(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 November 2005 (03.11.2005)

PCT

(10) International Publication Number
WO 2005/104429 A2

(51) International Patent Classification⁷: H04L 9/00

(21) International Application Number: PCT/US2005/013944

(22) International Filing Date: 21 April 2005 (21.04.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/830,580 22 April 2004 (22.04.2004) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 10/830,580 (CON)
Filed on 22 April 2004 (22.04.2004)

(71) Applicant *(for all designated States except US)*: **DECRU, INC.** [US/US]; 275 Shoreline Drive, Ste. 450, Redwood City, California 94065 (US).

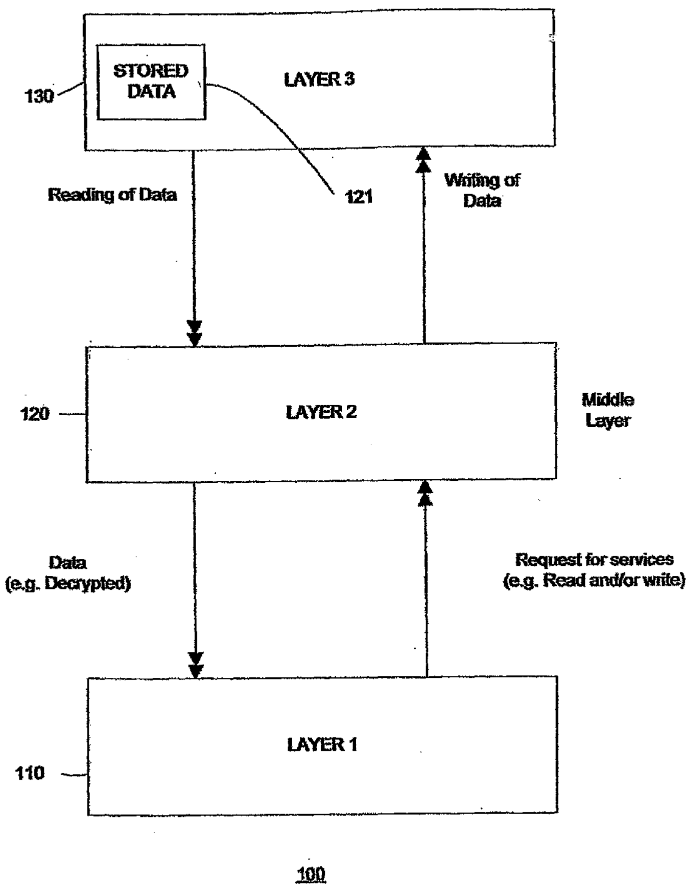
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **PLOTKIN, Serge** [US/US]; 163 Normandy Ct., San Carlos, California 94070 (US). **BOJINOV, Hristo, Iankov** [BG/US]; 151 Calderon Ave., #176, Mountain Veiw, California 94041 (US). **BROWN, Kevin** [US/US]; 185 Selby Lane, Atherton, California 94027 (US).

(74) Agents: **GALLENSON, Mavis, S.** et al.; LADAS & PARRY LLP, 5670 Wilshire Boulevard, Suite 2100, Los Angeles, California 90036-5679 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY,

[Continued on next page]

(54) Title: MANAGEMENT AND/OR DISCARD OF STORED DATA



(57) Abstract: Embodiments of methods, devices and/or systems for a method of managing the retention and/or discarding of stored data are described.

WO 2005/104429 A2

TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

MANAGEMENT AND/OR DISCARD OF STORED DATA

BACKGROUND

This disclosure is related to the management and/or discarding of stored data. This disclosure further relates to management of the retention of and/or discarding of stored data.

One difficulty with state of the art technology relates to the ability to manage the retention and/or the discarding of data that has been stored, such as on a computing platform and/or on a storage area network, for example. If the stored data is contained in an electronic file, for example, deleting the file may not delete the stored data. It may be possible, depending at least in part on the system architecture and/or file management system, to recover the file that has been deleted. In some circumstances, this may be undesirable, such as where the information relates to commercial secrets that a company or other entity would like to permanently discard.

BRIEF DESCRIPTION OF THE DRAWINGS

Subject matter is particularly pointed out and distinctly claimed in the concluding portion of the specification. The claimed subject matter, however, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference of the following detailed description when read with the accompanying drawings in which:

FIG. 1 is a schematic diagram illustrating an embodiment of a typical architecture in which an embodiment of a method of managing the retention and/or discarding of stored data may be implemented;

FIG. 2 is a schematic diagram illustrating an embodiment of a tree hierarchy for a set of keys for an embodiment of a method of managing the retention and/or discarding of stored data;

FIG. 3 is a schematic diagram of an embodiment of a portion network that may employ an embodiment of a method of managing the retention and/or discarding of stored data;

FIG. 4 is a schematic diagram illustrating an embodiment of a file structure for the embodiment shown in FIG. 3;

FIG. 5 is a schematic diagram of another embodiment of a network that may employ and embodiment of a method of managing the retention and/or discarding of stored data; and

FIG. 6 is a schematic diagram illustrating another embodiment of a hierarchy for a set of keys for an embodiment of a method of managing the retention and/or discarding of stored data.

SUMMARY

Broadly, this writing discloses methods, devices and/or systems for managing the retention of and/or the discarding of stored data.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth to provide a thorough understanding of the claimed subject matter. However, it will be understood by those skilled in the art that the claimed subject matter may be practiced without these specific details. In other instances, well-known methods, procedures, components and/or circuits have not been described in detail so as not to obscure the claimed subject matter.

One difficulty with state of the art technology relates to the ability to manage the retention and/or the discarding of data that has been stored, such as on computing platforms, storage area networks, storage arrays (e.g., EMC, DMX), file servers (e.g., Windows 2003), and/or filers (e.g., Net App filer), for example. If the stored data is contained in an electronic file, for example, deleting the file may not delete the stored data. It may be possible, depending at least in part on the system architecture and/or file management system, to recover the file that has been deleted. In some circumstances, this may be undesirable, such as where the information relates to commercial secrets that a company or other entity would like to permanently discard. One possible approach is to find all file fragments (including the file copies in backups, disaster recovery locations, etc) and overwrite each fragment several times with random and/or non-random data. This approach is usually impractical and/or time consuming.

FIG. 1 is a schematic diagram illustrating a typical architecture in which an embodiment of a technique to manage the retention and/ or discarding of stored data may be implemented, although the claimed subject matter is not limited in scope to this particular architecture. In this particular embodiment, FIG. 1 includes a first layer 110, a second layer 120 and a third layer 130. Thus, for this particular embodiment, first layer 110 may make a request for services, such as that data be written and/or read. Second layer 120 may receive the request and may then fulfill it, assuming, for example, that it is able to do so. There are a variety of services that may be provided by second layer 120. Frequently such services are data-related, such as authentication, authorization, and/or data storage and/or retrieval, although these are just examples.

In this particular approach, layer two (also referred to as second layer 120) may supplement or enhance services that may be available from layer three (also referred to as third layer 130). Again, although the claimed subject matter is not limited in scope to this approach or architecture, it is, nonetheless, a common one. For example, web proxy servers may employ this approach or architecture. One service that might also be provided by layer two includes security. For example, this may include firewall functionality, such as packet filtering, packet inspection (e.g., stateful and/or stateless), packet format validation, terminating IPSec connections, and/or the like. Another service that might be provided includes data encryption and/or decryption, as explained in more detail hereinafter. Without loss of generality, in this context, encryption includes a process in which data is coded so that the content of the data is not capable of being employed or understood by a person or a device without first being decoded back to the previous form or format it had prior to being encrypted. Thus, decryption, in this context, includes a process of decoding encrypted data back to the form or format it had prior to encryption.

Thus, in this particular example, if first layer 110 requests that data be written, second layer 120 may encrypt the data to be written. The data, once encrypted, may be stored by or at a third layer, such as 130. This is illustrated in FIG.1 by 121. Likewise, second layer 120 may, upon another request for services by first layer 110, such as a read request, retrieve the stored, encrypted data from layer three, decrypt it, and provide it to first layer 110. One potential advantage of an embodiment, such as previously described, is that encryption and/or decryption of the data may be made transparent to third layer 130, although it is not necessary that this be the case, even for this embodiment and, thus, the claimed subject matter is not limited in scope to embodiments where this is so. Likewise, although the claimed subject matter is not limited in scope in this respect, the encryption may be also made transparent to layer 1, e.g., the "consumer" of the services. Likewise, in another embodiment, any two layers, such as layer 1 and layer 3, may reside on the same computing platform and even comprise the same layer in some embodiments, although the claimed subject matter is not limited in scope in this respect, of course. Nonetheless, for such an embodiment, the encryption and/or decryption of data stored at or on third layer 130 will not impact the operation of layer 130. In this example embodiment, layer 130 treats the data substantially the same regardless of whether or not the data is encrypted. This may provide some benefits, such as making interoperability with other systems possible. Of course, this is just one example of an embodiment of a technique for managing the retention and/or discarding of stored data and, as previously stated, the claim subject matter is not limited in scope to such an embodiment.

The following discussion details several possible embodiments for accomplishing this, although these are merely examples and are not intended to limit the scope of the claimed subject matter. As described above, under some circumstances it may be desirable to discard stored data and have the data discarded in a manner so that it may not practicably be recovered. In this context, the term discard includes a practical inability to recover the data that has been discarded. Due at least in part to the frequency in which users of today's computing platforms tend to store files, such as spread sheets, word documents, e-mail and the like, multiple copies of such documents typically will exist on a variety of storage media and/or computing platforms. Likewise, additional copies may include tape backups, disaster recovery, local backups, etc. Thus, it may be extremely difficult, if not nearly impossible, with this type of behavior on the part of users, to discard a file. Likewise, even if there are no remaining copies of a specified file, meaning that they have all effectively been "deleted," on some computing platforms a file or other stored data is "deleted" simply by marking it "deleted," but the storage media is not "cleaned" by writing over the data so that it may no longer be recovered. For example, in some operating systems, such as "Data On Tap" produced by Network Appliance, sometimes new data does not overwrite prior data. Instead, a new sector of a disk or other storage media is written to and, thus, logically overwritten data is still possible to recover from the physical medium (e.g. hard disk), and in addition, deleted files are not necessarily overwritten. This may allow others, therefore, to recover data that has been stored and deleted.

Again, today's document retention policies in a variety of contexts make it desirable to be able to discard stored data at specific instances or times. In one embodiment of a method of discarding stored data, a key that was previously used to encrypt the stored data is discarded. In this context, a key includes any set of symbols, typically in the form of bits or bytes of data, employed to encrypt and/or decrypt a set of data using any currently known or to be later developed encryption technique. Likewise, depending at least in part on the context, the term key may be used to refer to multiple keys. To discard a particular file or data set, the key used to encrypt this file or data set can be discarded.

In this context, the term data set or set of data is intended to encompass any and all data storage, regardless of form, either currently known or to be later developed. Thus, this may include, for example, a file, a portion of a file, a sector or partition of a disk or LUN, a region of a database and/or any combinations thereof. Likewise, it may include storage on any type of media, such as CD-ROM, disk, flash memory, etc., and/or in any physical form, such as electronic signals, optical signals, etc., whether currently known or to be later developed. Furthermore, discarding of the key may be accomplished in any one of a number of ways.

For example, a key may be stored on a particular media and the media may be discarded in some fashion or otherwise permanently destroyed. Once this happens, if a sufficiently strong encryption is employed, for example, it is not practicable to recover the data. Thus, for an embodiment of a system for retaining and/or discarding stored data, if it is desired that ease of administration be present, the same key may be employed for all files and, hence, discarding the key in the manner previously described results in discarding the stored data in files that have been encrypted with that key.

As alluded to above, the key may be stored on conventional media, such as a disk, CD-ROM, floppy disks, or on printed paper, for example. Thus, simply discarding this media or permanently destroying the media discards the key, as is desired. Alternatively, a protected form of media may be employed, such as a smart card, tamper resistant hardware, or the like. Such a medium of storage or piece of hardware, for example, may have the capable to keep a key relatively secure and may also include the capability to when instructed physically "delete" the key so that it is at least practically not recoverable from the media or hardware and is, therefore, discarded. Here, again, destroying the protected media ultimately discards the data. It may be worth noting, in this context, that, depending upon the particular operating system and/or computing platform, storing the key with the encrypted data may make it difficult to successfully discard the stored data that has been encrypted. Again, as previously described above, for the stored data to be discarded, if the key has also been stored in a device that supports snapshots (e.g., Network Appliances filer), it is practically difficult to erase the stored key by writing over the locations that contain the key. It is likewise noted that where the key has been stored in multiple locations, it is, of course, desirable to discard or permanently destroy those multiple copies.

If better granularity is desired, in an alternative embodiment, each file may be assigned a separate and distinct key, for example. In this example, discarding a key results in the data stored in the associated file likewise being discarded. One disadvantage of this approach, however, is that saving the key separately from the file, such as in flash memory or a proxy device, for example, may mean that a large additional amount of memory will be employed, e.g., to support a large number of files. Likewise, as indicated above, storing the key in the file and/or together with the file may make it difficult to discard the key.

In yet an alternate embodiment, a hierarchical key scheme may be employed in which one or more keys may be manipulated to efficiently discard data. In this particular context, manipulating may include discarding at least one of the keys in the hierarchy and/or it may include changing at least one of the keys in the hierarchy. Here, the hierarchy has a tree structure, although the claimed subject matter is not limited in scope in this respect. FIG. 2 is a schematic diagram illustrating one embodiment of such a key hierarchy. In this particular embodiment, a "root key" includes a key at a top of a hierarchy. Retention key(s) include(s) key(s) that are at least one step removed from a root key. Three retention keys, for example, are designated by nodes 220, 230 and 240 in FIG. 2. Likewise, here, the file keys are one level removed from the retention keys, although, of course, the claimed subject matter is not limited in scope in this respect. For example, several additional layers of retention keys may separate the file keys from retention keys 220, 230, and 240. Here, however, the file keys are designated by nodes 260, 270, 280, 290, 215, 225, 235, 245 and 255. Likewise, additional layers of files and file keys may be present in which file keys higher in the hierarchy operate as retention keys for files lower in the hierarchy.

In this particular hierarchical structure, the root key designated by node 210 in FIG. 2 is used to encrypt and/or decrypt the retention keys, designated by 220, 230 and 240. Likewise, the retention keys are, in turn, used to encrypt and/or decrypt the file keys that here are one step removed from these retention keys. Therefore, retention key 220 is employed to encrypt file keys 260, 270 and 280. Likewise, retention key 230 is employed to encrypt file keys 290, 215 and 225. Furthermore, retention key 240 is used to encrypt file keys 235, 245 and 255. In this particular embodiment, therefore, the file keys may be stored together with the file data. Thus, for this particular embodiment, the root key and retention keys are stored separately.

For this particular embodiment, therefore, if it is desired to discard the data contained in the files, for example, the root key may be discarded, such as by permanently destroying the media in which it is stored, as previously described, for example, in connection with other embodiments. Alternatively, if a particular set of files are to be discarded, then the particular retention key for those files may be discarded. For example, if it is desired to discard the files that correspond to file keys 260, 270 and 280, then retention key 220 may be discarded. Likewise, if it is desired to discard a single file (e.g. the one that uses key 270), then a new retention key is created, all files that we desire to keep (260, 280) have their file keys re-encrypted with the new retention key, and the old one is deleted.

In the approach described above, although, of course, the claimed subject matter is not limited in scope to this embodiment, it should be clear that a file is decrypted by identifying the appropriate retention key, using the root key to decrypt the retention key. using the decrypted retention key to decrypt the file key and using the decrypted file key to decrypt the stored encrypted data. In one particular embodiment, for example, an index of files associated with a retention key may be stored in metadata as well as the encrypted file keys. Likewise, as previously suggested, the root key may be used to encrypt the retention keys and may be stored separately.

Although the previously described embodiment illustrates a key hierarchy with three levels, in an alternative embodiment, as previously suggested, the number of levels to be employed is not restricted. Likewise, in an alternative embodiment, the hierarchy may not have a pure tree-like structure or a structure that even resembles a tree. For example, in another implementation, the file key may be encrypted once with its retention key and once with the root key.

Likewise, in one embodiment, keys at different levels may be encrypted by keys from a prior level or such keys may be stored in plain text depending upon the approach that is desired. If a key resides in tamper-resistant hardware, it may be stored in "cleartext" to simplify implementation. In this context, cleartext refers to storing the data in the form that ultimately is used for data processing, rather than storing the data in a form that is decrypted before use. It may be desirable, for example, for the root key to be in "cleartext" inside a tamper-proof hardware module. Each time a decryption operation is performed, in one possible embodiment; this key may be used to decrypt appropriate sets of keys from the hierarchy to use those keys to decrypt the file or data set. In addition, it may be possible for there to be several "root" keys. Also, in an embodiment, instead of using the key from the previous level, one could use one or more keys from multiple levels above to encrypt the key at the current level. Thus, depending, of course, upon the particular context, it might be beneficial for some of the keys to not be encrypted by a key in such an embodiment. Likewise, the keys may be stored as part of an encrypted file, separately from the file but on the same media, on separate (conventional) media, or on separate protected media, for example, such as previously described.

With such a system, data (including key data), may be discarded by discarding the key used to encrypt it. It may also now be appreciated that a tree hierarchy, as previously described, may be implemented seamlessly in some embodiments. For example, if it desired to discard particular files, those files may be "deleted" and reside in the recycle bin. Thus, in one embodiment, a computer platform may be configured to automatically cycle through the files that have not been designated as "deleted," re-encrypt those files keys with one or more new retention keys, and then discard one or more prior retention keys. If a file is encrypted with a "file key," the file key may be encrypted with a retention key also stored in the file metadata. Thus, to discard files, a new retention key may be selected and used to re-encrypt the file keys of the files that one would like to retain. This might be advantageous since re-encrypting a file key is typically more convenient than re-encrypting a file. In an alternative embodiment, the file key (encrypted with a retention key or some other set of keys in a hierarchy) might reside on a different file system/device/array/etc. than the file itself. In this case, by re-encrypting the file key, one can effectively discard a file or files even if these files reside on a Write-Once-Read-Many (WORM) device or any other device that does not support writing over previously written data (e.g. Network Appliance's Snap-Lock device). Likewise, in still another embodiment, a "history" of recently "deleted" retention keys may be retained to facilitate recovery from erroneous deletions. Such recently deleted keys can be periodically purged.

As will now be appreciated, any system of key dependencies is included within the scope of the claimed subject matter. The prior description merely describes several potential embodiments. Thus, in yet another embodiment, a hierarchy of keys may topologically comprise a directed acyclic graph, potentially with several "root" nodes, e.g., nodes without incoming edges, as illustrated, for example, in FIG. 6. In such an embodiment, each node may correspond to a key, and a key may be stored with other keys, depending upon the particular embodiment, corresponding to some or all of the nodes that have outgoing edges pointing to the node corresponding to this key. Likewise, this key may be stored as an encrypted key. In this particular embodiment, a key may be encrypted with those keys corresponding to the parent nodes. Thus, typically, in such an embodiment, keys except root key(s) are stored encrypted. The keys in the middle of the hierarchy or graph or at the "leaves," which correspond to nodes of a tree with no outgoing edges, are decrypted as appropriate for use. Likewise, in this embodiment, to decrypt a key involves decrypting its parents.

Instead of deleting data, such as a partition of a disk or LUN, a fraction of a file, a storage region where a certain part of a database is stored, and/or a collection of any of the above, it is sufficient to discard the associated keys. One may either discard the key used to encrypt the data, or its parent keys in the hierarchy, or parents of parents, etc. In graph-theoretic terms, for this particular embodiment, it is sufficient to delete any node on one of the paths from a root to the node to be discarded. However, discarding the key associated with a node in the hierarchy, for this embodiment, discards its children. To discard some of its children, a new key for the associated node may be created and the children to be retained may be re-encrypted with this new value. Likewise, as previously alluded to, data may be discarded based at least in part on a number of parameters, such as, for example, time, type of file, storage location, size of file, keywords in the file, etc.

The embodiment immediately described above provides a system in which a key in a tree hierarchy depends on the keys pointing to it and the keys that point to those keys, as shown, for example, in FIG. 6, referred to here as an "AND" scheme. An alternative scheme may comprise one in which a key may be recovered as long as there is at least one path from the root that has not been discarded, referred to here as an "OR" scheme. Thus, depending on the particular embodiment, a mix of these approaches may be used to both provide ways to discard data and also provide redundant key paths in situations in which it may prove useful. Thus, in this context, in one embodiment, an "AND" scheme may, in one embodiment, involve using multiple encryptions, as previously discussed; whereas an "OR" scheme may, in one embodiment, involve encrypting a key with each parent key separately and storing the encrypted copy separately, or, alternatively, allowing a certain set of parent keys to be available. For example, any M out of N parents keys may be available, where N is greater than M, such as by representing the particular keys as a polynomial P(x) of degree M, evaluating P(x) for N distinct values of x, and then encrypting each result with the corresponding parent key. In this case, as long as M parent keys are present, e.g. Key 1 to Key M, for example, then the polynomial evaluated at M values may be recovered. From there the whole polynomial may be recovered and, hence, the key it represents. It is noted, of course, that this is simply one example of a technique for implementation an "OR" scheme and the claimed subject matter is not limited in scope to employing this approach. Many other approaches are possible and are included within the scope of the claimed subject matter.

In an alternate embodiment, a key management scheme may be employed to implement a time-based retention and/or data discard policy, as suggested above with respect to a data retention and/or data discard policy based at least in part on a set of parameters. As one example, a new retention key for the files created during a given week may be applied using a key hierarchy similar to the previously described key hierarchy. Likewise, retention keys may be discarded based on age. For example, if one retains the most recently created 52 retention keys and discards any older retention keys files one year old are effectively discarded. Likewise, if it is desired to extend the life of a particular file one may re-encrypt the file key with another retention key, such as just described. Thus, for the discarding of data that is based at least in part on time, in one embodiment, "weekly retention keys" may be employed. Of course, in alternative embodiments, this may be extended to "daily keys" ,"yearly keys", etc.

Likewise, in an alternative embodiment, a generalized key retention scheme may provide for keys that map to any number of different time intervals having any number of different durations. Likewise, such keys may be different in "type", in this context implying different retention policies. Likewise, such keys may furthermore belong to different key hierarchies. As one example, in such a system, a retention key may map to a specified time interval, but on expiration may not be discarded unless administrative action occurs. Such an approach may be desirable in a scheme to ensure that stored data is not inadvertently discarded before it is confirmed that the stored data is to be discarded. Likewise, retention keys may expire in an order unrelated to when they were created and/or may be employed to discard data from a specific time interval, while retaining data from other specific time intervals. Likewise, as previously described, for a given set of files, the retention key may be replaced by a new retention key so that the file key is re-encrypted, effectively discarding those files in which the file key was not re-encrypted.

In another embodiment, retention keys may correspond to a particular time of creation and, therefore, have a time of expiration, but belong to different key hierarchies in which the hierarchies are related at least in part to how long it is desired for the data to be retained. In yet another embodiment, keys from different levels in a particular key hierarchy may correspond to different levels of nested directories in a file system. For example, this may make it possible to discard a directory with all of its contents by discarding the key that maps to this directory from the hierarchy. Once the key is discarded, all keys encrypted by that key are discarded; hence all directories and files contained in the target directory may be discarded.

The embodiments described above, it may be noted, are independent of system architecture. Thus, it is not necessary that three layers be employed and it is not necessary that encryption and/or decryption be transparent to a particular layer; of course, embodiments that include such features are also within the scope of the claimed subject matter, as previously described.

As previously described, one disadvantage of state of the art technology is the ability, potentially, for an unauthorized entity or individual to gain access to data stored on and/or being processed, such as may occur in networking, for example. In this context, networking is typically implemented using at least two computing platforms. A computing platform refers to a system and/or a device that includes the ability to process and store data in the form of signals. Thus, a computing platform, in this context, may comprise hardware, software, firmware and/or any combination thereof.

FIG. 3 is a schematic diagram illustrating an embodiment of a portion of a network that may employ an embodiment of a method of managing the retention and/or discarding of stored data. Reference numerals 310, 320, 330 and 340 denote units that access stored data. These may comprise, for example, clients, servers and the like. Reference numeral 360 denotes a file server that stores encrypted data. Also depicted in FIG. 3 is a directory of file server 360 that includes files respectively denoted 370, 380, and 390. These files, in this example, were encrypted by device 350. Therefore, as illustrated in FIG. 3, units 310-340 comprise previously described layer one, device 350 comprises previously described layer two, and server 360 comprises previously described layer three. Of course, this is merely an example embodiment and any one of a number of different network architectures may be employed that are within the scope of the claimed subject matter.

For this particular embodiment, files 370-390 are illustrated in more detail in FIG. 4. Here, the notation E(x,y) refers to an encryption process where data denoted as x is encrypted with a key denoted y. Thus, as illustrated, the contents of file 370, denoted k1, has been encrypted by file key Fk1. However, also stored in file 370 is key Fk1 having been encrypted by retention key Fr1. Likewise, retention key Fr1 has also been employed to encrypt the keys for files 380 ad 390. These keys are respectively denoted Fk2 and Fk3. Likewise, key Fk2 was employed to encrypt content k2 and Fk3 was employed to encrypt content k3. It is noted that other retention keys and a root key are not shown in this figure, although, depending upon the particular embodiment, the associated files and retention keys may follow a similar structure as previously described. For example, files 370-309 may also store encrypted retention keys.

One approach or technique that may be employed to make unauthorized access to data more difficult is the previously described embodiment. It is worth noting, in this context, that data storage may take any one of a variety of forms and the claimed subject matter is not limited in scope to any particular form of storing such data signals. Any and all methods and/or techniques for storing data signals now known or that may subsequently be developed are included within the scope of the claimed subject matter. As is well-known, there are a variety of file types and/or structures currently in use for storing data. In this context, a file includes stored data related at least in part by the particular format in which the data is stored. As just one example, most clients that employ a Unix-based operating system use the Network File System (NFS) for remote file access. Sun® Microsystems introduced NFS in 1985. Since then, it has become a *de facto* standard protocol, used by over ten million systems worldwide. NFS is particularly common on Unix-based systems, but NFS implementations are available for virtually every modern computing platform in current use, from desktops to supercomputers.

Although the NFS file system and Unix-based operating systems are specifically mentioned above, the issue of management of the retention and/or discarding of stored data may arise for systems other than those that employ Unix or NFS. Essentially, for any instance in which data is stored, this issue may arise. Thus, the scope of the claimed subject matter is not limited to a particular hardware platform, software platform, file type, data type, file structure, data structure, operating system, application, or the like. Furthermore, the claimed subject matter is not limited to a particular implementation of encryption or other security measures.

Referring, again, to the embodiment of FIG. 1, it is desired that encryption and/or decryption be transparent to a third layer, although, again, this is just an example embodiment. While in a previously described embodiment, encryption and/or decryption was assumed transparent to layer three, as previously indicated, this need not be the case in an alternative embodiment. For example, communications may take place between the layers to provide relevant and/or useful encryption and/or decryption information to layer three. Thus, information and/or data may be passed that may reduce the processing load for layer two, for example.

As previously described, embodiments of the claimed subject matter are well suited to a variety of networking applications and/or systems, such as computer network systems, employing a variety of different topologies, including, for example, storage area networking (SAN), although, of course, the claimed subject matter is not limited in scope in this respect. In such an embodiment, although the claimed subject matter is not limited in scope in this respect, a configuration may be employed in which management is accomplished of small, medium, or large networks comprised of storage devices, computers, other computing platforms, and/or the like, that are communicatively coupled to dissimilar storage devices, computers, other computing platforms, and/or the like.

FIG. 5 is a schematic diagram of an example embodiment of a communications network system 400 that may employ an embodiment in accordance with the claimed subject matter. In this example, embodiment 500 comprises a switched fabric 410 and a plurality of devices, such as 420, 422, 424, and/or groups of devices, such as 434, 436, and 438, as indicated with respect to a logical loop 430, for example. References to "a switch" or to "switches" are intended to refer to a generic switch. In this context, then, the term switch includes a device having at least a processor and memory in which the device is adapted to or has the capability to route frames or packets between two or more separate, other devices. In general, a switched fabric, such as fabric 410, may be communicatively coupled to various devices, such as, here, 420, 422, and 424, and may operate as a switching network to allow these devices to communicate with one another. Devices 420, 422, and 424 may comprise any type of device, such as, for example, a computing platform, a storage device, and/or the like, and may be communicatively coupled via fabric 410 by employing point-to-point communications technology or techniques, as one example. In this particular embodiment, fabric 410 comprises a variety of communicatively coupled switches. In this particular embodiment, fabric 410 is also in communication with logical loop 430. Loop 430 here includes devices 434, 436 and 438. In this particular embodiment, loop 430 comprises an arbitrated loop with ring couplings for providing multiple nodes with the ability to arbitrate access to shared bandwidth. It is, of course, appreciated that this description is merely an illustrative example and the claimed subject matter is not limited in scope in any way to this particular embodiment.

As another example, one embodiment may be in hardware, such as implemented to operate on a device or combination of devices, for example, whereas another embodiment may be in software. Likewise, an embodiment may be implemented in firmware, or as any combination of hardware, software, and/or firmware, for example. Likewise, although the claimed subject matter is not limited in scope in this respect, one embodiment may comprise one or more articles, such as a storage medium or storage media. This storage media, such as, one or more CD-ROMs and/or disks, for example, may have stored thereon instructions, that when executed by a system, such as a computer system, computing platform, or other system, for example, may result in an embodiment of a method in accordance with the claimed subject matter being executed, such as one of the embodiments previously described, for example. As one potential example, a computing platform may include one or more processing units or processors, one or more input/output devices, such as a display, a keyboard and/or a mouse, and/or one or more memories, such as static random access memory, dynamic random access memory, flash memory, and/or a hard drive, although, again, the claimed subject matter is not limited in scope to this example. It will, of course, be understood that, although particular embodiments have just been described, the claimed subject matter is not limited in scope to a particular embodiment or implementation.

In the preceding description, various aspects of the claimed subject matter have been described. For purposes of explanation, specific numbers, systems and configurations were set forth to provide a thorough understanding of the claimed subject matter. However, it should be apparent to one skilled in the art having the benefit of this disclosure that the claimed subject matter may be practiced without the specific details. In other instances, well-known features were omitted and/or simplified so as not to obscure the claimed subject matter. While certain features have been illustrated and/or described herein, many modifications, substitutions, changes and/or equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and/or changes as fall within the true spirit of the claimed subject matter.

Claims:

1. A method of discarding stored data comprising:

discarding a key previously used to encrypt said stored data.

2. The method of claim 1, and further comprising:

encrypting multiple sets of stored data using multiple keys; and

encrypting said multiple keys so as to create at least one hierarchy of key encryption dependencies among said multiple keys;

wherein discarding a key previously used to encrypt said stored data includes discarding a key used to encrypt the key previously used to encrypt said stored data.

3. The method of claim 2, wherein said at least one hierarchy of key encryption dependencies includes more than one hierarchy level of key encryption dependencies.

4. The method of claim 2, wherein discarding a key used to encrypt the key previously said to encrypt said stored data includes discarding a key higher in said at least one hierarchy than the key previously used to encrypt said stored data.

5. The method of claim 1, wherein said stored data is contained in a file structure;
said key comprising a file key;
said discarding a key comprising discarding said file key.

6. The method of claim 5, and further comprising:

encrypting said file key with a retention key; and wherein said discarding said file key comprises discarding said retention key.

7. The method of claim 6, wherein the encrypted file key is stored in said file structure.

8. The method of claim 6, wherein the encrypted file key is stored separately from said file structure.

9. The method of claim 8, wherein said retention key is also encrypted and the encrypted retention key is also stored separately from said file structure.

10. The method of claim 8, wherein stored separately comprises stored via a separate storage medium.

11. The method of claim 5, wherein the file key is stored in said file structure.

12. The method of claim 5, wherein the file key is stored separately from said file structure.

13. The method of claim 12, wherein stored separately comprises stored via a separate storage medium.

14. The method of claim 5, wherein said file structure is positioned in a hierarchy of file structures; and further comprising:

encrypting said file key with a file key for a file structure higher in said hierarchy of file structures.

15. The method of claim 14, wherein said discarding said file key comprises discarding said file key for said file structure higher in said hierarchy.

16. The method of claim 15, wherein said discarding said file key for said file structure higher in the file hierarchy comprises discarding a root key.

17. The method of claim 16, wherein said discarding said root key comprises discarding data stored in said hierarchy of file structures.

18. The method of claim 15, wherein discarding said file key for said file structure higher in said file hierarchy comprises discarding data stored in multiple file structures in said file hierarchy below said file structure higher in said file hierarchy.

19. The method of claim 14, wherein said hierarchy of file structures comprises multiple hierarchies of file structures; each of said hierarchies having a root key such that discarding the root key discards the data contents of the file structures of the particular hierarchy.

20. The method of claim 14, wherein encrypting said file key with a file key for a file structure higher in said hierarchy of file structures includes encrypting said file key with at least some file keys higher in said hierarchy along a path to a root key of said hierarchy.

21. A method of implementing data retention comprising:

manipulating one or more keys employed to encrypt said data.

22. The method of claim 21, wherein said manipulating includes discarding at least one of said one or more keys.

23. The method of claim 21, wherein said manipulating includes replacing at least one of said one or more keys.

24. The method of claim 23, wherein said one or more keys comprise a hierarchy of keys; said keys being related in said hierarchy through an AND scheme, an OR scheme, or a combination of an AND/OR scheme.

25. The method of claim 23, wherein said one or more keys comprise a hierarchy of keys; wherein said manipulating includes discarding at least one of said keys in said hierarchy.

26. The method of claim 25, wherein said hierarchy of keys includes parent keys and children keys; and wherein discarding a parent key discards all of its children keys.

27. The method of claim 23, wherein said one or more keys comprise a hierarchy of keys in which at least some of said keys in said hierarchy are encrypted;

said replacing at least one of said one or more keys comprises re-encrypting at least one of said at least some of said keys with a different key.

28. The method of claim 27, wherein said replacing at least one of said one or more keys comprises replacing said at least one of said one or more keys based at least in part on at least one parameter of a set of parameters, said set of parameters comprising: time, type of file, storage location, or size of file.

29. The method of claim 28, wherein said at least one parameter of said set of parameters comprises time.

30. The method of claim 29, wherein said replacing said at least one of said one or more keys takes place after a certain number of weeks, months, or years.

31. A method comprising:

discarding encrypted data stored on a system;

said system comprising a first layer, a second layer and a third layer, and said encrypted data residing on said third layer;

wherein said first layer is capable of making read and write requests directed to said third layer; said second layer is capable of performing services including encryption; and said third layer is capable of performing services including data storage, said encryption being transparent to said third layer.

32. The method of claim 31, wherein said encryption is also transparent to said first layer.

33. The method of claim 32, wherein at least any two of the three layers reside on the same computing platform.

34. The method of claim 31, wherein said encrypted data is discarded by discarding a key previously used to encrypt said encrypted data.

35. An article comprising: a storage medium having stored thereon instructions, that when executed, result in performance of a method of discarding stored data comprising:

discarding a key previously used to encrypt said stored data.

36. The article of claim 35, wherein said instructions, when executed, result in said method further comprising:

encrypting multiple sets of stored data using multiple keys; and

encrypting said multiple keys so as to create at least one hierarchy of key encryption dependencies among said multiple keys;

wherein discarding a key previously used to encrypt said stored data includes discarding a key used to encrypt the key previously used to encrypt said stored data.

37. The article of claim 36, wherein said instructions, when executed, result in said at least one hierarchy of key encryption dependencies including more than one hierarchy level of key encryption dependencies.

38. The article of claim 36, wherein said instructions, when executed, result in discarding a key used to encrypt the key previously said to encrypt said stored data including discarding a key higher in said at least one hierarchy than the key previously used to encrypt said stored data.

39. The article of claim 35, wherein said instructions, when executed, result in said stored data being contained in a file structure; and
said key comprising a file key; and
said discarding a key comprising discarding said file key.

40. The article of claim 39, wherein said instructions, when executed, result in said method further comprising:

encrypting said file key with a retention key; and wherein said discarding said file key comprises discarding said retention key.

41. The article of claim 39, wherein said instructions, when executed, result in: said file structure being positioned in a hierarchy of file structures; and said method comprising:

encrypting said file key with a file key for a file structure higher in said hierarchy of file structures.

42. The article of claim 41, wherein said instructions, when executed, result in said discarding said file key comprising discarding said file key for said file structure higher in said hierarchy.

43. The article of claim 42, wherein said instructions, when executed, result in said discarding said file key for said file structure higher in the file hierarchy comprising discarding a root key.

44. The article of claim 43, wherein said instructions, when executed, result in said discarding said root key comprising discarding data stored in said hierarchy of file structures.

45. The article of claim 42, wherein said instructions, when executed, result in discarding said file key for said file structure higher in said file hierarchy comprising discarding data stored in multiple file structures in said file hierarchy below said file structure higher in said file hierarchy.

46. The article of claim 41, wherein said instructions, when executed, result in said hierarchy of file structures comprising multiple hierarchies of file structures; each of said hierarchies having a root key such that discarding the root key discards the data contents of the file structures of the particular hierarchy.

47. The article of claim 41, wherein said instructions, when executed, result in encrypting said file key with a file key for a file structure higher in said hierarchy of file structures including encrypting said file key with at least some file keys higher in said hierarchy along a path to a root key of said hierarchy.

48. An article comprising: a storage medium having stored thereon instructions that, when executed, result in performance of a method of implementing data retention comprising:

manipulating one or more keys employed to encrypt said data.

49. The article of claim 48, wherein instructions, when executed, result in said manipulating including discarding at least one of said one or more keys.

50. The article of claim 48, wherein instructions, when executed, result in said manipulating including replacing at least one of said one or more keys.

51. The article of claim 50, wherein instructions, when executed, result in said one or more keys comprising a hierarchy of keys; said keys being related in said hierarchy through an AND scheme, an OR scheme, or a combination of an AND/OR scheme.

52. The article of claim 50, wherein instructions, when executed, result in said one or more keys comprising a hierarchy of keys; and said manipulating including discarding at least one of said keys in said hierarchy.

53. The article of claim 52, wherein instructions, when executed, result in said hierarchy of keys including parent keys and children keys; and discarding a parent key discards all of its children keys.

54. The article of claim 50, wherein instructions, when executed, result in said one or more keys comprising a hierarchy of keys in which at least some of said keys in said

hierarchy are encrypted; and said replacing at least one of said one or more keys comprising re-encrypting at least one of said at least some of said keys with a different key.

55. The article of claim 54, wherein instructions, when executed, result in said replacing at least one of said one or more keys comprising replacing said at least one of said one or more keys based at least in part on at least one parameter of a set of parameters, said set of parameters comprising: time, type of file, storage location, or size of file.

56. The article of claim 55, wherein instructions, when executed, result in said at least one parameter of said set of parameters comprising time.

57. The article of claim 56, wherein instructions, when executed, result in said replacing said at least one of said one or more keys taking place after a certain number of weeks, months, or years.

58. An apparatus comprising:

means for encrypting data with one or more keys; and

means for manipulating said one or more keys employed to encrypt data.

59. The apparatus of claim 58, wherein said means for manipulating includes means for discarding at least one of said one or more keys.

60. The apparatus of claim 58, wherein said means for manipulating includes means for replacing at least one of said one or more keys.

61. The apparatus of claim 60, wherein said one or more keys comprise a hierarchy of keys; said keys being related in said hierarchy through an AND scheme, an OR scheme, or a combination of an AND/OR scheme.

62. The apparatus of claim 60, wherein said one or more keys comprise a hierarchy of keys; wherein said means for manipulating includes means for discarding at least one of said keys in said hierarchy.

63. The apparatus of claim 62, wherein said hierarchy of keys includes parent keys and children keys; and wherein said means for discarding a parent key includes means for discarding all of its children keys.

64. The apparatus of claim 60, wherein said one or more keys comprise a hierarchy of keys in which at least some of said keys in said hierarchy are encrypted;

said means for replacing at least one of said one or more keys comprises means for re-encrypting at least one of said at least some of said keys with a different key.

65. The apparatus of claim 64, wherein said means for replacing at least one of said one or more keys comprises means for replacing said at least one of said one or more keys based at least in part on at least one parameter of a set of parameters, said set of parameters comprising: time, type of file, storage location, or size of file.

66. The apparatus of claim 65, wherein said at least one parameter of said set of parameters comprises time.

67. An apparatus comprising:

a computing platform, said computing platform adapted to manipulate one or more keys employed to encrypt data.

68. The apparatus of claim 67, wherein said computing platform is further adapted to manipulate by discarding at least one of said one or more keys.

69. The apparatus of claim 67, wherein said computing platform is further adapted to manipulate by replacing at least one of said one or more keys.

70. The apparatus of claim 69, wherein said one or more keys comprise a hierarchy of keys; said keys being related in said hierarchy through an AND scheme, an OR scheme, or a combination of an AND/OR scheme.

71. The apparatus of claim 69, wherein said one or more keys comprise a hierarchy of keys; wherein said computing platform is further adapted to manipulate by discarding at least one of said keys in said hierarchy.

72. The apparatus of claim 71, wherein said hierarchy of keys includes parent keys and children keys; and wherein said computing platform is further adapted to discard children keys by discarding a parent key of said children keys.

73. The apparatus of claim 69, wherein said one or more keys comprise a hierarchy of keys in which at least some of said keys in said hierarchy are encrypted;

said computing platform is further adapted to replace at least one of said one or more keys by re-encrypting at least one of said at least some of said keys with a different key.

74. The apparatus of claim 73, wherein said computing platform is further adapted to replace at least one of said one or more keys by replacing said at least one of said one or more keys based at least in part on at least one parameter of a set of parameters, said set of parameters comprising: time, type of file, storage location, or size of file.

75. The apparatus of claim 67, wherein said computing platform comprises a switch.

76. The apparatus of claim 67, wherein said computing platform is coupled in a network.

77. The apparatus of claim 67, and further comprising media to store said one or more keys, wherein said media is capable of discarding at least one of said one or more keys

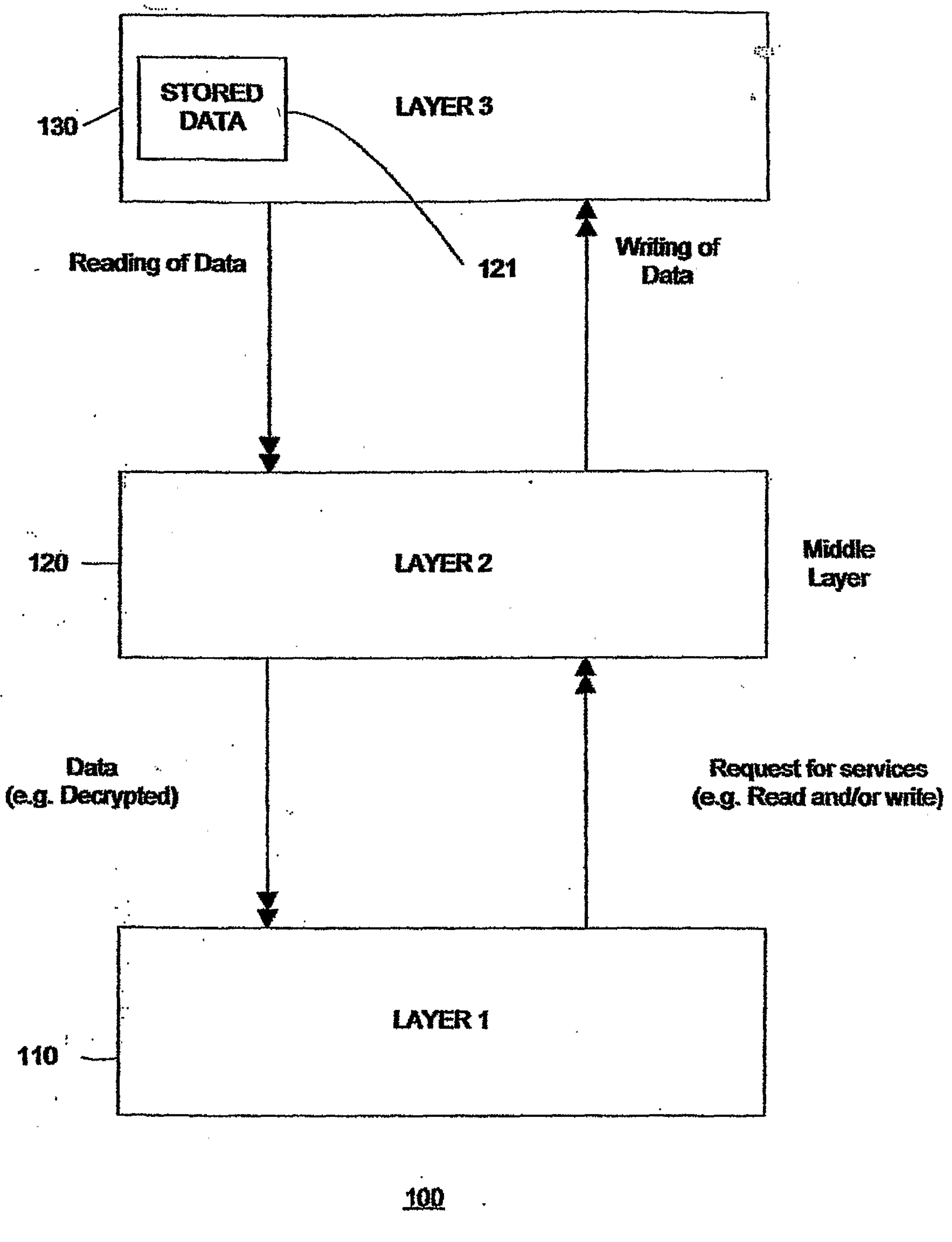

STORED DATA
LAYER 3
130
121
Reading of Data
Writing of Data
LAYER 2
120
Middle Layer
Data (e.g. Decrypted)
Request for services (e.g. Read and/or write)
LAYER 1
110
100

FIG. 1

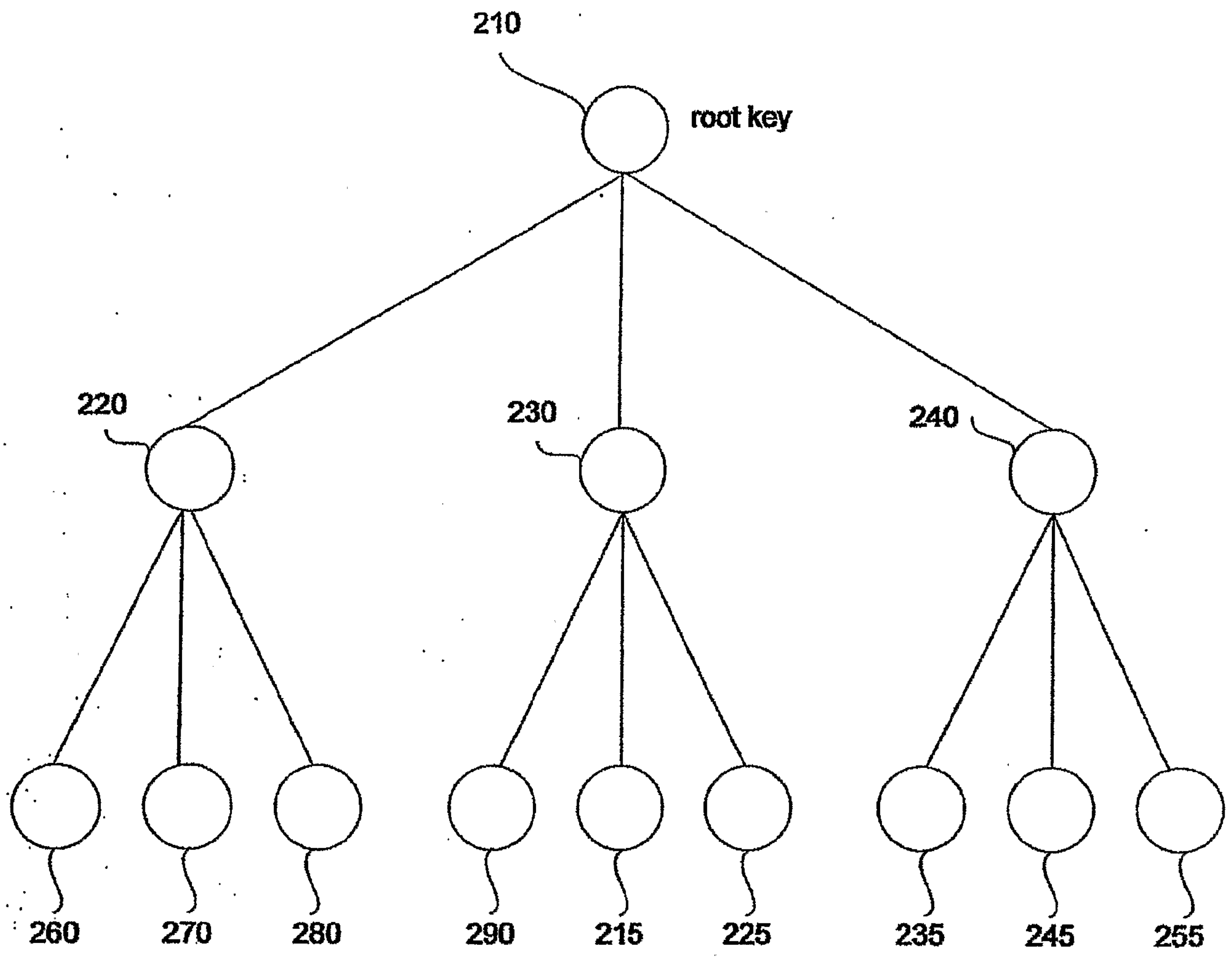


FIG. 2

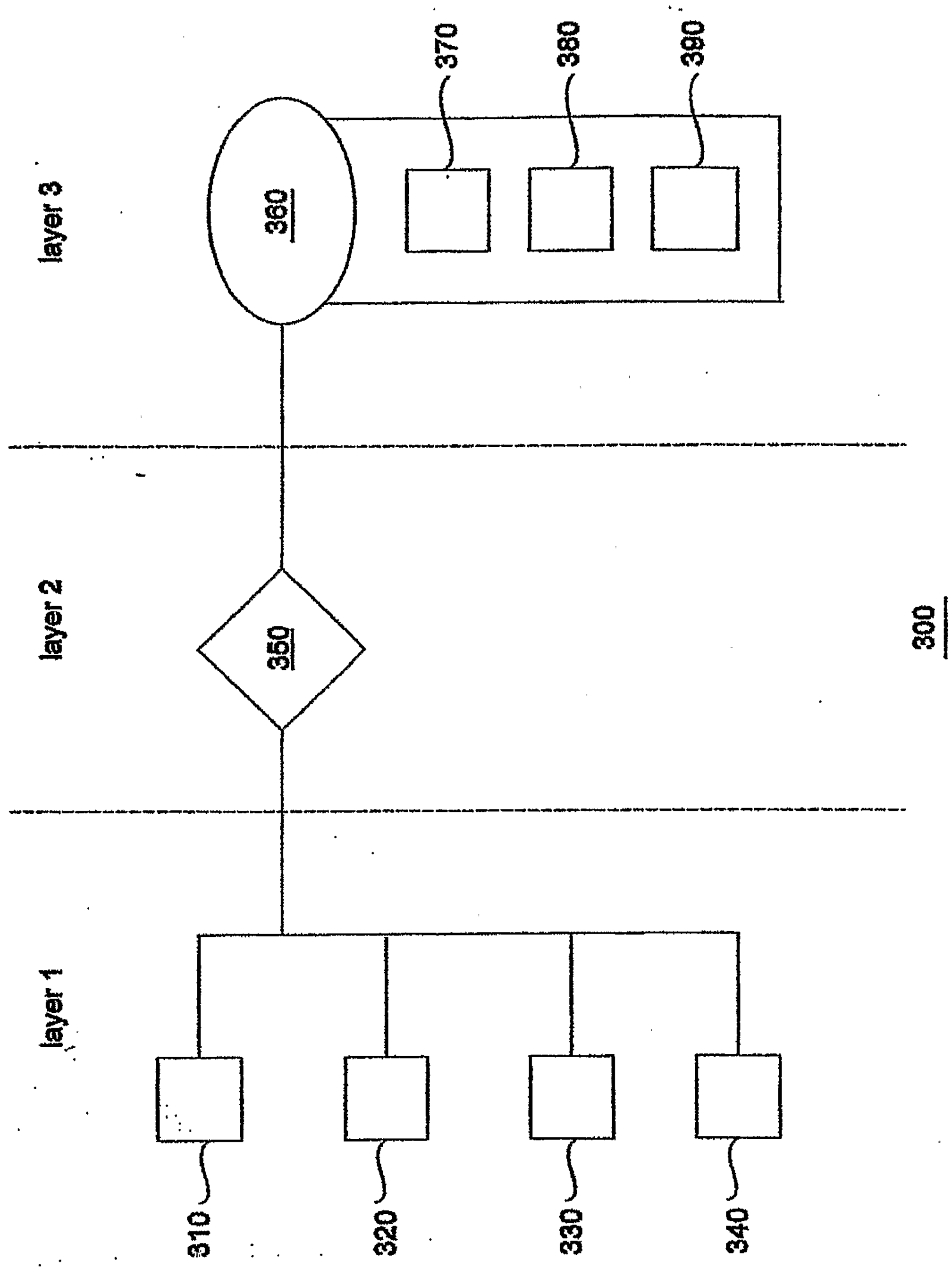
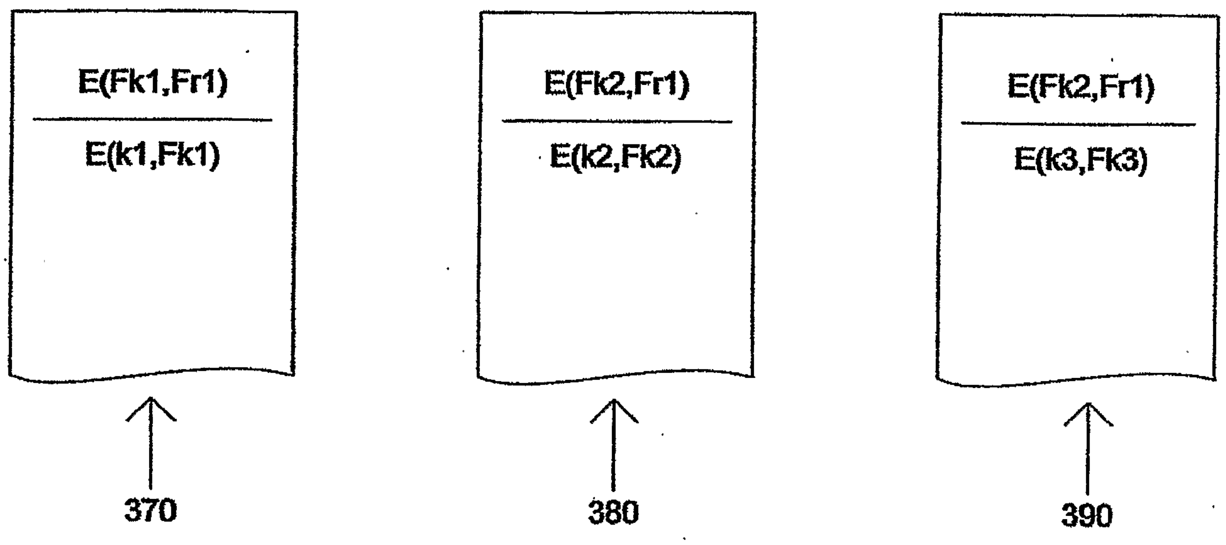


FIG. 3

FIG. 4



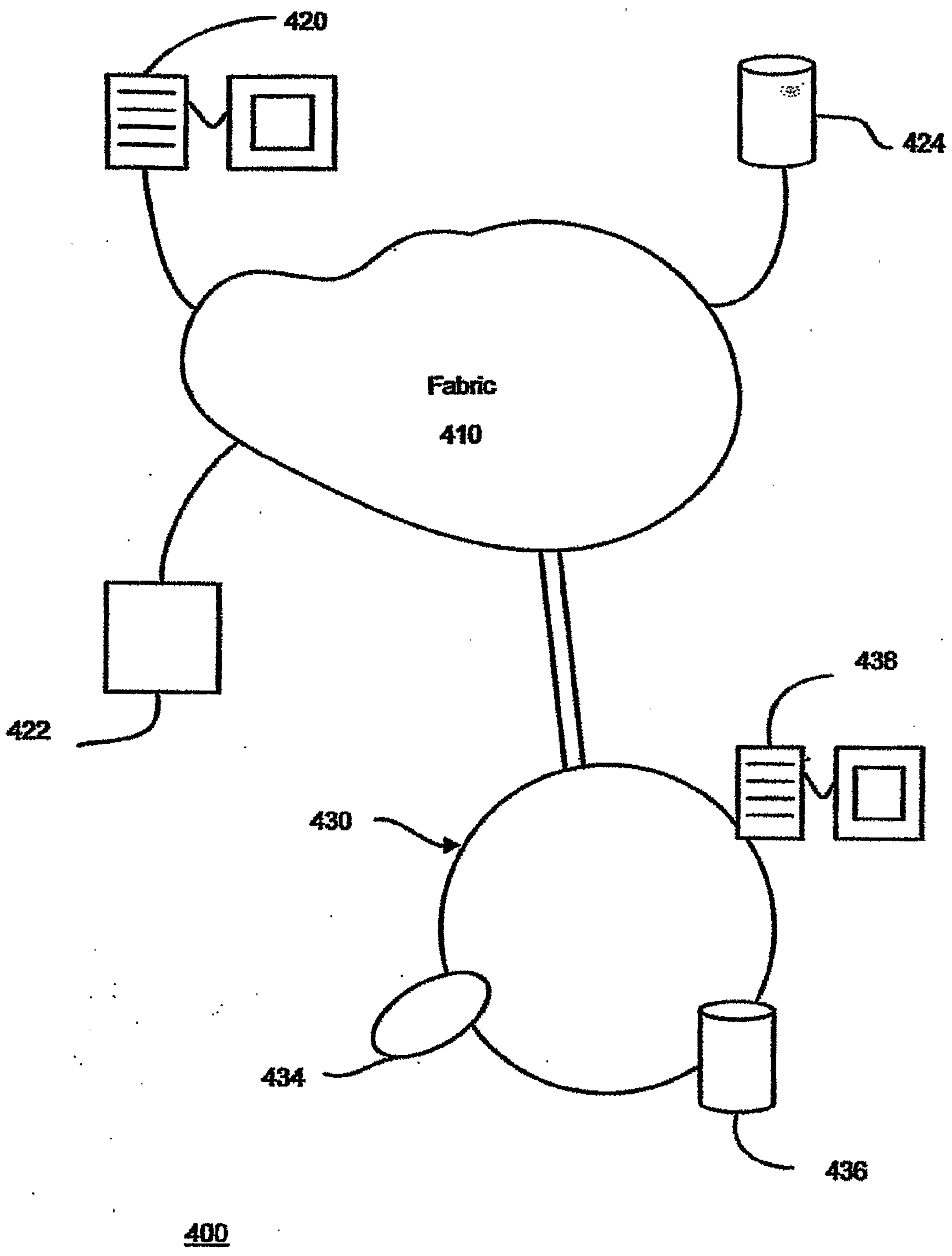


FIG. 5

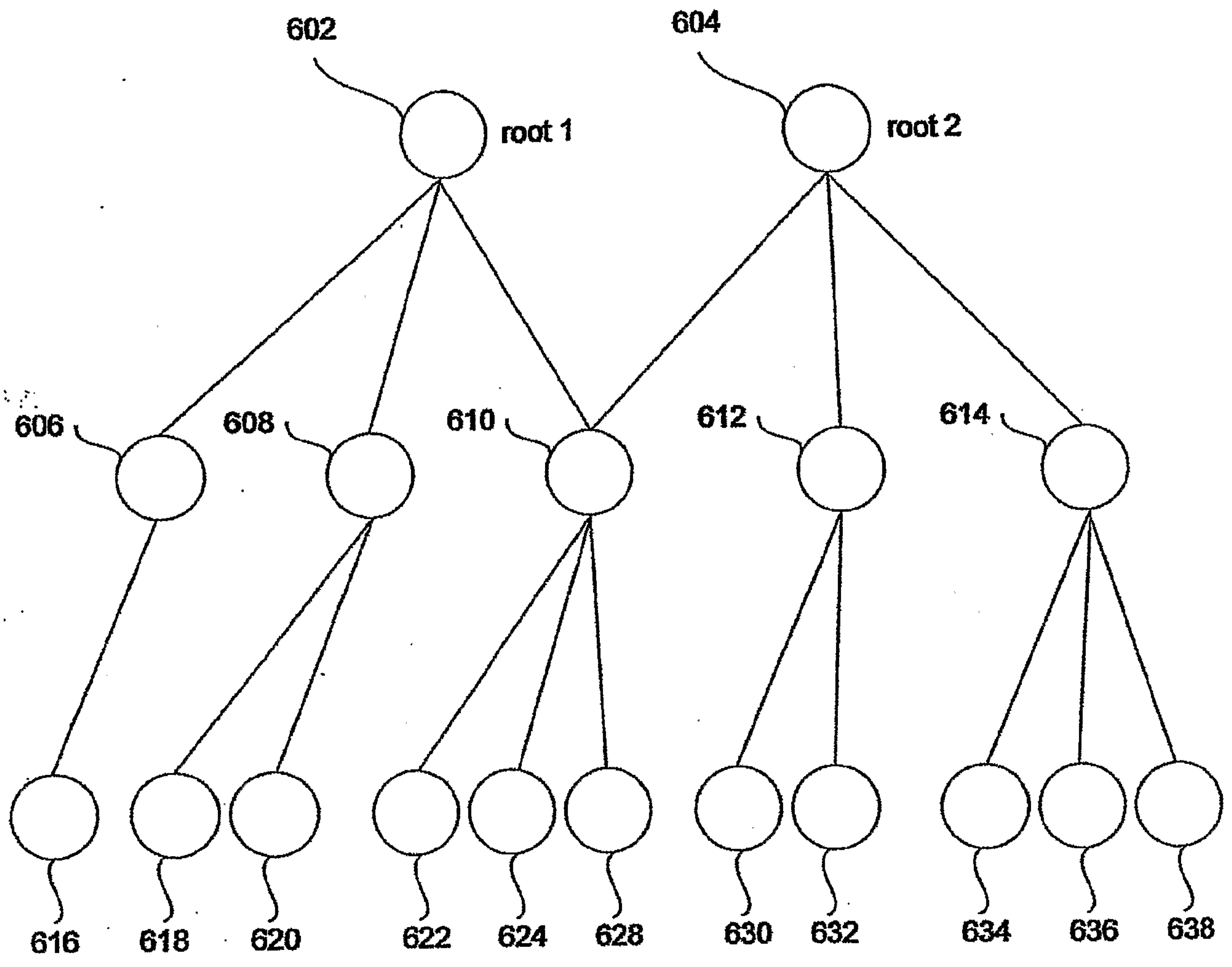

602
604
root 1
root 2
606
608
610
612
614
616
618
620
622
624
628
630
632
634
636
638

FIG. 6