts based on the disclosure herein. Such other techniques are also contemplated to be within the scope and spirit of the present invention.

Broadcast block 330 may broadcast television signals (containing the hot spot data in the display data portion) in a known way. It should be noted that the television signal can be in progressive scan format or interlaced format. Production block 310 and authoring block 320 need to be implemented taking into consideration the transmission standard (progressive vs. interlaced, and digital vs. analog) of the television signals. Thus, broadcast block 330 generates television signals containing data which may be used to enable television viewers to bid on the auction items.

Transaction enabler 160 receives the television signals and enables a viewer to bid on

the auction items. Example embodiments of transaction enabler 160 are described below in further detail. Before describing example embodiments of transaction enabler 160 in detail, it is helpful to understand some typical problems with the user interface.

5. Problems and Solutions

In one embodiment, a highest present bid may be encoded in the television signal, and the user may submit a higher bid that the highest present bid. One problem associated in the environments of Figures 1 and 2 is that many bidders may bid for the auction item based on the same highest bid. As the bids are generally marginally more than the present highest bid, the approach may not maximize the return for the seller.

Accordingly, an improvement may be implemented in which an "auction close time" (time at which the auction for the auction item ends) may be associated with the auction item. The auction close time may also be encoded and transmitted in the television signals. Thus, viewers may choose a later convenient time for bidding on the auction item. However, in such a situation, viewer bidding system 150 may need to store the required data.

Yet another problem is, a viewer may wish to know an updated highest bidding price before actually submitting a bid. Thus, the viewer may be provided a convenient user interface to request a 'present highest bid' associated with an auction item of interest. The updated price may also be received on virtual link 163. In this case also, viewer bidding system 150 may need to store the required data.

In yet another scenario, a viewer may wish continuous updates of the highest bidding price. Accordingly, a viewer may be provided an option of initiating a small window in which the updates to the highest bids are provided continuously (e.g., when highest bid changes or every 3 seconds). An embodiment of transaction enabler 160, which provides for at least these features is described below.

6. Transaction Enabler

Figure 4 is a block diagram illustrating the internals of an example embodiment of transaction enabler 160 containing image decoder 410, memory 430, recovery block 420, processor 450, digital to analog converter (DAC) 485, multiplexor 480, infra-red (IR) receiver 460, telephone interface 470 and broadband interface 475. Each component is described below in further detail.

Image decoder 410 generates pixel data elements representing image frames encoded in a television signal received on broadcast channel 146. In response to the operation of remote control unit 180, image decoder 410 may store the pixel data elements representing an image frame in memory 430. Such storage enables overlays. Image decoder 410 may be implemented in a known way. Memory 430 may represent several memory modules such as fast random access memories and relatively slower non-volatile memories. The non-volatile memories may store data and program instructions which enable the operation of the present invention.

Recovery block 420 recovers the data related to auction items encoded in the received television signal. In general, recovery block 420 needs to be implemented consistent with any conventions or protocols used at broadcaster end 380 for encoding the hot spot data. If the data is encoded in non-display portions (e.g., VBI), the data may be recovered in a known way.If the data is encoded in display data portion (i.e., in images), recovery block 420 may examine the pixel data elements stored in memory 430 to recover the data. Further details of recovery are noted in co-pending U.S. Patent Application Entitled, "Encoding Hot Spots in Television Signals", Serial Number: 09/276,266, Filing Date: March 25, 1999, which is incorporated in its entirety into the present application.

Infra-red (IR) receiver 460 receives remote control signals from remote control unit 180, and provides digital data representing the remote control signals to processor 450. The control signals may indicate whether the user wishes to see auction item related data, to enter the bid, to receive an updated present highest bid, etc. Several features of the user interface may be activated by a viewer using IR receiver 460. IR receiver 460 may be implemented in a known way. It may be noted that other receivers which receive control signals from viewers and provide corresponding digital data to processor 450 may be implemented.

Telephone interface 470 enables a telephone call to be initiated. Such telephone calls may be generally initiated either to connect to the Internet via an ISP or to contact a phone with a live-operator. When a telephone call is initiated with a live operation, telephone interface 470 may provide the necessary micro-phone (for a viewer to speak) and receiver for reproducing audible voice. Alternatively, a user may utilize a conventional telephone set that is attached to line 335.

Broadband interface 475 may provide a high speed connection (e.g., using a local area network, digital subscriber loop technology or cable interface) to connect with a web server (corresponding to an URL) or even initiate a voice call (e.g., using voice over Internet Protocol). Telephone interface and broadband interface may be logically viewed as being part of line 163 of Figure 1. In general, broadband interface 475 and telephone interface 470 provide the communication to a system (specified by access address) providing auction service.

Processor 450 receives data related to auction items from recovery block 420, and enables a user to send a bid to a system identified by an access address. The transmission of the bid may be either by broadband interface 475 or telephone interface 470 as specified by the type of access address. Processor 450 may also implement the user interface features noted

in the section above.

For a suitable user-interface, processor 450 may control the images displayed on television system 110. For example, processor 450 may overlay information in the auction items related data on the television signal image. Specifically, the portion to be overlaid on television images may be provided by processor 450, and control line 481 may be controlled to accomplish the overlay function. However, when a user does not wish to bid or when data related to auction items is absent in television signals, processor 450 may control select line 481 to cause the television signal received on line 146 to be passed directly on line 167. In addition, processor 450 may cause auction related data to be displayed in a transparent mode. Typically, techniques such a half-tone control are used for achieving such transparency of display.

If the access address is a URL, transaction enabler 160 may need to operate as a web-browser. Processor 450 may enable such an operation by executing the program instructions provided by memory 430. The web-browser enables transaction enabler 160 to receive different web-pages in a known way. Processor 450 may convert the web pages into image frames, and encode the image frames into a television signal having a format compatible with conventional television signals such that the images can be displayed on television system 110. Well known methods may be employed for such conversion and encoding.

Therefore, transaction enabler 160 may operate in conjunction with broadcast system 140 to enable a television viewer to participate in auctions. As a result, viewers of television systems may be drawn to participate in auctions which are generally accessed mostly by users surfing the world-wide-web. It should be understood that web site 130 and broadcast system 140 may be integrated as one unit depending on the available technologies, and in such a case,

transaction enabler 160 may communicate with such a unit directly. The present invention is described in further detail below with reference to an example user interface considering some of the description of above.

7. User Interface

Figure 5 is a diagram illustrating the manner in which transaction enabler 160 may enable a viewer of television programs to participate in auctions. It should be understood that transaction enabler may use other display devices from which a user can participate in auctions. In addition, other types of systems (such as computers) which display images in television signals may also be used to participate in auctions in accordance with the present invention.

Continuing the description with reference to Figure 5, there is shown television display 500 (for example, on television 170). Auction related data may be received in accordance with the present invention, and the relevant data may be displayed in a small window 540. Window 540 is preferably overlaid on television program images as a transparent window using techniques such as half-toning well known in the relevant arts. By using a transparent display, a viewer may be able to watch the programs encoded in the television signal while participating in the auctions.

Window 540 may be used to display the description of the auction item ("McGuire's 70th Home Run Bat" in the example there), the present highest bid, bidder of the highest bid, and the time at which the auction for this item is expected to close may be displayed. The present highest bid may be periodically updated using the data received on the broadcast television signal. On the other hand, a viewer may select (click on) 'Update' text to cause transaction enabler 160 to initiate a dialogue with web server 130, and retrieve updated information for a presently watched auction item. Thus, in Figure 5, such a selection may