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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 May 2010 (20.05.2010)

PCT

(10) International Publication Number
WO 2010/057181 A2

(51) International Patent Classification:
G06F 12/14 (2006.01) *G06F 12/08* (2006.01)
H04L 9/08 (2006.01)

(21) International Application Number:
PCT/US2009/064786

(22) International Filing Date:
17 November 2009 (17.11.2009)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

12/272,012	17 November 2008 (17.11.2008)	US
12/336,562	17 December 2008 (17.12.2008)	US
12/336,564	17 December 2008 (17.12.2008)	US
12/336,559	17 December 2008 (17.12.2008)	US
12/336,568	17 December 2008 (17.12.2008)	US
12/336,558	17 December 2008 (17.12.2008)	US
12/342,464	23 December 2008 (23.12.2008)	US
12/342,500	23 December 2008 (23.12.2008)	US
12/342,575	23 December 2008 (23.12.2008)	US
12/342,523	23 December 2008 (23.12.2008)	US
12/342,414	23 December 2008 (23.12.2008)	US
12/342,438	23 December 2008 (23.12.2008)	US
12/342,610	23 December 2008 (23.12.2008)	US
12/342,636	23 December 2008 (23.12.2008)	US
12/342,547	23 December 2008 (23.12.2008)	US
12/342,379	23 December 2008 (23.12.2008)	US
12/346,561	30 December 2008 (30.12.2008)	US
12/346,578	30 December 2008 (30.12.2008)	US

(71) Applicant *(for all designated States except US)*: **UNISYS CORPORATION** [US/US]; Unisys Way, MS/S1-108, Blue Bell, PA 19424-0001 (US).

(72) Inventors: **SUMMERS, Scott**; 14 Sugarpine Lane, Collegeville, PA 19426 (US). **FRENCH, Albert**; 355 Grubb Road, Schwenksville, PA 19473 (US).

(74) Agent: **GREGSON, Richard, J.**; Unisys Corporation, Unisys Way, MS/S1-108, Blue Bell, PA 19424-0001 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

WO 2010/057181 A2

(54) Title: SIMULTANEOUS STATE-BASED CRYPTOGRAPHIC SPLITTING IN A SECURE STORAGE APPLIANCE

(57) Abstract: Methods and systems for managing data blocks and I/O requests are provided. One method is a method of managing data blocks in a secure storage appliance. The method includes receiving a block of data associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices, and storing the block of data in a buffer. The method also includes associating the block of data with a state from among a plurality of states, each of the states corresponding to a status of the block of data. The method further includes processing the block of data by performing at least one cryptographic operation on the block of data, and upon completion of processing the block of data, updating the state of the block of data.

SIMULTANEOUS STATE-BASED CRYPTOGRAPHIC SPLITTING IN A SECURE STORAGE APPLIANCE

Technical Field

The present disclosure relates to data storage systems, and security for such systems. In particular, the present disclosure relates to simultaneous state-based cryptographic splitting in a secure storage appliance.

Background

Modern organizations generate and store large quantities of data. In many instances, organizations store much of their important data at a centralized data storage system. It is frequently important that such organizations be able to quickly access the data stored at the data storage system. In addition, it is frequently important that data stored at the data storage system be recoverable if the data is written to the data storage system incorrectly or if portions of the data stored at the repository is corrupted. Furthermore, it is important that data be able to be backed up to provide security in the event of device failure or other catastrophic event.

The large scale data centers managed by such organizations typically require mass data storage structures and storage area networks that are capable of providing both long-term mass data storage and access capabilities for application servers using that data. Some data security measures are usually implemented in such large data storage networks, and are intended to ensure proper data privacy and prevent data corruption. Typically, data security is accomplished via encryption of data and/or access control to a network within which the data is stored. Data can be

stored in one or more locations, e.g. using a redundant array of inexpensive disks (RAID) or other techniques.

One example of an existing mass data storage system 10 is illustrated in Figure 1. As shown, an application server 12 (e.g. a database or file system provider) connects to a number of storage devices 14_1-14_N providing mass storage of data to be maintained accessible to the application server via direct connection 15, an IP-based network 16, and a Storage Area Network 18. Each of the storage devices 14 can host disks 20 of various types and configurations useable to store this data.

The physical disks 20 are made visible/accessible to the application server 12 by mapping those disks to addressable ports using, for example, logical unit numbering (LUN), internet SCSI (iSCSI), or common internet file system (CIFS) connection schemes. In the configuration shown, five disks are made available to the application server 12, bearing assigned letters I-M. Each of the assigned drive letters corresponds to a different physical disk 20 (or at least a different portion of a physical disk) connected to a storage device 14, and has a dedicated addressable port through which that disk 20 is accessible for storage and retrieval of data. Therefore, the application server 12 directly addresses data stored on the physical disks 20.

A second typical data storage arrangement 30 is shown in Figure 2. The arrangement 30 illustrates a typical data backup configuration useable to tape-backup files stored in a data network. The network 30 includes an application server 32, which makes a snapshot of data 34 to send to a backup server 36. The backup server 36 stores the snapshot, and operates a tape management system 38 to record that snapshot to a magnetic tape 40 or other long-term storage device.

These data storage arrangements have a number of disadvantages. For example, in the network 10, a number of data access vulnerabilities exist. An unauthorized user can steal a physical disk 20, and thereby obtain access to sensitive files stored on that disk. Or, the unauthorized user can exploit network vulnerabilities to observe data stored on disks 20 by monitoring the data passing in any of the networks 15, 16, 18 between an authorized application server 12 or other authorized user and the physical disk 20. The network 10 also has inherent data loss risks. In the network 30, physical data storage can be time consuming, and physical backup tapes can be subject to failure, damage, or theft.

To overcome some of these disadvantages, systems have been introduced which duplicate and/or separate files and directories for storage across one or more physical disks. The files and directories are typically stored or backed up as a monolith, meaning that the files are logically grouped with other like data before being secured. Although this provides a convenient arrangement for retrieval, in that a common security construct (e.g. an encryption key or password) is related to all of the data, it also provides additional risk exposure if the data is compromised.

For these and other reasons, improvements are desirable.

Summary

In accordance with the following disclosure, the above and other problems are solved by the following:

In a first aspect, a method of managing data blocks in a secure storage appliance is disclosed. The method includes receiving a block of data associated with a volume, the volume associated with a plurality of shares stored on

a plurality of physical storage devices, and storing the block of data in a buffer. The method also includes associating the block of data with a state from among a plurality of states, each of the states corresponding to a status of the block of data. The method further includes processing the block of data by performing at least one cryptographic operation on the block of data, and upon completion of processing the block of data, updating the state of the block of data.

In a second aspect, a method of managing I/O requests in a secure storage appliance is disclosed. The method includes receiving an I/O request at the secure storage appliance, the I/O request associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices. The method further includes determining whether a block of data referenced by the I/O request is present in a buffer. The method also includes transferring the block of data to a buffer and associating the block of data with a transfer state, determining whether the block of data in the buffer is updated, and processing the block of data.

In a third aspect, a secure storage appliance is disclosed. The secure storage appliance includes a plurality of buffers, each buffer capable of holding a block of data having a state from among a plurality of states. The secure storage appliance also includes a programmable circuit capable of accessing the plurality of buffers. The programmable circuit is configured to execute program instructions to receive an I/O request, the I/O request associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices communicatively connected to the secure storage appliance. The programmable circuit is further configured to execute program instructions to determine whether a block of data referenced by the I/O request is present in a buffer, transfer the block

of data to a buffer among the plurality of buffers, determine whether the block of data in the buffer is updated, and process the block of data.

In a fourth aspect, a secure storage appliance is disclosed. The secure storage appliance includes a plurality of buffers, each buffer capable of holding a block of data having a state selected from among a plurality of states. The secure storage appliance further includes a programmable circuit capable of accessing the plurality of buffers. The programmable circuit is configured to execute program instructions to receive a block of data associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices. The programmable circuit is also configured to execute program instructions to store the block of data in a buffer from among the plurality of buffers and associate the buffer with a state corresponding to a status of the block of data. The programmable circuit is further configured to execute program instructions to process the block of data by performing at least one cryptographic operation on the block of data and update the state of the buffer upon completion of processing the block of data.

Brief Description of the Drawings

Figure 1 illustrates an example prior art network providing data storage;

Figure 2 illustrates an example prior art network providing data backup capabilities;

Figure 3 illustrates a data storage system according to a possible embodiment of the present disclosure;

Figure 4 illustrates a data storage system according to a further possible embodiment of the present disclosure;

Figure 5 illustrates a portion of a data storage system including a secure storage appliance, according to a possible embodiment of the present disclosure;

Figure 6 illustrates a block diagram of logical components of a secure storage appliance, according to a possible embodiment of the present disclosure.

Figure 7 illustrates a portion of a data storage system including a secure storage appliance, according to a further possible embodiment of the present disclosure;

Figure 8 illustrates dataflow of a write operation according to a possible embodiment of the present disclosure;

Figure 9 illustrates dataflow of a read operation according to a possible embodiment of the present disclosure;

Figure 10 illustrates a further possible embodiment of a data storage network including redundant secure storage appliances, according to a possible embodiment of the present disclosure;

Figure 11 illustrates incorporation of secure storage appliances in a portion of a data storage network, according to a possible embodiment of the present disclosure;

Figure 12 illustrates an arrangement of a data storage network according to a possible embodiment of the present disclosure;

Figure 13 illustrates a physical block structure of data to be written onto a physical storage device, according to aspects of the present disclosure;

Figure 14 shows a flowchart of systems and methods for providing access to secure storage in a storage area network according to a possible embodiment of the present disclosure;

Figure 15 shows a flowchart of systems and methods for reading block-level secured data according to a possible embodiment of the present disclosure;

Figure 16 shows a flowchart of systems and methods for writing block-level secured data according to a possible embodiment of the present disclosure;

Figure 17 shows a possible arrangement for providing secure storage data backup, according to a possible embodiment of the present disclosure;

Figure 18 shows a possible arrangement for providing secure storage for a thin client computing network, according to a possible embodiment of the present disclosure;

Figure 19 shows a state diagram for simultaneous state-based cryptographic splitting in a secure storage appliance, according to aspects of the present disclosure;

Figure 20 shows a flowchart of methods and systems for simultaneous state-based cryptographic splitting in a secure storage appliance, according to aspects of the present disclosure;

Figure 21 shows a flowchart of methods and systems for reconstituting data in a secure storage appliance, according to a possible aspect of the present disclosure;

Figure 22 shows a flowchart of methods and systems for cryptographically splitting data in a secure storage appliance, according to a possible aspect of the present disclosure;

Figure 23 shows a flowchart of methods and systems for managing data blocks in a secure storage appliance, according to a possible embodiment of the present disclosure; and

Figure 24 shows a flowchart of methods and systems for managing I/O requests in a secure storage appliance, according to a possible embodiment of the present disclosure.

Detailed Description

Various embodiments of the present invention will be described in detail with reference to the drawings, wherein like reference numerals represent like parts and assemblies throughout the several views. Reference to various embodiments does not limit the scope of the invention, which is limited only by the scope of the claims attached hereto. Additionally, any examples set forth in this specification are not intended to be limiting and merely set forth some of the many possible embodiments for the claimed invention.

The logical operations of the various embodiments of the disclosure described herein are implemented as: (1) a sequence of computer implemented steps, operations, or procedures running on a programmable circuit within a computer, and/or (2) a sequence of computer implemented steps, operations, or procedures running on a programmable circuit within a directory system, database, or compiler.

In general the present disclosure relates to a block-level data storage security system. By block-level, it is intended that the data storage and security performed according to the present disclosure is not performed based on the size or arrangement of logical files (e.g. on a per-file or per-directory level), but rather that the data security is based on individual read and write operations related to physical

blocks of data. In various embodiments of the present disclosure, the data managed by the read and write operations are split or grouped on a bitwise or other physical storage level. These physical storage portions of files can be stored in a number of separated components and encrypted. The split, encrypted data improves data security for the data "at rest" on the physical disks, regardless of the access vulnerabilities of physical disks storing the data. This is at least in part because the data cannot be recognizably reconstituted without having appropriate access and decryption rights to multiple, distributed disks. The access rights limitations provided by such a system also makes deletion of data simple, in that deletion of access rights (e.g. encryption keys) provides for effective deletion of all data related to those rights.

The various embodiments of the present disclosure are applicable across a number of possible networks and network configurations; in certain embodiments, the block-level data storage security system can be implemented within a storage area network (SAN) or Network-Attached Storage (NAS) system. Other possible networks in which such systems can be implemented exist as well.

In certain aspects of the present disclosure, simultaneous state-based processing of cryptographic splitting and reconstituting operations are provided in a secure storage appliance. This state-based, pipelined processing allows the secure storage appliance to perform sub-tasks as resources of the secure storage appliance become available for use. The secure storage appliance can therefore improve throughput of processed data and I/O requests related to that data, based on these operations.

Referring now to Figure 3, a block diagram illustrating an example data storage system 100 is shown, according to the principles of the present

disclosure. In the example of Figure 3, system 100 includes a set of client devices 105A through 105N (collectively, "client devices 105"). Client devices 105 can be a wide variety of different types of devices. For example, client devices 105 can be personal computers, laptop computers, network telephones, mobile telephones, television set top boxes, network televisions, video gaming consoles, web kiosks, devices integrated into vehicles, mainframe computers, personal media players, intermediate network devices, network appliances, and other types of computing devices. Client devices 105 may or may not be used directly by human users.

Client devices 105 are connected to a network 110. Network 110 facilitates communication among electronic devices connected to network 110. Network 110 can be a wide variety of electronic communication networks. For example, network 110 can be a local-area network, a wide-area network (e.g., the Internet), an extranet, or another type of communication network. Network 110 can include a variety of connections, including wired and wireless connections. A variety of communications protocols can be used on network 110 including Ethernet, WiFi, WiMax, Transfer Control Protocol, and many other communications protocols.

In addition, system 100 includes an application server 115. Application server 115 is connected to the network 110, which is able to facilitate communication between the client devices 105 and the application server 115. The application server 115 provides a service to the client devices 105 via network 110. For example, the application server 115 can provide a web application to the client devices 105. In another example, the application server 115 can provide a network-attached storage server to the client devices 105. In another example, the application

server 115 can provide a database access service to the client devices 105. Other possibilities exist as well.

The application server 115 can be implemented in several ways. For example, the application server 115 can be implemented as a standalone server device, as a server blade, as an intermediate network device, as a mainframe computing device, as a network appliance, or as another type of computing device. Furthermore, it should be appreciated that the application server 115 can include a plurality of separate computing devices that operate like one computing device. For instance, the application server 115 can include an array of server blades, a network data center, or another set of separate computing devices that operate as if one computing device. In certain instances, the application server can be a virtualized application server associated with a particular group of users, as described in greater detail below in Figure 18.

The application server 115 is communicatively connected to a secure storage appliance 120 that is integrated in a storage area network (SAN) 125. Further, the secure storage appliance 120 is communicatively connected to a plurality of storage devices 130A through 130N (collectively, "storage devices 130"). Similar to the secure storage appliance 120, the storage devices 130 can be integrated with the SAN 125.

The secure storage appliance 120 can be implemented in several ways. For example, the secure storage appliance 120 can be implemented as a standalone server device, as a server blade, as an intermediate network device, as a mainframe computing device, as a network appliance, or as another type of computing device. Furthermore, it should be appreciated that, like the application server 115, the secure storage appliance 120 can include a plurality of separate

computing devices that operate like one computing device. In certain embodiments, SAN 125 may include a plurality of secure storage appliances. Each of secure storage appliances 214 is communicatively connected to a plurality of the storage devices 130. In addition, it should be appreciated that the secure storage appliance 120 can be implemented on the same physical computing device as the application server 115.

The application server 115 can be communicatively connected to the secure storage appliance 120 in a variety of ways. For example, the application server 115 can be communicatively connected to the secure storage appliance 120 such that the application server 115 explicitly sends I/O commands to secure storage appliance 120. In another example, the application server 115 can be communicatively connected to secure storage appliance 120 such that the secure storage appliance 120 transparently intercepts I/O commands sent by the application server 115. On a physical level, the application server 115 and the secure storage appliance 120 can be connected via most physical interfaces that support a SCSI command set. Examples of such interfaces include Fibre Channel and iSCSI interfaces.

The storage devices 130 can be implemented in a variety of different ways as well. For example, one or more of the storage devices 130 can be implemented as disk arrays, tape drives, JBODs ("just a bunch of disks"), or other types of electronic data storage devices.

In various embodiments, the SAN 125 is implemented in a variety of ways. For example, the SAN 125 can be a local-area network, a wide-area network (e.g., the Internet), an extranet, or another type of electronic communication network. The SAN 125 can include a variety of connections, including wired and

wireless connections. A variety of communications protocols can be used on the SAN 125 including Ethernet, WiFi, WiMax, Transfer Control Protocol, and many other communications protocols. In certain embodiments, the SAN 125 is a high-bandwidth data network provided using, at least in part, an optical communication network employing Fibre Channel connections and Fibre Channel Protocol (FCP) data communications protocol between ports of data storage computing systems.

The SAN 125 additionally includes an administrator device 135. The administrator device 135 is communicatively connected to the secure storage appliance 120 and optionally to the storage devices 130. The administrator device 135 facilitates administrative management of the secure storage appliance 120 and to storage devices. For example, the administrator device 135 can provide an application that can transfer configuration information to the secure storage appliance 120 and the storage devices 130. In another example, the administrator device 135 can provide a directory service used to store information about the SAN 125 resources and also centralize the SAN 125.

In various embodiments, the administrator device 135 can be implemented in several ways. For example, the administrator device 135 can be implemented as a standalone computing device such as a PC or a laptop, or as another type of computing device. Furthermore, it should be appreciated that, like the secure storage appliance 120, the administrator device 135 can include a plurality of separate computing devices that operate as one computing device.

Now referring to Figure 4, a data storage system 200 is shown according to a possible embodiment of the present disclosure. The data storage system 200 provides additional security by way of introduction of a secure storage

appliance and related infrastructure/functionality into the data storage system 200, as described in the generalized example of Figure 3.

In the embodiment shown, the data storage system 200 includes an application server 202, upon which a number of files and databases are stored. The application server 202 is generally one or more computing devices capable of connecting to a communication network and providing data and/or application services to one or more users (e.g. in a client-server, thin client, or local account model). The application server 202 is connected to a plurality of storage systems 204. In the embodiment shown, storage systems $204_{1\text{-}5}$ are shown, and are illustrated as a variety of types of systems including direct local storage, as well as hosted remote storage. Each of storage systems 204 manages storage on one or more physical storage devices 206. The physical storage devices 206 generally correspond to hard disks or other long-term data storage devices. In the specific embodiment shown, the JBOD storage system 204_1 connects to physical storage devices 206_1, the NAS storage system 204_2 connects to physical storage device 206_2, the JBOD storage system 204_3 connects to physical storage devices $206_{3\text{-}7}$, the storage system 204_4 connects to physical storage devices $206_{8\text{-}12}$, and the JBOD storage system 204_5 connects to physical storage device 206_{13}. Other arrangements are possible as well, and are in general a matter of design choice.

In the embodiment shown, a plurality of different networks and communicative connections reside between the application server 202 and the storage systems 204. For example, the application server 202 is directly connected to JBOD storage system 204_1 via a plurality of physical storage devices 208 (JBOD connection), e.g. for local storage. The application server 202 is also communicatively connected to storage systems $204_{2\text{-}3}$ via network 210, which uses

any of a number of IP-based protocols such as Ethernet, WiFi, WiMax, Transfer Control Protocol, or any other of a number of communications protocols. The application server 202 also connects to storage systems 204_{4-5} via a storage area network (SAN) 212, which can be any of a number of types of SAN networks described in conjunction with SAN 125, above.

A secure storage appliance 120 is connected between the application server 202 and a plurality of the storage systems 204. The secure storage appliance 120 can connect to dedicated storage systems (e.g. the JBOD storage system 204_5 in Figure 4), or to storage systems connected both directly through the SAN 212, and via the secure storage appliance 120 (e.g. the JBOD storage system 204_3 and storage system 204_4). Additionally, the secure storage appliance 120 can connect to systems connected via the network 210 (e.g. the JBOD storage system 204_3). Other arrangements are possible as well. In instances where the secure storage appliance 120 is connected to one of storage systems 204, one or more of the physical storage devices 206 managed by the corresponding system is secured by way of data processing by the secure storage appliance. In the embodiment shown, the physical storage devices 206_{3-7}, 206_{10-13} are secured physical storage devices, meaning that these devices contain data managed by the secure storage appliance 120, as explained in further detail below.

Generally, inclusion of the secure storage appliance 120 within the data storage system 200 may provide improved data security for data stored on the physical storage devices. As is explained below, this can be accomplished, for example, by cryptographically splitting the data to be stored on the physical devices, such that generally each device contains only a portion of the data required to

reconstruct the originally stored data, and that portion of the data is a block-level portion of the data encrypted to prevent reconstitution by unauthorized users.

Through use of the secure storage appliance 120 within the data storage system 200, a plurality of physical storage devices 208 can be mapped to a single volume, and that volume can be presented as a virtual disk for use by one or more groups of users. In comparing the example data storage system 200 to the prior art system shown in Figure 1, it can be seen that the secure storage appliance 120 allows a user to have an arrangement other than one-to-one correspondence between drive volume letters (in Figure 1, drive letters I-M) and physical storage devices. In the embodiment shown, two additional volumes are exposed to the application server 202, virtual disk drives T and U, in which secure copies of data can be stored. Virtual disk having volume label T is illustrated as containing secured volumes F3 and F7 (i.e. the drives mapped to the iSCSI2 port of the application server 202, as well as a new drive), thereby providing a secured copy of information on either of those drives for access by a group of users. Virtual disk having volume label U provides a secured copy of the data held in DB1 (i.e. the drive mapped to LUN03). By distributing volumes across multiple disks, security is enhanced because copying or stealing data from a single physical disk will generally be insufficient to access that data (i.e. multiple disks of data, as well as separately-held encryption keys, must be acquired)

Referring now to Figure 5, a portion of the data storage system 200 is shown, including details of the secure storage appliance 120. In the embodiment shown, the secure storage appliance 120 includes a number of functional modules that generally allow the secure storage appliance to map a number of physical disks to one or more separate, accessible volumes that can be made available to a client,

and presenting a virtual disk to clients based on those defined volumes. Transparently to the user, the secure storage appliance applies a number of techniques to stored and retrieved data to provide data security.

In the embodiment shown, the secure storage appliance 120 includes a core functional unit 216, a LUN mapping unit 218, and a storage subsystem interface 220. The core functional unit 216 includes a data conversion module 222 that operates on data written to physical storage devices 206 and retrieved from the physical storage devices 206. In general, when the data conversion module 222 receives a logical unit of data (e.g. a file or directory) to be written to physical storage devices 206, it splits that primary data block at a physical level (i.e. a "block level") and encrypts the secondary data blocks using a number of encryption keys.

The manner of splitting the primary data block, and the number of physical blocks produced, is dictated by additional control logic within the core functional unit 216. As described in further detail below, during a write operation that writes a primary data block to physical storage (e.g. from an application server 202), the core functional unit 216 directs the data conversion module 222 to split the primary data block received from the application server 202 into N separate secondary data blocks. Each of the N secondary data blocks is intended to be written to a different one of physical storage devices 206 within the data storage system 200. The core functional unit 216 also dictates to the data conversion module 222 the number of shares (for example, denoted as M of the N total shares) that are required to reconstitute the primary data block when requested by the application server 202.

The secure storage appliance 120 connects to a metadata store 224, which is configured to hold metadata information about the locations, redundancy,

and encryption of the data stored on the physical storage devices 206. The metadata store 224 is generally held locally or in proximity to the secure storage appliance 120, to ensure fast access of metadata regarding the shares. The metadata store 224 can be, in various embodiments, a database or file system storage of data describing the data connections, locations, and shares used by the secure storage appliance. Additional details regarding the specific metadata stored in the metadata store 224 are described below.

The LUN mapping unit 218 generally provides a mapping of one or more physical storage devices 206 to a volume. Each volume corresponds to a specific collection of physical storage devices 206 upon which the data received from client devices is stored. In contrast, typical prior art systems assign a LUN (logical unit number) or other identifier to each physical storage device or connection port to such a device, such that data read operations and data write operations directed to one of storage systems 204 can be performed specific to a device associated with the system. In the embodiment shown, the LUNs correspond to target addressable locations on the secure storage appliance 120, of which one or more is exposed to a client device, such as an application server 202. Based on the mapping of LUNs to a volume, the virtual disk related to that volume appears as a directly-addressable component of the data storage system 200, having its own LUN. From the perspective of the application server 202, this obscures the fact that primary data blocks written to a volume can in fact be split, encrypted, and written to a plurality of physical storage devices across one or more storage systems 204.

The storage subsystem interface 220 routes data from the core functional unit 216 to the storage systems 204 communicatively connected to the secure storage appliance 120. The storage subsystem interface 220 allows

addressing various types of storage systems 204. Other functionality can be included as well.

In the embodiment shown, a plurality of LUNs are made available by the LUN mapping unit 218, for addressing by client devices. As shown by way of example, LUNs LUN04-LUNnn are illustrated as being addressable by client devices. Within the core functional unit 216, the data conversion module 222 associates data written to each LUN with a share of that data, split into N shares and encrypted. In the embodiment shown in the example of Fig. 5, a block read operation or block write operation to LUN04 is illustrated as being associated with a four-way write, in which secondary data blocks L04.a through L04.d are created, and mapped to various devices connected to output ports, shown in Figure 5 as network interface cards (NICs), a Fibre Channel interface, and a serial ATA interface. An analogous operation is also shown with respect to LUN05, but written to a different combination of shares and corresponding physical disks.

The core functional unit 216, LUN mapping unit 218, and storage subsystem interface 220 can include additional functionality as well, for managing timing and efficiency of data read and write operations. Additional details regarding this functionality are described in another embodiment, detailed below in conjunction with the secure storage appliance functionality described in Figure 6.

The secure storage appliance 120 includes an administration interface 226 that allows an administrator to set up components of the secure storage appliance 120 and to otherwise manage data encryption, splitting, and redundancy. The administration interface 226 handles initialization and discovery on the secure storage appliance, as well as creation, modifying, and deletion of individual volumes and virtual disks; event handling; data base administration; and other system

services (such as logging). Additional details regarding usage of the administration interface 226 are described below in conjunction with Figure 14.

In the embodiment shown of the secure storage appliance 120, the secure storage appliance 120 connects to an optional enterprise directory 228 and a key manager 230 via the administration interface 226. The enterprise directory 228 is generally a central repository for information about the state of the secure storage appliance 120, and can be used to help coordinate use of multiple secure storage appliances in a network, as illustrated in the configuration shown in Figure 10, below. The enterprise directory 228 can store, in various embodiments, information including a remote user table, a virtual disk table, a metadata table, a device table, log and audit files, administrator accounts, and other secure storage appliance status information.

In embodiments lacking the enterprise directory 228, redundant secure storage appliances 214 can manage and prevent failures by storing status information of other secure storage appliances, to ensure that each appliance is aware of the current state of the other appliances.

The key manager 230 stores and manages certain keys used by the data storage system 200 for encrypting data specific to various physical storage locations and various individuals and groups accessing those devices. In certain embodiments, the key manager 230 stores workgroup keys. Each workgroup key relates to a specific community of individuals (i.e. a "community of interest") and a specific volume, thereby defining a virtual disk for that community. The key manager 230 can also store local copies of session keys for access by the secure storage appliance 120. Secure storage appliance 120 uses each of the session keys to locally encrypt data on different ones of physical storage devices 206. Passwords

can be stored at the key manager 230 as well. In certain embodiments, the key manager 230 is operable on a computing system configured to execute any of a number of key management software packages, such as the Key Management Service provided for a Windows Server environment, manufactured by Microsoft Corp. of Redmond, Washington.

Although the present disclosure provides for encryption keys including session keys and workgroup keys, additional keys may be used as well, such as a disk signature key, security group key, client key, or other types of keys. Each of these keys can be stored on one or more of physical storage devices 206, at the secure storage appliance 120, or in the key manager 230.

Although Figures 4-5 illustrate a particular arrangement of a data storage system 200 for secure storage of data, additional arrangements are possible as well that can operate consistently with the concepts of the present disclosure. For example, in certain embodiments, the system can include a different number or type of storage systems or physical storage devices, and can include one or more different types of client systems in place of or in addition to the application server 202. Furthermore, the secure storage appliance 120 can be placed in any of a number of different types of networks, but does not require the presence of multiple types of networks as illustrated in the example of Figure 4.

Figure 6 is a block diagram that illustrates example logical components of the secure storage appliance 120. Figure 6 represents only one example of the logical components of the secure storage appliance 120, for performing the operations described herein. The operations of the secure storage appliance 120 can be conceptualized and implemented in many different ways.

As illustrated in the example of Figure 6, the secure storage appliance 120 comprises a primary interface 300 and a secondary interface 302. The primary interface 300 enables secure storage appliance 120 to receive primary I/O requests and to send primary I/O responses. For instance, the primary interface 300 can enable secure storage appliance 120 to receive primary I/O requests (e.g. read and write requests) from the application server device 202 and to send primary I/O responses to the application server 202. Secondary interface enables the secure storage appliance 120 to send secondary I/O requests to the storage systems 204, and to receive secondary I/O responses from those storage systems 204.

In addition, the secure storage appliance 120 comprises a parser driver 304. The parser driver 304 generally corresponds to the data conversion module 222 of Figure 5, in that it processes primary I/O requests to generate secondary I/O requests and processes secondary I/O responses to generate primary I/O responses. To accomplish this, the parser driver 304 comprises a read module 305 that processes primary read requests to generate secondary read requests and processes secondary read responses to generate primary read responses. In addition, the parser driver 304 comprises a decryption module 308 that enables the read module 305 to reconstruct a primary data block using secondary blocks contained in secondary read responses. Example operations performed by the read module 305 are described below with reference to Figs. 15, 22, and 24. Furthermore, the parser driver 304 comprises a write module 306 that processes primary write requests to generate secondary write requests and processes secondary write responses to generate primary write responses. The parser driver 304 also comprises an encryption module 310 that enables the write module 306 to cryptographically split primary data blocks in primary write requests into secondary data blocks to put in

secondary write requests. An example operation performed by the write module 306 is described below as well with reference to Figs. 16, 23, and 25.

In the example of Figure 6, the secure storage appliance 120 also comprises a cache driver 315. When enabled, the cache driver 315 receives primary I/O requests received by the primary interface 300 before the primary I/O requests are received by parser driver 304. When the cache driver 315 receives a primary read request to read data at a primary storage location of a virtual disk, the cache driver 315 determines whether a write-through cache 316 at the secure storage appliance 120 contains a primary write request to write a primary data block to the primary storage location of the virtual disk. If the cache driver 315 determines that the write-through cache 316 contains a primary write request to write a primary data block to the primary storage location of the virtual disk, the cache driver 315 outputs a primary read response that contains the primary data block. When the parser driver 304 receives a primary write request to write a primary data block to a primary storage location of a virtual disk, the cache driver 315 caches the primary write request in the write-through cache 316. A write-through module 318 performs write operations to memory from the write-through cache 316.

The secure storage appliance 120 also includes an outstanding write list (OWL) module 326. When enabled, the OWL module 326 receives primary I/O requests from the primary interface 300 before the primary I/O requests are received by the parser driver 304. The OWL module 326 uses an outstanding write list 320 to process the primary I/O requests.

In addition, the secure storage appliance 120 comprises a backup module 324. The backup module 324 performs an operation that backs up data at

the storage systems 204 to backup devices, as described below in conjunction with Figures 17-18.

The secure storage appliance 120 also comprises a configuration change module 312. The configuration change module 312 performs an operation that creates or destroys a volume, and sets its redundancy configuration. Example redundancy configurations (i.e. "M of N" configurations) are described throughout the present disclosure, and refer to the number of shares formed from a block of data, and the number of those shares required to reconstitute the block of data. Further discussion is provided with respect to possible redundancy configurations below, in conjunction with Figures 8-9.

It should be appreciated that many alternate implementations of the secure storage appliance 120 are possible. For example, a first alternate implementation of the secure storage appliance 120 can include the OWL module 326, but not the cache driver 315, or vice versa. In other examples, the secure storage appliance 120 might not include the backup module 324 or the configuration change module 312. Furthermore, there can be many alternate operations performed by the various modules of the secure storage appliance 120.

Figure 7 illustrates further details regarding connections to and operational hardware and software included in secure storage appliance 120, according to a possible embodiment of the present disclosure. The secure storage appliance 120 illustrates the various operational hardware modules available in the secure storage appliance to accomplish the data flow and software module operations described in Figures 4-6, above. In the embodiment shown, the secure storage appliance 120 is communicatively connected to a client device 402, an

administrative console 404, a key management server 406, a plurality of storage devices 408, and an additional secure storage appliance 120'.

In the embodiment shown, the secure storage appliance 120 connects to the client device 402 via both an IP network connection 401 and a SAN network connection 403. The secure storage appliance 120 connects to the administrative console 404 by one or more IP connections 405 as well. The key management server 406 is also connected to the secure storage appliance 120 by an IP network connection 407. The storage devices 408 are connected to the secure storage appliance 120 by the SAN network connection 403, such as a Fibre Channel or other high-bandwidth data connection. Finally, in the embodiment shown, secure storage appliances 120 and 120' are connected via any of a number of types of communicative connections 411, such as an IP or other connection, for communicating heartbeat messages and status information for coordinating actions of the secure storage appliance 120 and the secure storage appliance 120'. Although in the embodiment shown, these specific connections and systems are included, the arrangement of devices connected to the secure storage appliance 120, as well as the types and numbers of devices connected to the appliance may be different in other embodiments.

The secure storage appliance 120 includes a number of software-based components, including a management service 410 and a system management module 412. The management service 410 and the system management module 412 each connect to the administrative console 404 or otherwise provide system management functionality for the secure storage appliance 120. The management service 410 and system management module 412 are generally used to set various settings in the secure storage appliance 120, view logs 414 stored on the appliance,

and configure other aspects of a network including the secure storage appliance 120. Additionally, the management service 410 connects to the key management server 406, and can request and receive keys from the key management server 406 as needed.

A cluster service 416 provides synchronization of state information between the secure storage appliance 120 and secure storage appliance 120'. In certain embodiments, the cluster service 416 manages a heartbeat message and status information exchanged between the secure storage appliance 120 and the secure storage appliance 120'. Secure storage appliance 120 and secure storage appliance 120' periodically exchange heartbeat messages to ensure that secure storage appliance 120 and secure storage appliance 120' maintain contact. Secure storage appliance 120 and secure storage appliance 120' maintain contact to ensure that the state information received by each secure storage appliance indicating the state of the other secure storage appliance is up to date. An active directory services 418 stores the status information, and provides status information periodically to other secure storage appliances via the communicative connections 411.

Additional hardware and/or software components provide datapath functionality to the secure storage appliance 120 to allow receipt of data and storage of data at the storage devices 408. In the embodiment shown, the secure storage appliance 120 includes a SNMP connection module 420 that enables secure storage appliance 120 to communicate with client devices via the IP network connection 401, as well as one or more high-bandwidth data connection modules, such as a Fibre Channel input module 422 or SCSI input module 424 for receiving data from the client device 402 or storage devices 408. Analogous data output modules including a Fibre Channel connection module 421 or SCSI connection module 423

can connect to the storage devices 408 or client device 402 via the SAN network connection 403 for output of data.

Additional functional systems within the secure storage appliance 120 assist in datapath operations. A SCSI command module 425 parses and forms commands to be sent out or received from the client device 402 and storage devices 408. A multipath communications module 426 provides a generalized communications interface for the secure storage appliance 120, and a disk volume 428, disk 429, and cache 316 provide local data storage for the secure storage appliance 120.

Additional functional components can be included in the secure storage appliance 120 as well. In the embodiment shown, a parser driver 304 provides data splitting and encryption capabilities for the secure storage appliance 120, as previously explained. A provider 434 includes volume management information, for creation and destruction of volumes. An events module 436 generates and handles events based on observed occurrences at the secure storage appliance (e.g. data errors or communications errors with other systems).

Figures 8-9 provide a top level sense of a dataflow occurring during write and read operations, respectively, passing through a secure storage appliance, such as the secure storage appliance described above in conjunction with Figures 3-7. Figure 8 illustrates a dataflow of a write operation according to a possible embodiment of the present disclosure, while Figure 9 illustrates dataflow of a read operation. In the write operation of Figure 8, a primary data block 450 is transmitted to a secure storage appliance (e.g. from a client device such as an application server). The secure storage appliance can include a functional block 460 to separate the primary data block into N secondary data blocks 470, shown as S-1 through S-N.

In certain embodiments, the functional block 460 is included in a parser driver, such as parser driver 304, above. The specific number of secondary data blocks can vary in different networks, and can be defined by an administrative user having access to control settings relevant to the secure storage appliance. Each of the secondary data blocks 470 can be written to separate physical storage devices. In the read operation of Figure 9, M secondary data blocks are accessed from physical storage devices, and provided to the functional block 460 (e.g. parser driver 304). The functional block 460 then performs an operation inverse to that illustrated in Figure 8, thereby reconstituting the primary data block 450. The primary data block can then be provided to the requesting device (e.g. a client device).

In each of Figures 8-9, the N secondary data blocks 470 each represent a cryptographically split portion of the primary data block 450, such that the functional block 460 requires only M of the N secondary data blocks (where M <= N) to reconstitute the primary data block 450. The cryptographic splitting and data reconstitution of Figures 8-9 can be performed according to any of a number of techniques. In one embodiment, the parser driver 304 executes SecureParser software provided by Security First Corporation of Rancho Santa Margarita, California.

Although, in the embodiment shown in Figure 9, the parser driver 304 uses the N secondary data blocks 470 to reconstitute the primary data block 450, it is understood that in certain applications, fewer than all of the N secondary data blocks 470 are required. For example, when the parser driver 304 generates N secondary data blocks during a write operation such that only M secondary data blocks are required to reconstitute the primary data block (where M < N), then data

conversion module 60 only needs to read that subset of secondary data block from physical storage devices to reconstitute the primary data block 450.

For example, during operation of the parser driver 304 a data conversion routine may generate four secondary data blocks 470, of which two are needed to reconstitute a primary data block (i.e. M = 2, N = 4). In such an instance, two of the secondary data blocks 470 may be stored locally, and two of the secondary data blocks 470 may be stored remotely to ensure that, upon failure of a device or catastrophic event at one location, the primary data block 450 can be recovered by accessing one or both of the secondary data blocks 470 stored remotely. Other arrangements are possible as well, such as one in which four secondary data blocks 470 are stored locally and all are required to reconstitute the primary data block 450 (i.e. M = 4, N = 4). At its simplest, a single share could be created (M = N = 1).

Figure 10 illustrates a further possible embodiment of a data storage system 250, according to a possible embodiment of the present disclosure. The data storage system 250 generally corresponds to the data storage system 200 of Figure 4, above, but further includes redundant secure storage appliances 214. Each of secure storage appliances 214 may be an instance of secure storage appliance 120. Inclusion of redundant secure storage appliances 214 allows for load balancing of read and write requests in the data storage system 250, such that a single secure storage appliance is not required to process every secure primary read command or primary write command passed from the application server 202 to one of the secure storage appliances 214. Use of redundant secure storage appliances also allows for failsafe operation of the data storage system 250, by ensuring that requests made of a failed secure storage appliance are rerouted to alternative secure storage appliances.

In the embodiment of the data storage system 250 shown, two secure storage appliances 214 are shown. Each of the secure storage appliances 214 can be connected to any of a number of clients (e.g. the application server 202), as well as secured storage systems 204, the metadata store 224, and a remote server 252. In various embodiments, the remote server 252 could be, for example, an enterprise directory 228 and/or a key manager 230.

The secure storage appliances 214 are also typically connected to each other via a network connection. In the embodiment shown in the example of Fig. 10, the secure storage appliances 214 reside within a network 254. In various embodiments, network 254 can be, for example, an IP-based network, SAN as previously described in conjunction with Figures 4-5, or another type of network. In certain embodiments, the network 254 can include aspects of one or both types of networks. An example of a particular configuration of such a network is described below in conjunction with Figures 11-12.

The secure storage appliances 214 in the data storage system 250 are connected to each other across a TCP/IP portion of the network 254. This allows for the sharing of configuration data, and the monitoring of state, between the secure storage appliances 214. In certain embodiments there can be two IP-based networks, one for sharing of heartbeat information for resiliency, and a second for configuration and administrative use. The secure storage appliance 120 can also potentially be able to access the storage systems 204, including remote storage systems, across an IP network using a data interface.

In operation, sharing of configuration data, state data, and heartbeat information between the secure storage appliances 214 allows the secure storage appliances 214 to monitor and determine whether other secure storage appliances are

present within the data storage system 250. Each of the secure storage appliances 214 can be assigned specific addresses of read operations and write operations to process. Secure storage appliances 214 can reroute received I/O commands to the appropriate one of the secure storage appliances 214 assigned that operation based upon the availability of that secure storage appliance and the resources available to the appliance. Furthermore, the secure storage appliances 214 can avoid addressing a common storage device 204 or application server 202 port at the same time, thereby avoiding conflicts. The secure storage appliances 214 also avoid reading from and writing to the same share concurrently to prevent the possibility of reading stale data.

When one of the secure storage appliances 214 fails, a second secure storage appliance can determine the state of the failed secure storage appliance based upon tracked configuration data (e.g. data tracked locally or stored at the remote server 252). The remaining operational one of the secure storage appliances 214 can also access information in the metadata store 224, including share and key information defining volumes, virtual disks and client access rights, to either process or reroute requests assigned to the failed device.

As previously described, the data storage system 250 is intended to be exemplary of a possible network in which aspects of the present disclosure can be implemented; other arrangements are possible as well, using different types of networks, systems, storage devices, and other components.

Referring now to Figure 11, one possibility of a methodology of incorporating secure storage appliances into a data storage network, such as a SAN, is shown according to a possible embodiment of the present disclosure. In the embodiment shown, a secure storage network 500 provides for fully redundant

storage, in that each of the storage systems connected at a client side of the network is replicated in mass storage, and each component of the network (switches, secure storage appliances) is located in a redundant array of systems, thereby providing a failsafe in case of component failure. In alternative embodiments, the secure storage network 500 can be simplified by including only a single switch and/or single secure storage appliance, thereby reducing the cost and complexity of the network (while coincidentally reducing the protection from component failure).

In the embodiment shown, an overall secure storage network 500 includes a plurality of data lines 502a-d interconnected by switches 504a-b. Data lines 502a-b connect to storage systems 506a-c, which connect to physical storage disks 508a-f. The storage systems 506a-c correspond generally to smaller-scale storage servers, such as an application server, client device, or other system as previously described. In the embodiment shown in the example of Fig. 11, storage system 506a connects to physical storage disks 508a-b, storage system 506b connects to physical storage disks 508c-d, and storage system 506c connects to physical storage disks 508e-f. The secure storage network 500 can be implemented in a number of different ways, such as through use of Fibre Channel or iSCSI communications as the data lines 502a-d, ports, and other data communications channels. Other high bandwidth communicative connections can be used as well.

The switches 504a-b connect to a large-scale storage system, such as the mass storage 510 via the data lines 502c-d. The mass storage 510 includes, in the embodiment shown, two data directors 512a-b, which respectively direct data storage and requests for data to one or more of the back end physical storage devices 514a-d. In the embodiment shown, the physical storage devices 514a-c are

unsecured (i.e. not cryptographically split and encrypted), while the physical storage device 514d stores secure data (i.e. password secured or other arrangement).

The secure storage appliances 516a-b also connect to the data lines 502a-d, and each connect to the secure physical storage devices 518a-e. Additionally, the secure storage appliances 516a-b connect to the physical storage devices 520a-c, which can reside at a remote storage location (e.g. the location of the large-scale storage system mass storage 510).

In certain embodiments providing redundant storage locations, the secure storage network 500 allows a user to configure the secure storage appliances 516a-b such that, using the M of N cryptographic splitting enabled in each of the secure storage appliances 516a-b, M shares of data can be stored on physical storage devices at a local location to provide fast retrieval of data, while another M shares of data can be stored on remote physical storage devices at a remote location. Therefore, failure of one or more physical disks or secure storage appliances does not render data unrecoverable, because a sufficient number of shares of data remain accessible to at least one secure storage appliance capable of reconstituting requested data.

Figure 12 illustrates a particular cluster-based arrangement of a data storage network 600 according to a possible embodiment of the present disclosure. The data storage network 600 is generally arranged such that clustered secure storage appliances access and store shares on clustered physical storage devices, thereby ensuring fast local storage and access to the cryptographically split data. The data storage network 600 is therefore a particular arrangement of the networks and systems described above in Figures 1-11, in that it represents an arrangement in which physical proximity of devices is accounted for.

In the embodiment shown, the data storage network 600 includes two clusters, 602a-b. Each of the clusters 602a-b includes a pair of secure storage appliances 604a-b, respectively. In the embodiment shown, the clusters 602a-b are labeled as clusters A and B, respectively, with each cluster including two secure storage appliances 604a-b (shown as appliances A1 and A2 in cluster 602a, and appliances B1 and B2 in cluster 602b, respectively). The secure storage appliances 604a-b within each of the clusters 602a-b are connected via a data network 605 (e.g. via switches or other data connections in an iSCSI, Fibre Channel, or other data network, as described above and indicated via the nodes and connecting lines shown within the data network 605) to a plurality of physical storage devices 610. Additionally, the secure storage appliances 604a-b are connected to client devices 612, shown as client devices C1-C3, via the data network 605. The client devices 612 can be any of a number of types of devices, such as application servers, database servers, or other types of data-storing and managing client devices.

In the embodiment shown, the client devices 612 are connected to the secure storage appliances 604a-b such that each of client devices 612 can send I/O operations (e.g. a read request or a write request) to two or more of the secure storage appliances 604a-b, to ensure a backup datapath in case of a connection failure to one of secure storage appliances 604a-b. Likewise, the secure storage appliances 604a-b of each of clusters 602a-b are both connected to a common set of physical storage devices 610. Although not shown in the example of Fig. 12, the physical storage devices 610 can be, in certain embodiments, managed by separate storage systems, as described above. Such storage systems are removed from the illustration of the data storage network 600 for simplicity, but can be present in practice.

An administrative system 614 connects to a maintenance console 616 via a local area network 618. Maintenance console 616 has access to a secured domain 620 of an IP-based network 622. The maintenance console 616 uses the secured domain 620 to access and configure the secure storage appliances 604a-b. One method of configuring the secure storage appliances is described below in conjunction with Figure 14.

The maintenance console 616 is also connected to both the client devices 612 and the physical storage devices 610 via the IP-based network 622. The maintenance console 616 can determine the status of each of these devices to determine whether connectivity issues exist, or whether the device itself has become non-responsive.

Referring now to Figure 13, an example physical block structure of data written onto one or more physical storage devices is shown, according to aspects of the present disclosure. The example of Fig. 13 illustrates three strips 700A, 700B, and 700C (collectively, "shares"). Each of strips 700 is a share of a physical storage device devoted to storing data associated with a common volume. For example, in a system in which a write operation splits a primary data block into three secondary data blocks (i.e. N = 3), the strips 700 (shares) would be appropriately used to store each of the secondary data blocks. As used in this disclosure, a volume is grouped storage that is presented by a secure storage appliance to clients of secure storage appliance (e.g. secure storage appliance 120 or one of secure storage appliances 214 as previously described), such that the storage appears as a contiguous, unitary storage location. Secondary data blocks of a volume are distributed among strips 700. In systems implementing a different number of shares (e.g. N = 2, 4, 6, etc.), a different, corresponding number of shares

would be used. As basic as a 1 of 1 configuration (M = 1, N = 1) configuration could be used.

Each of the strips 700 corresponds to a reserved portion of memory of a different one of physical storage devices (e.g. physical storage devices 206 previously described), and relates to a particular I/O operation from storage or reading of data to/from the physical storage device. Typically, each of the strips 700 resides on a different one of physical storage devices. Furthermore, although three different strips are shown in the illustrative embodiment shown, more or fewer strips can be used as well. In certain embodiments, each of the strips 700 begins on a sector boundary. In other arrangements, the each of the strips 700 can begin at any other memory location convenient for management within the share.

Each of strips 700 includes a share label 704, a signature 706, header information 708, virtual disk information 710, and data blocks 712. The share label 704 is written on each of strips 700 in plain text, and identifies the volume and individual share. The share label 704 can also, in certain embodiments, contain information describing other header information for the strips 700, as well as the origin of the data written to the strip (e.g. the originating cluster).

The signature 706 contain information required to construct the volume, and is encrypted by a workgroup key. The signatures 706 contain information that can be used to identify the physical device upon which data (i.e. the share) is stored. The workgroup key corresponds to a key associated with a group of one or more users having a common set of usage rights with respect to data (i.e. all users within the group can have access to common data.) In various embodiments, the workgroup key can be assigned to a corporate department using common data, a

common group of one or more users, or some other community of interest for whom common access rights are desired.

The header information 708 contains session keys used to encrypt and decrypt the volume information included in the virtual disk information 710, described below. The header information 708 is also encrypted by the workgroup key. In certain embodiments, the header information 708 includes headers per section of data. For example, the header information 708 may include one header for each 64 GB of data. In such embodiments, it may be advantageous to include at least one empty header location to allow re-keying of the data encrypted with a preexisting session key, using a new session key.

The virtual disk information 710 includes metadata that describes a virtual disk, as it is presented by a secure storage appliance. The virtual disk information 710, in certain embodiments, includes names to present the virtual disk, a volume security descriptor, and security group information. The virtual disk information 710 can be, in certain embodiments, encrypted by a session key associated with the physical storage device upon which the strips 700 are stored, respectively.

The secondary data blocks 712 correspond to a series of memory locations used to contain the cryptographically split and encrypted data. Each of the secondary data blocks 712 contains data created at a secure storage appliance, followed by metadata created by the secure storage appliance as well. The N secondary data blocks created from a primary data block are combined to form a stripe 714 of data. The metadata stored alongside each of the secondary data blocks 712 contains an indicator of the header used for encrypting the data. In one example implementation, each of the secondary data blocks 712 includes metadata that

specifies a number of times that the secondary data block has been written. A volume identifier and stripe location of an primary data block an be stored as well.

It is noted that, although a session key is associated with a volume, multiple session keys can be used per volume. For example, a volume may include one session key per 64 GB block of data. In this example, each 64 GB block of data contains an identifier of the session key to use in decrypting that 64 GB block of data. The session keys used to encrypt data in each of strips 700 can be of any of a number of forms. In certain embodiments, the session keys use an AES-256 Counter with Bit Splitting. In other embodiments, it may be possible to perform bit splitting without encryption. Therefore, alongside each secondary data block 712, an indicator of the session key used to encrypt the data block may be provided.

A variety of access request prioritization algorithms can be included for use with the volume, to allow access of only quickest-responding physical storage devices associated with the volume. Status information can be stored in association with a volume and/or share as well, with changes in status logged based on detection of event occurrences. The status log can be located in a reserved, dedication portion of memory of a volume. Other arrangements are possible as well.

It is noted that, based on the encryption of session keys with workgroup keys and the encryption of the secondary data blocks 712 in each of strips 700 with session keys, it is possible to effectively delete all of the data on a disk or volume (i.e. render the data useless) by deleting all workgroup keys that could decrypt a session key for that disk or volume.

Referring now to Figures 14-16, basic example flowcharts of setup and use of the networks and systems disclosed herein are described. Although these flowcharts are intended as example methods for administrative and I/O operations,

such operations can include additional steps/modules, can be performed in a different order, and can be associated with different number and operation of modules. In certain embodiments, the various modules can be executed concurrently.

Figure 14 shows a flowchart of systems and methods 800 for providing access to secure storage in a storage area network according to a possible embodiment of the present disclosure. The systems and methods 800 correspond to a setup arrangement for a network including a secure data storage system such as those described herein, including one or more secure storage appliances. The embodiments of the systems and methods described herein can be performed by an administrative user or administrative software associated with a secure storage appliance, as described herein.

Operational flow is instantiated at a start operation 802, which corresponds to initial introduction of a secure storage appliance into a network by an administrator or other individuals of such a network in a SAN, NAS, or other type of networked data storage environment. Operational flow proceeds to a client definition module 804 that defines connections to client devices (i.e. application servers or other front-end servers, clients, or other devices) from the secure storage appliance. For example, the client definition module 804 can correspond to mapping connections in a SAN or other network between a client such as application server 202 and a secure storage appliance 120 of Figure 4.

Operational flow proceeds to a storage definition module 806. The storage definition module 806 allows an administrator to define connections to storage systems and related physical storage devices. For example, the storage

definition module 806 can correspond to discovering ports and routes to storage systems 204 within the system 200 of Figure 4, above.

Operational flow proceeds to a volume definition module 808. The volume definition module 808 defines available volumes by grouping physical storage into logical arrangements for storage of shares of data. For example, an administrator can create a volume, and assign a number of attributes to that volume. A storage volume consists of multiple shares or segments of storage from the same or different locations. The administrator can determine a number of shares into which data is cryptographically split, and the number of shares required to reconstitute that data. The administrator can then assign specific physical storage devices to the volume, such that each of the N shares is stored on particular devices. The volume definition module 808 can generate session keys for storing data on each of the physical storage devices, and store that information in a key server and/or on the physical storage devices. In certain embodiments, the session keys generated in the volume definition module 808 are stored both on a key server connected to the secure storage appliance and on the associated physical storage device (e.g. after being encrypted with an appropriate workgroup key generated by the communities of interest module 810, below). Optionally, the volume definition module 808 includes a capability of configuring preferences for which shares are first accessed upon receipt of a request to read data from those shares.

Operational flow proceeds to a communities of interest module 810. The communities of interest module 810 corresponds to creation of one or more groups of individuals having interest in data to be stored on a particular volume. The communities of interest module 810 module further corresponds to assigning of access rights and visibility to volumes to one or more of those groups.

In creating the groups via the communities of interest module 810, one or more workgroup keys may be created, with each community of interest being associated with one or more workgroup keys. The workgroup keys are used to encrypt access information (e.g. the session keys stored on volumes created during operation of the volume definition module 808) related to shares, to ensure that only individuals and devices from within the community of interest can view and access data associated with that group. Once the community of interest is created and associated with a volume, client devices identified as part of the community of interest can be provided with a virtual disk, which is presented to the client device as if it is a single, unitary volume upon which files can be stored.

In use, the virtual disks appear as physical disks to the client and support SCSI or other data storage commands. Each virtual disk is associated on a many-to-one basis with a volume, thereby allowing multiple communities of interest to view common data on a volume (e.g. by replicating the relevant session keys and encrypting those keys with relevant workgroup keys of the various communities of interest). A write command will cause the data to be encrypted and split among multiple shares of the volume before writing, while a read command will cause the data to be retrieved from the shares, combined, and decrypted.

Operational flow terminates at end operation 812, which corresponds to completion of the basic required setup tasks to allow usage of a secure data storage system.

Figure 15 shows a flowchart of systems and methods 820 for reading block-level secured data according to a possible embodiment of the present disclosure. The systems and methods 820 correspond to a read or input command related to data stored via a secure storage appliance, such as those described herein.

Operational flow in the system and methods 820 begins at a start operation 822. Operational flow proceeds to a receive read request module 824, which corresponds to receipt of a primary read request at a secure storage appliance from a client device (e.g. an application server or other client device, as illustrated in Figures 3-4). The read request generally includes an identifier of a virtual disk from which data is to be read, as well as an identifier of the requested data.

Operational flow proceeds to an identity determination module 826, which corresponds to a determination of the identity of the client from which the read request is received. The client's identity generally corresponds with a specific community of interest. This assumes that the client's identity for which the secure storage appliance will access a workgroup key associated with the virtual disk that is associated with the client.

Operational flow proceeds to a share determination module 828. The share determination module 828 determines which shares correspond with a volume that is accessed by way of the virtual disk presented to the user and with which the read request is associated. The shares correspond to at least a minimum number of shares needed to reconstitute the primary data block (i.e. at least M of the N shares). In operation, a read module 830 issues secondary read requests to the M shares, and receives in return the secondary data blocks stored on the associated physical storage devices.

A success operation 832 determines whether the read module 830 successfully read the secondary data blocks. The success operation may detect for example, that data has been corrupted, or that a physical storage device holding one of the M requested shares has failed, or other errors. If the read is successful, operational flow branches "yes" to a reconstitute data module 834. The reconstitute

data module 834 decrypts a session key associated with each share with the workgroup key accessed by the identity determination module 826. The reconstitute data module 834 provides the session key and the encrypted and cryptographically split data to a data processing system within the secure storage appliance, which reconstitutes the requested data in the form of an unencrypted block of data physical disk locations in accordance with the principles described above in Figures 8-9 and 13. A provide data module 836 sends the reconstituted block of data to the requesting client device. A metadata update module 838 updates metadata associated with the shares, including, for example, access information related to the shares. From the metadata update module 838, operational flow proceeds to an end operation 840, signifying completion of the read request.

If the success operation 832 determines that not all of the M shares are successfully read, operational flow proceeds to a supplemental read operation 842, which determines whether an additional share exists from which to read data. If such a share exists (e.g. M < N), then the supplemental read operation reads that data, and operational flow returns to the success operation 832 to determine whether the system has now successfully read at least M shares and can reconstitute the primary data block as requested. If the supplemental read operation 842 determines that no further blocks of data are available to be read (e.g. M = N or M + failed reads > N), operational flow proceeds to a fail module 844, which returns a failed read response to the requesting client device. Operational flow proceeds to the metadata update module 838 and end operation 840, respectively, signifying completion of the read request.

Optionally, the fail module 844 can correspond to a failover event in which a backup copy of the data (e.g. a second N shares of data stored remotely

from the first N shares) are accessed. In such an instance, once those shares are tested and failed, a fail message is sent to a client device.

In certain embodiments, commands and data blocks transmitted to the client device can be protected or encrypted, such as by using a public/private key or symmetric key encryption techniques, or by isolating the data channel between the secure storage appliance and client. Other possibilities exist for protecting data passing between the client and secure storage appliance as well.

Furthermore, although the system and methods 820 of Figure 15 illustrates a basic read operation, it is understood that certain additional cases related to read errors, communications errors, or other anomalies may occur which can alter the flow of processing a read operation. For example, additional considerations may apply regarding which M of the N shares to read from upon initially accessing physical storage devices 206. Similar considerations apply with respect to subsequent secondary read requests to the physical storage devices in case those read requests fail as well.

Figure 16 shows a flowchart of systems and methods 850 for writing block-level secured data according to a possible embodiment of the present disclosure. The systems and methods 850 as disclosed provide a basic example of a write operation, and similarly to the read operation of Figure 15 additional cases and different operational flow may be used.

In the example systems and methods 850 disclosed, operational flow is instantiated at a start operation 852. Operational flow proceeds to a write request receipt module 854, which corresponds to receiving a primary write request from a client device (e.g. an application server as shown in Figures 3-4) at a secure storage

appliance. The primary write request generally addresses a virtual disk, and includes a block of data to be written to the virtual disk.

Operational flow proceeds to an identity determination module 856, which determines the identity of the client device from which the primary write request is received. After determining the identity of the client device, the identity determination module 856 accesses a workgroup key based upon the identity of the client device and accesses the virtual disk at which the primary write request is targeted. Operational flow proceeds to a share determination module 858, which determines the number of secondary data blocks that will be created, and the specific physical disks on which those shares will be stored. The share determination module 858 obtains the session keys for each of the shares that are encrypted with the workgroup key obtained in the identity determination module 856 (e.g. locally, from a key manager, or from the physical disks themselves). These session keys for each share are decrypted using the workgroup key.

Operational flow proceeds to a data processing module 860, which provides to the parser driver 304 the share information, session keys, and the primary data block. The parser driver 304 operates to cryptographically split and encrypt the primary data block, thereby generating N secondary data blocks to be written to N shares in accordance with the principles described above in the examples of Figures 8-9 and 13. Operational flow proceeds to a secondary write module 862 which transmits the share information to the physical storage devices for storage.

Operational flow proceeds to a metadata storage module 864, which updates a metadata repository by logging the data written, allowing the secure storage appliance to track the physical disks upon which data has been written, and

with what session and workgroup keys the data can be accessed. Operational flow terminates at an end operation 866, which signifies completion of the write request.

As previously mentioned, in certain instances additional operations can be included in the system and methods 850 for writing data using the secure storage appliance. For example, confirmation messages can be returned to the secure storage appliance confirming successful storage of data on the physical disks. Other operations are possible as well.

Now referring to Figures 17-18 of the present disclosure, certain applications of the present disclosure are discussed in the context of (1) data backup systems and (2) secure network thin client network topology used in the business setting. Figure 17 shows an example system 900 for providing secure storage data backup, according to a possible embodiment of the present disclosure. In the system 900 shown, a virtual tape server 902 is connected to a secure storage appliance 904 via a data path 906, such as a SAN network using Fibre Channel or iSCSI communications. The virtual tape server 902 includes a management system 908, a backup subsystem interface 910, and a physical tape interface 912. The management system 908 provides an administrative interface for performing backup operations. The backup subsystem interface 910 receives data to be backed up onto tape, and logs backup operations. A physical tape interface 912 queues and coordinates transmission of data to be backed up to the secure storage appliance 904 via the network. The virtual tape server 902 is also connected to a virtual tape management database 914 that stores data regarding historical tape backup operations performed using the system 900.

The secure storage appliance 904 provides a virtual tape head assembly 916 which is analogous to a virtual disk but appears to the virtual tape

server 902 to be a tape head assembly to be addressed and written to. The secure storage appliance 904 connects to a plurality of tape head devices 918 capable of writing to magnetic tape, such as that typically used for data backup. The secure storage appliance 904 is configured as described above. The virtual tape head assembly 916 provides an interface to address data to be backed up, which is then cryptographically split and encrypted by the secure storage appliance and stored onto a plurality of distributed magnetic tapes using the tape head devices 918 (as opposed to a generalized physical storage device, such as the storage devices of Figures 3-4).

In use, a network administrator could allocate virtual disks that would be presented to the virtual tape head assembly 916. The virtual tape administrator would allocate these disks for storage of data received from the client through the virtual tape server 902. As data is written to the disks, it would be cryptographically split and encrypted via the secure storage appliance 904.

The virtual tape administrator would present virtual tapes to a network (e.g. an IP or data network) from the virtual tape server 902. The data in storage on the tape head devices 918 is saved by the backup functions provided by the secure storage appliance 904. These tapes are mapped to the virtual tapes presented by the virtual tape head assembly 916. Information is saved on tapes as a collection of shares, as previously described.

An example of a tape backup configuration illustrates certain advantages of a virtual tape server over the standard tape backup system as described above in conjunction with Figure 2. In one example of a tape backup configuration, share 1 of virtual disk A, share 1 of virtual disk B, and other share 1's can be saved to a tape using the tape head devices 918. Second shares of each of

these virtual disks could be stored to a different tape. Keeping the shares of a virtual tape separate preserves the security of the information, by distributing that information across multiple tapes. This is because more than one tape is required to reconstitute data in the case of a data restoration. Data for a volume is restored by restoring the appropriate shares from the respective tapes. In certain embodiments an interface that can automatically restore the shares for a volume can be provided for the virtual tape assembly. Other advantages exist as well.

Now referring to Figure 18, one possible arrangement of a thin client network topology is shown in which secure storage is provided. In the network 950 illustrated, a plurality of thin client devices 952 are connected to a consolidated application server 954 via a secured network connection 956.

The consolidated application server 954 provides application and data hosting capabilities for the thin client devices 952. In addition, the consolidated application server 954 can, as in the example embodiment shown, provide specific subsets of data, functionality, and connectivity for different groups of individuals within an organization. In the example embodiment shown, the consolidated application server 954 can connect to separate networks and can include separate, dedicated network connections for payroll, human resources, and finance departments. Other departments could have separate dedicated communication resources, data, and applications as well. The consolidated application server 954 also includes virtualization technology 958, which is configured to assist in managing separation of the various departments' data and application accessibility.

The secured network connection 956 is shown as a secure Ethernet connection using network interface cards 957 to provide network connectivity at the

server 954. However, any of a number of secure data networks could be implemented as well.

The consolidated application server 954 is connected to a secure storage appliance 960 via a plurality of host bus adapter connections 961. The secure storage appliance 960 is generally arranged as previously described in Figures 3-16. The host bus adapter connections 961 allow connection via a SAN or other data network, such that each of the dedicated groups on the consolidated application server 954 has a dedicated data connection to the secure storage appliance 960, and separately maps to different port logical unit numbers (LUNs). The secure storage appliance 960 then maps to a plurality of physical storage devices 962 that are either directly connected to the secure storage appliance 960 or connected to the secure storage appliance 960 via a SAN 964 or other data network.

In the embodiment shown, the consolidated application server 954 hosts a plurality of guest operating systems 955, shown as guest operating systems 955a-c. The guest operating systems 955 host user-group-specific applications and data for each of the groups of individuals accessing the consolidated application server. Each of the guest operating systems 955a-c have virtual LUNs and virtual NIC addresses mapped to the LUNs and NIC addresses within the server 954, while virtualization technology 958 provides a register of the mappings of LUNS and NIC addresses of the server 954 to the virtual LUNs and virtual NIC addresses of the guest operating systems 955a-c. Through this arrangement, dedicated guest operating systems 955 can be mapped to dedicated LUN and NIC addresses, while having data that is isolated from that of other groups, but shared across common physical storage devices 962.

As illustrated in the example of Figure 18, the physical storage devices 962 provide a typical logistical arrangement of storage, in which a few storage devices are local to the secure storage appliance, while a few of the other storage devices are remote from the secure storage appliance 960. Through use of (1) virtual disks that are presented to the various departments accessing the consolidated application server 954 and (2) shares of virtual disks assigned to local and remote storage, each department can have its own data securely stored across a plurality of locations with minimal hardware redundancy and improved security.

Although Figures 17-18 present a few options for applications of the secure storage appliance and secure network storage of data as described in the present disclosure, it is understood that further applications are possible as well. Furthermore, although each of these applications is described in conjunction with a particular network topology, it is understood that a variety of network topologies could be implemented to provide similar functionality, in a manner consistent with the principles described herein.

Now referring to Figures 19-24, various additional details are provided relating to internal details of handling I/O requests (e.g. read and write operation requests) in a secure storage appliance. In the various methods and systems disclosed in the below-described figures, state-based processing of data blocks is performed, allowing the secure storage appliance to perform sub-tasks as resources of the secure storage appliance become available for use. In such a manner, the secure storage appliance can improve throughput of processed data and I/O requests related to that data, based on these "pipelined" operations.

Referring now to Figure 19, a state diagram 1000 for simultaneous state-based cryptographic splitting in a secure storage appliance is shown, according

to aspects of the present disclosure. The state diagram represents states assignable to various stripes of data, represented as blocks of data written to or received from storage devices and managed in memory of a secure storage appliance. By assigning various states to the buffers and blocks of data stored in those buffers, a secure storage appliance using such states can employ thread-level parallelism to process multiple blocks of data (i.e. stripes).

In the embodiment shown, the state diagram 1000 includes an idle state 1002, a read state 1004, a decode state 1006, a transfer state 1008, an encode state 1010, and a write state 1012. The idle state 1002 represents a state in which a stripe (i.e. a block of data able to be stored in a buffer) is currently not in use. The buffer can include, for example, a direct buffer or a data buffer intended to hold a block of data (i.e. a stripe of data). So long as the state diagram remains in the idle state 1002 for that buffer, the buffer is available for reuse, signifying that no data is being tracked in that buffer. The idle state 1002 can be entered from any of the other states, upon failure of an operation, or upon completion of an I/O request.

The read state 1004 signifies that data is being read from a plurality of shares and is destined for storage in a buffer on the secure storage appliance. The read state 1004 can be entered from the idle state any time the data being accessed is not present in memory, or upon retrying a previously unsuccessful read or decode operation.

The decode state 1006 signifies that data read (e.g. while the buffer is associated with the read state 1004) has occurred, and that currently the plurality of secondary data blocks are being decoded to form the block of data stored in the stripe. The decode state 1006 generally signifies that the data is being operated on by a cryptographic decoding operation, such as through use of a parser driver as

previously described. The decode state 1006 can be entered, for example, from the read state 1004 upon determination of a successful read and available parser module, or upon retrying a previously failed decode operation.

The transfer state 1008 indicates that the I/O request received at the secure storage appliance is being transferred into or out from the block of data held in the buffer. In the case of a write I/O request, the transfer state 1008 corresponds to writing data into the block of data, and marking the block of data as dirty (i.e. the data stored on physical storage devices is not up to date). In the case of a read I/O request, the transfer state 1008 corresponds to return of some or all of the block of data to a client device from the buffer. The transfer state 1008 can be entered, for example, from the decode states 1006 upon determining that a stripe is present (i.e. that the block of data making up the stripe is present in a buffer).

The transfer state 1008 can also be entered from the idle state 1002 in case either the stripe is already present (e.g. based on action in response to a different I/O request) or in the case that a full block write is taking place (in which case a read of that block is unnecessary, as the entire block will be overwritten).

The encode state 1010 signifies that the transfer state 1008 has completed, such that the data in the buffer is the most up-to-date data related to that block. The encode state 1010 corresponds to operation of the parser module of a secure storage appliance to cryptographically split and encrypt the data in the block of data. The encode state 1010 can be entered from the transfer state 1008 upon determination that the transfer state has completed and that the parser module is available. The encode state 1010 can also be entered from itself, such as upon retry of a failed encode operation.

The write state 1012 signifies completion of the encode state 1010, and schedules writing of the encoded data to a plurality of shares associated with the volume and strip to which the data is stored. The write state 1012 can track the existence of the buffer in an outstanding write list (e.g. as described above in conjunction with Figure 6) or other write operation. The write state 1012 can be entered from the encode state 1010 or upon reentry from itself, in the case of a failed write.

Additional states can be included as well, depending upon whether any additional processing actions are required which may use a resource of the secure storage appliance which may require scheduling and coordination of use.

The state diagram 1000 of Figure 19 can be implemented in software of the secure storage appliance to track the state of a number of data buffers that may be present in the secure storage appliance. In certain embodiments, the data buffers can be held in cache or RAM memory of any of the embodiments of the secure storage appliance previously described.

Figure 20 shows a flowchart of methods and systems 1100 for simultaneous state-based cryptographic splitting in a secure storage appliance, according to aspects of the present disclosure. The flowchart tracks processing of a single buffer and related block of data in the secure storage appliance. As illustrated, the methods and systems 1100 split each I/O request into a plurality of tasks, thereby allowing each I/O request to be managed in parallel and executed as resources in the secure storage appliance (read and write data lines, buffers, parser driver, etc) become available.

The system 1100 is instantiated at a start operation 1102, which corresponds to initial operation of a secure storage appliance or connection of the

secure storage appliance to a client device such that the secure storage appliance can begin receiving I/O requests. Operational flow proceeds to a request receipt module 1104 which receives an I/O request (e.g. a read or write request) associated with a particular block of data on a volume. The block of data, as referred to herein, corresponds to a block of data as expected to be received by the client device, rather than the cryptographically split secondary data blocks stored on the shares of the physical storage device.

Operational flow proceeds to a stripe presence determination operation 1106. The stripe presence determination operation 1106 determines whether the stripe related to the received I/O request is present in the secure storage appliance. This may be the case, for example, if the stripe (i.e. the block of data) has previously been requested by a different I/O operation and is present in a buffer of the secure storage appliance. The stripe presence determination operation 1106 also determines whether the I/O request is a full block write (and therefore there is no need to acquire the data block prior to performing a read or write).

If the stripe is not present in the secure storage appliance (and the I/O request is not a full block write), operational flow branches "no" from the stripe presence determination operation 1106 to a read module 1108. The read module 1108 initiates a read process to obtain a block of data from a volume by accessing cryptographically split secondary data blocks on a plurality of shares on a plurality of storage devices. During execution of the read module 1108, a buffer can also be reserved for the read block of data, and the buffer state can be set to a read state, as described in Figure 19, above.

A read assessment operation 1110 determines whether the read module executed successfully. If the read assessment operation 1110 determines

that the read module 1108 did not execute successfully, operational flow branches "no" and returns to the read module to retry the read operation. In certain embodiments, after a number of failed read assessment operations, operational flow fails, and the entire system 1100 is restarted.

If the read assessment operation 1110 determines that the read module 1108 executed successfully, operational flow branches "yes" to a decode module 1112. The decode module 1112 receives the cryptographically split secondary data blocks, and reconstitutes the block of data from those secondary data blocks. In certain embodiments, this can be accomplished by using a parser driver, as previously explained in conjunction with Figure 6. During operation of the decode module, the state of the buffer reserved for use in conjunction with the block of data can be set to a decode state, as described above in Figure 19.

A decode assessment operation 1114 determines whether the decode module 1112 executed successfully, by checking the correctness of the block of data output to a buffer from the parser driver or other software/hardware used to reconstitute the block of data. If the decode assessment operation 1114 determines that the decode module 1112 executed successfully, operational flow branches "yes" to a transfer module 1116. If the decode assessment operation 1114 determines that the decode module 1112 failed, operational flow branches "no" and returns to the read module 1108 to retry the reading and decoding process. In such an instance, the state of the buffer is changed from the decode state to the read state, and the read operation is reattempted.

In an alternative embodiment, if the decode assessment operation 1114 determines that the decode module 1112 failed, operational flow can branch

"no" and return to the decode module 1112 to retry the decoding process only (e.g. as shown in Figure 19).

After successful decoding of a block of data identified in the overall system 1100, a desired block of data is residing in a buffer on the secure storage appliance, or a full block write is to be performed. Operational flow proceeds to a transfer operation 1116 either (1) from the decode assessment operation 1114, as described above, or, (2) if a stripe was previously present in the secure storage appliance and detected by the stripe presence determination operation 1106 or was a full block write. The transfer module associates the I/O request with the block of data (now in unencrypted, whole, clear text form) stored in a system buffer, and I/O requests related to that buffer are processed (e.g. in FIFO order). The transfer module 1116 either transfers in data relating to a write request addressing a data location within the block of data, or copies out data from the buffer related to a read request addressing a location within the block of data. If a write request is executed, then the data block will be marked as "dirty" using a flag or other means to indicate that it contains changed data. During operation of the transfer module 1116, the state of the buffer reserved for use in conjunction with the block of data can be set to a transfer state, as described above in Figure 19.

Related to that selected I/O request for processing, operational flow proceeds to an update determination operation 1118. The update determination operation determines whether data in a stripe has been updated (e.g. by a write of a full or partial data block to the buffer). If the stripe has updated data, operational flow branches "yes" to an encode module 1120. The encode module 1120 applies cryptographic splitting to the now-updated data held in a buffer on the secure storage appliance, to generate a plurality of secondary data blocks in accordance with the

techniques described above. The encode module 1120 can do so by, for example, passing the data block that is the subject of the I/O request to a parser driver in the secure storage appliance. During operation of the encode module, the state of the buffer reserved for use in conjunction with the block of data can be set to an encode state, as described above in Figure 19.

Operational flow proceeds to an encode assessment operation 1122, which determines whether the encoding of data was successful. If the encode was successful, operational flow branches "yes" and proceeds from the encode assessment operation 1122 to a write module 1124. If the encode was not successful, operational flow branches "no" and returns to the encode module 1120 to retry the encoding operation.

The write module 1124 schedules a write operation of the secondary data blocks to the plurality of shares on the physical storage devices that are associated with the volume to which the I/O is addressed and where the block of data resides. The write module 1124 in certain embodiments, does so by adding the secondary data blocks to an outstanding write list, which schedules a write operation upon the availability of a network connection (e.g. a SAN network connection such as a Fibre Channel or iSCSI) to a physical storage device. During operation of the write module, the state of the buffer reserved for use in conjunction with the block of data can be set to a write state, as described above in Figure 19.

Operational flow proceeds to a write assessment operation 1126, which determines whether the write operation completed successfully. If the write does complete successfully, operational flow branches "yes" and proceeds to a subsequent I/O determination operation 1128. If the write does not complete

successfully, operational flow branches "no" and returns to the write module 1124 to retry the write operation.

The subsequent I/O determination operation 1128 assesses whether an additional I/O request is present and which relates to the block of data. If an additional I/O request is present and awaiting execution, operational flow branches "yes" and returns to the transfer module 1116, to process that subsequent I/O request. If no additional I/O request is present, operational flow branches "no" to an end operation 1130.

Referring back to the update determination operation 1118, if the stripe has not been updated, no write operation back to a physical storage device is necessary. Therefore, operational flow branches "no" from the update determination operation and completes, terminating at end operation 1130. Additionally, and as previously described, if the I/O tracking operation determines that no additional I/O requests exist with respect to that block of data, operational flow branches "no" to the end operation 1130. Upon reaching the end operation 1130, the state can be set to an idle state, and the buffer can be made available other use, as described above in Figure 19.

Although in the system 1100 a specific set of functional modules is presented, no particular ordering of modules is required or implied. Additional states and modules can be included in the system 1100 as well, depending upon the determination of the data block status or the type of I/O request made.

Now referring to Figures 21-22, additional details regarding encoding and decoding a block of data in a secure storage device are provided in further detail. Figure 21 shows a flowchart of methods and systems for reconstituting data in a secure storage appliance, according to a possible aspect of the present

disclosure. The methods and systems 1200 described herein can correspond, in various aspects, to particular steps performed in a decode operation in association with a read or write I/O request.

The system 1200 is instantiated at a start operation 1202, which corresponds to initial scheduling of a decode operation using a parser driver of a secure storage appliance, such as any of the secure storage appliances described above. Operational flow proceeds to an obtain data module 1204, which obtains encoded data from an encoded data buffer pool and assigns a free buffer or direct buffer for storage of decoded data. The encoded data, in various embodiments, corresponds to a number of secondary data blocks that represent a cryptographically split block of data.

Operational flow proceeds to a reconstitution module 1206, which reconstitutes a block of data from the secondary data blocks. The reconstitution module 1206 generally corresponds to the decode module of Figure 20, and can, in certain embodiments, operate using a parser driver as described above in Figure 6.

Operational flow proceeds to a success determination operation 1208, which determines whether the operation performed by the decode module 1206 was successful. If the success determination operation 1208 determines that the reconstitution module 1206 has reconstituted the block of data successfully, operational flow branches "yes" to a transfer scheduling module 1210. The transfer scheduling module 1210 schedules a transfer to occur in accordance with the I/O request received by the secure storage appliance. If the I/O request is a write request, the transfer scheduling module 1210 schedules a write of received data to update the data that has been decoded. If the I/O request is a read request, the transfer scheduling module 1210 schedules a return of requested data to be sent to

the client device sending the read request, the timing of which occurs based on the availability of the connection to the client device. From the transfer scheduling module 1210, operational flow proceeds to an end operation 1212, signifying completion of the decode flow.

If the success determination operation 1208 determines that the reconstitution module 1206 has not reconstituted the block of data, operational flow branches "no" to a read scheduling module 1214, which schedules a read operation to occur, thereby retrying the request of the secondary data blocks of data from which the block of data (i.e. stripe) is reconstituted. From the read scheduling module 1214, operational flow proceeds to the end operation 1212, which represents completion of the current read and decode operation and thereby allowing the system to restart, e.g. retrying to read the block or failing and freeing the parser driver to act on another set of secondary data blocks.

Figure 22 shows a flowchart of methods and systems for cryptographically splitting data in a secure storage appliance, according to a possible aspect of the present disclosure. The methods and systems 1300 of the present disclosure generally correspond to processing of an encoding and writing portion of an I/O request, providing additional detail regarding certain portions of the overall data flow of Figure 20, above. The system 1300 is instantiated at a start operation 1302, which corresponds to initial scheduling of an encode operation using a parser driver of a secure storage appliance, such as any of the secure storage appliances described above. Operational flow proceeds to an obtain buffer module 1304, which obtains buffers for use in encoding data. The data block represents a clear text, set size data block that can be encoded into a number of secondary data blocks that represent the cryptographically split block of data.

Operational flow proceeds to an encode module 1306, which reconstitutes a block of data from the secondary data blocks. The encode module 1306 generally corresponds to the encode module of Figure 19, and can, in certain embodiments, operate using a parser driver as described above in Figure 6.

Operational flow proceeds to a success determination operation 1308, which determines whether the operation performed by the encode module 1306 was successful. If the success determination operation 1308 determines that the encode module 1306 has cryptographically split the block of data successfully, operational flow branches "yes" to a write scheduling module 1310. The write scheduling module 1310 schedules a write of the secondary data blocks to corresponding shares stored on physical storage devices connected to the secure storage appliance. From the write scheduling module 1310, operational flow proceeds to an end operation 1312, signifying completion of the encode flow.

If the success determination operation 1308 determines that the encode module 1306 has not encoded the block of data successfully, operational flow branches "no" to a transfer function module 1314, which reschedules a transfer function so that the I/O request can be reprocessed. From the transfer function module 1314, operational flow proceeds to the end operation 1312, which represents completion of the current operation and thereby allowing the system to restart, e.g. retrying to encode and write the block or failing and freeing the parser driver to act on another set of secondary data blocks.

Referring to Figures 21-22 generally, in various embodiments, each of the systems 1200, 1300 corresponds to an overall process which is tracked as a resource. In such embodiments, separate blocks of data may be operated on concurrently, and the state of each block can be tracked using states and other status

information, such as the states described in Figure 19 above. Figures 23-24 illustrate possible generalized applications of use of the state-enabled parallelism provided by the systems and methods of Figures 19-22, above.

Figure 23 shows a flowchart of methods and systems 1400 for managing data blocks in a secure storage appliance, according to a possible embodiment of the present disclosure. The systems 1400 disclosed operate within a secure storage appliance, and provide state-based management of blocks of data to provide the possibility of pipelined operation of the secure storage appliance to improve throughput and I/O request handling.

The system 1400 as shown is instantiated at a start operation 1402, which corresponds to initial operation of a secure storage appliance in conjunction with a client device and a plurality of physical storage devices. From the start operation 1402, operational flow proceeds to a data receipt module 1404. The data receipt module 1404 receives a block of data associated with a volume. The block of data is associated with an I/O request, such as a read or write request, and can be received either from a client device (e.g. in the case of a write request) or from a physical storage device (e.g. in the case of a read request). In the case the block of data is received from a physical storage request, the block of data may be a block referred to herein as a secondary data block, such that the secondary block of data represents a cryptographically split portion of a block of data as it would be viewed or presented to a client device. In further embodiments, additional blocks of data can be received as well.

Operational flow proceeds to a data storage module 1406. The data storage module 1406 stores the block of data received by the data receipt module 1404 and stores that data in a buffer. The buffer can be any of a number of buffers

available within a secure storage appliance, such as a direct buffer, any of a number of work buffers, a number of secondary buffers used to hold secondary data blocks (e.g. in the case of a read request), or other types or sizes of buffers.

Operational flow proceeds to a state association module 1408. The state association module 1408 associates the stripe with a state from among a plurality of states assignable to stipes in the system. Any of a number of states may be assigned to the stripe, depending upon the currently-pending action to be taken on data stored in the buffer (or signifying the lack of meaningful data in the buffer if the buffer has been released due to a completed I/O request). Example states are illustrated in Figure 19, above; however, other states may be possible as well.

Operational flow proceeds to a data processing module 1410. The data processing module 1410 processes the block of data by performing a cryptographic operation on it. In the case of a write operation in which a block of data is to be written to a physical storage device from a work buffer, the data processing module corresponds to cryptographic splitting of the block of data into a plurality of secondary blocks of data for storage. In the case of a read operation or a partial write operation (less than an entire block being modified), data from a physical storage device must be retrieved and reconstituted from secondary blocks of data, so an inverse function is performed. Examples of these operations are illustrated in the flowcharts of Figures 21-22, above.

While the data processing module 1410 remains in operation, the overall system preferably maintains the state of the stripe so that the secure storage appliance can track resource usage based on states of stipes. Other resource tracking arrangements are possible as well.

Operational flow proceeds to a state update module 1412. The state update module changes the state of the stripe maintaining the block of data after the data processing module 1410 completes operation, for example, once a write operation or read operation completes. In various embodiments, the state update module 1412 can change states according to the state transitions illustrated in Figure 19, above.

Operational flow within the system proceeds to an end operation 1414, which ends the process of managing the data block within the designated state.

Although Figure 23 is illustrated as including specific functionality occurring in a specified order, it is understood that the module ordering is not required, other than as necessary to maintain different states for a stripe at different stages of I/O request execution.

Figure 24 shows a flowchart of methods and systems 1500 for managing I/O requests in a secure storage appliance, according to a possible embodiment of the present disclosure. The methods and systems as described herein represent an arrangement in which multiple I/O requests can be handled in parallel, such as by tracking the state of each stripe being acted upon by I/O requests and allowing processing of an I/O request as a resource. In the context of the systems and methods described herein, these resources can include: the parser driver software and/or encryption/decryption hardware used in cryptographic splitting and reconstituting data; a data line or network connection (e.g. host bus adapter port) available between the secure storage appliance and one or more of the physical storage devices; a data connection between the secure storage appliance and a client device; a buffer, such as a work buffer, and encrypted secondary buffer, or a direct

buffer; or tracking resources available within the appliance. Other resources can be included as well.

Operational flow is instantiated at a start operation 1502, which corresponds to initial setup to allow a secure storage appliance to communicate with a client device and a plurality of physical storage devices in a network such as those described herein. Operational flow proceeds to a request receipt module 1504, which receives a plurality of I/O requests at the secure storage appliance. Each of the I/O requests received by the request receipt module 1504 corresponds to a volume and a block of data on that volume, where the volume is associated with a plurality of shares on physical storage devices. Example I/O requests will generally include the type of request (e.g. a read or write request), a location related to the request (i.e. the address of a portion of a block of data or a block of data related to the request), and, if a write operation, data to be written to the location. Other data can be transmitted as well.

Operational flow proceeds to a storage module 1506. The storage module stores blocks of data in buffers of the secure storage device. Each of the blocks of data are associated with one or more of the plurality of I/O requests. For example, the blocks of data can be reconstituted blocks of data stored in work buffers in response to read I/O requests or write I/O requests addressing only a partial block. Or, the blocks of data can include a block of data to be written to a device storage location.

Operational flow proceeds to a state association module 1508, which associates a state with each of the stripes in the secure storage appliance. The state association module 1508 can associate any of a number of states with a stripe, generally indicating the current processing state of the stripe. For example the state

assigned to the stripe can indicate how far the data associated with the stripe is processed by a currently-active I/O request. Example states are illustrated in Figure 19, above, but other states may be available as well.

Operational flow proceeds to a resource availability determination operation 1510. The resource availability determination operation 1510 determines which resource is necessary to be used next sequentially by the data in each of the buffers, and determines whether that resource is currently available. If the resource availability determination operation 1510 determines that a requested resource is available, operational flow branches "yes" and proceeds to a resource application module 1512. The resource application module 1512 applies the resource to the data in the buffer, thereby performing a required operation to process the I/O request, such as reading data, decoding data, transferring I/O requests, encoding data, writing data, or idling. Operational flow proceeds to an end operation 1514 upon completion of the resource application module 1512.

If the resource availability determination operation 1510 determines that the requested resource is not available, operational flow branches "no", issues the appropriate I/O request, and waits for the appropriate system event or events to signify that the resource to become available. Once the resource becomes available, operational flow can proceed to the resource application module 1512, above.

Optionally, the resource availability determination operation 1510 includes a time-out feature in which an operation is deemed to have failed if a resource does not become available to it within a set period of time. In other embodiments, the resource availability determination operation 1510 requires that the current I/O request and buffer must wait to complete its next operational step. However, other buffers associated with other I/O requests can be processed, so long

as the same resource is not required or used. In still further embodiments, the system uses an interrupt-based scheme to trigger use of a resource by the data in the buffer, in which a thread is notified when a resource can be allocated for its use.

Through use of the systems and methods herein, it is understood that each I/O request is split into a number of tasks, which can be pipelined to improve efficiency through the secure storage appliance.

It is recognized that the above networks, systems, and methods operate using computer hardware and software in any of a variety of configurations. Such configurations can include computing devices, which generally include a processing device, one or more computer readable media, and a communication device. Other embodiments of a computing device are possible as well. For example, a computing device can include a user interface, an operating system, and one or more software applications. Several example computing devices include a personal computer (PC), a laptop computer, or a personal digital assistant (PDA). A computing device can also include one or more servers, one or more mass storage databases, and/or other resources.

A processing device is a device that processes a set of instructions. Several examples of a processing device include a microprocessor, a central processing unit, a microcontroller, a field programmable gate array, and others. Further, processing devices may be of any general variety such as reduced instruction set computing devices, complex instruction set computing devices, or specially designed processing devices such as an application-specific integrated circuit device.

Computer readable media includes volatile memory and non-volatile memory and can be implemented in any method or technology for the storage of

information such as computer readable instructions, data structures, program modules, or other data. In certain embodiments, computer readable media is integrated as part of the processing device. In other embodiments, computer readable media is separate from or in addition to that of the processing device. Further, in general, computer readable media can be removable or non-removable. Several examples of computer readable media include, RAM, ROM, EEPROM and other flash memory technologies, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium that can be used to store desired information and that can be accessed by a computing device. In other embodiments, computer readable media can be configured as a mass storage database that can be used to store a structured collection of data accessible by a computing device.

A communications device establishes a data connection that allows a computing device to communicate with one or more other computing devices via any number of standard or specialized communication interfaces such as, for example, a universal serial bus (USB), 802.11 a/b/g network, radio frequency, infrared, serial, or any other data connection. In general, the communication between one or more computing devices configured with one or more communication devices is accomplished via a network such as any of a number of wireless or hardwired WAN, LAN, SAN, Internet, or other packet-based or port-based communication networks.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.

Claims:

1. A method of managing data blocks in a secure storage appliance, the method comprising:

receiving a block of data associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices;

storing the block of data in a buffer;

associating the block of data with a state from among a plurality of states, each of the states corresponding to a status of the block of data;

processing the block of data by performing at least one cryptographic operation on the block of data; and

upon completion of processing the block of data, updating the state of the block of data.

2. The method of claim 1, wherein processing the block of data includes cryptographically splitting the block of data into a plurality of secondary data blocks.

3. The method of claim 1, wherein processing the block of data includes reconstituting a primary block of data from a plurality of secondary blocks of data, the plurality of secondary blocks of data including the block of data.

4. The method of claim 1, wherein the plurality of states includes at least one of a read state, a decode state, an idle state, a transfer state, an encode state, or a write state.

5. The method of claim 1, further comprising, while processing the block of data, assigning a state to the block of data indicating that the block of data is being processed.

6. The method of claim 1, wherein receiving the block of data includes receiving the block of data from a client device.

7. The method of claim 5, wherein the buffer is a work buffer.

8. The method of claim 1, wherein processing the block of data includes reconstituting the block of data from a plurality of secondary data blocks stored in the plurality of shares.

9. A method of managing I/O requests in a secure storage appliance, the method including:

receiving an I/O request at the secure storage appliance, the I/O request associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices;

determining whether a block of data referenced by the I/O request is present in a buffer;

transferring the block of data to a buffer and associating the block of data with a transfer state;

determining whether the block of data in the buffer is updated; and

processing the block of data.

10. The method of claim 9, wherein processing the block of data includes determining whether the data block has been updated.

11. The method of claim 10, further comprising, upon determining that the data block has been updated, cryptographically splitting the block of data into a plurality of secondary data blocks and storing the plurality of secondary data blocks on the plurality of physical storage devices.

12. The method of claim 11, further comprising, while cryptographically splitting the block of data, associating the block of data with an encode state.

13. The method of claim 11, further comprising, while storing the plurality of secondary data blocks on the plurality of physical storage devices, associating the block of data with a write state.

14. The method of claim 9, wherein transferring the block of data to a buffer includes receiving the data block from a client device.

15. The method of claim 9, further comprising, prior to transferring the block of data to a buffer:

reading a plurality of secondary data blocks stored in the plurality of shares; and

reconstituting the block of data from the plurality of secondary data blocks.

16. The method of claim 15, further comprising, while reconstituting the block of data, associating the block of data with a decode state.

17. A secure storage appliance comprising;

a plurality of buffers, each buffer capable of holding a block of data having a state selected from among a plurality of states;

a programmable circuit capable of accessing the plurality of buffers, the programmable circuit configured to execute program instructions to:

receive an I/O request, the I/O request associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices communicatively connected to the secure storage appliance;

determine whether a block of data referenced by the I/O request is present in a buffer;

transfer the block of data to a buffer among the plurality of buffers;

determine whether the block of data in the buffer is updated; and

process the block of data.

18. The secure storage appliance of claim 17, wherein the plurality of states includes at least one of a read state, a decode state, an idle state, a transfer state, an encode state, or a write state.

19. The secure storage appliance of claim 17, wherein the programmable circuit is further programmed to, while transferring the block of data to a buffer, associating the block of data with a transfer state.

20. The secure storage appliance of claim 17, wherein the plurality of shares store cryptographically split data stored to the volume.

21. A secure storage appliance comprising:

a plurality of buffers, each buffer capable of holding a block of data having a state selected from among a plurality of states;

a programmable circuit capable of accessing the plurality of buffers, the programmable circuit configured to execute program instructions to:

receive a block of data associated with a volume, the volume associated with a plurality of shares stored on a plurality of physical storage devices;

store the block of data in a buffer from among the plurality of buffers;

associate the buffer with a state corresponding to a status of the block of data;

process the block of data by performing at least one cryptographic operation on the block of data; and

update the state of the buffer upon completion of processing the block of data.

22. The secure storage appliance of claim 21, wherein the programmable circuit is configured to execute program instructions to cryptographically split the block of data into a plurality of secondary data blocks.

23. The secure storage appliance of claim 21, wherein the programmable circuit is configured to execute program instructions to reconstitute a primary block of data from a plurality of secondary blocks of data, the plurality of secondary blocks of data including the block of data.

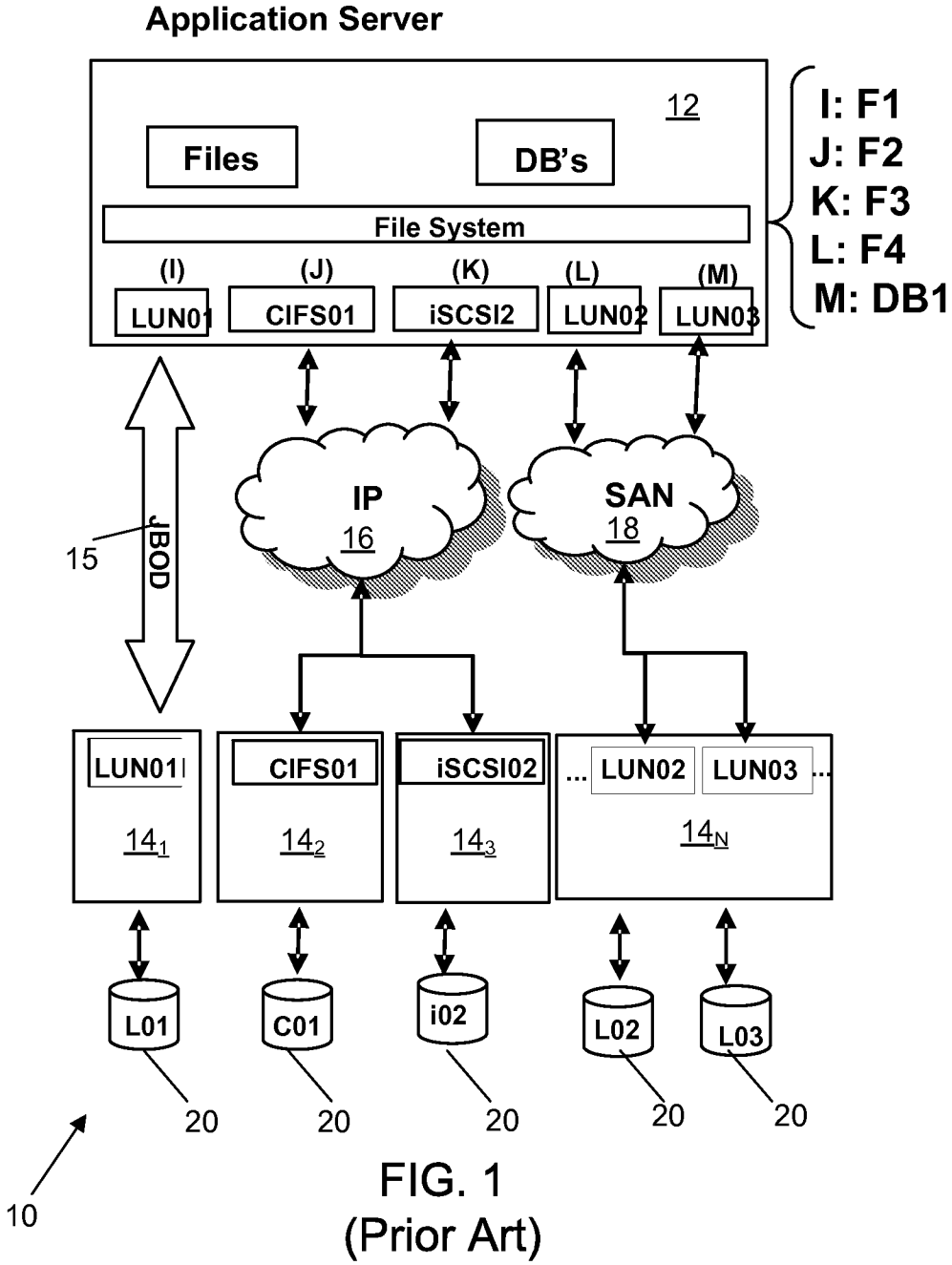

FIG. 1
(Prior Art)

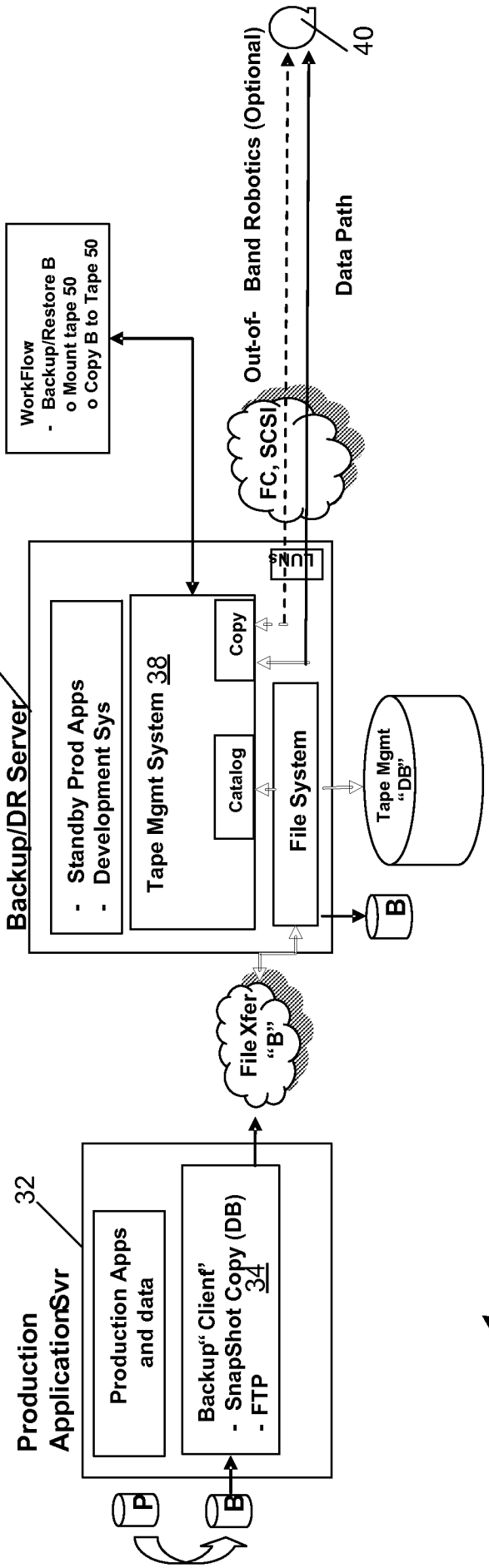

FIG. 2
(Prior Art)

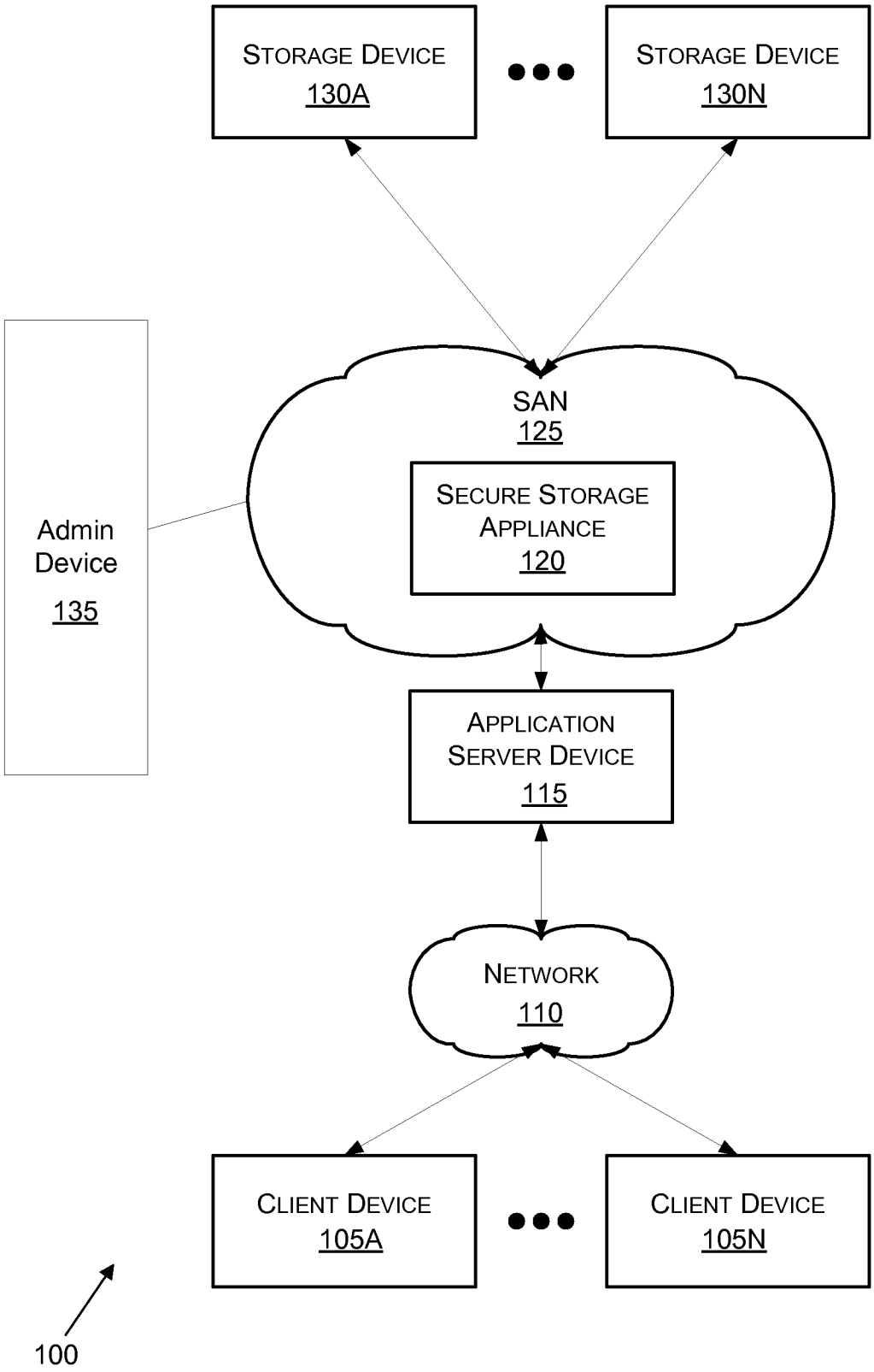
STORAGE DEVICE
130A
STORAGE DEVICE
130N
SAN
125
SECURE STORAGE
APPLIANCE
120
Admin
Device
135
APPLICATION
SERVER DEVICE
115
NETWORK
110
CLIENT DEVICE
105A
CLIENT DEVICE
105N
100

FIG. 3

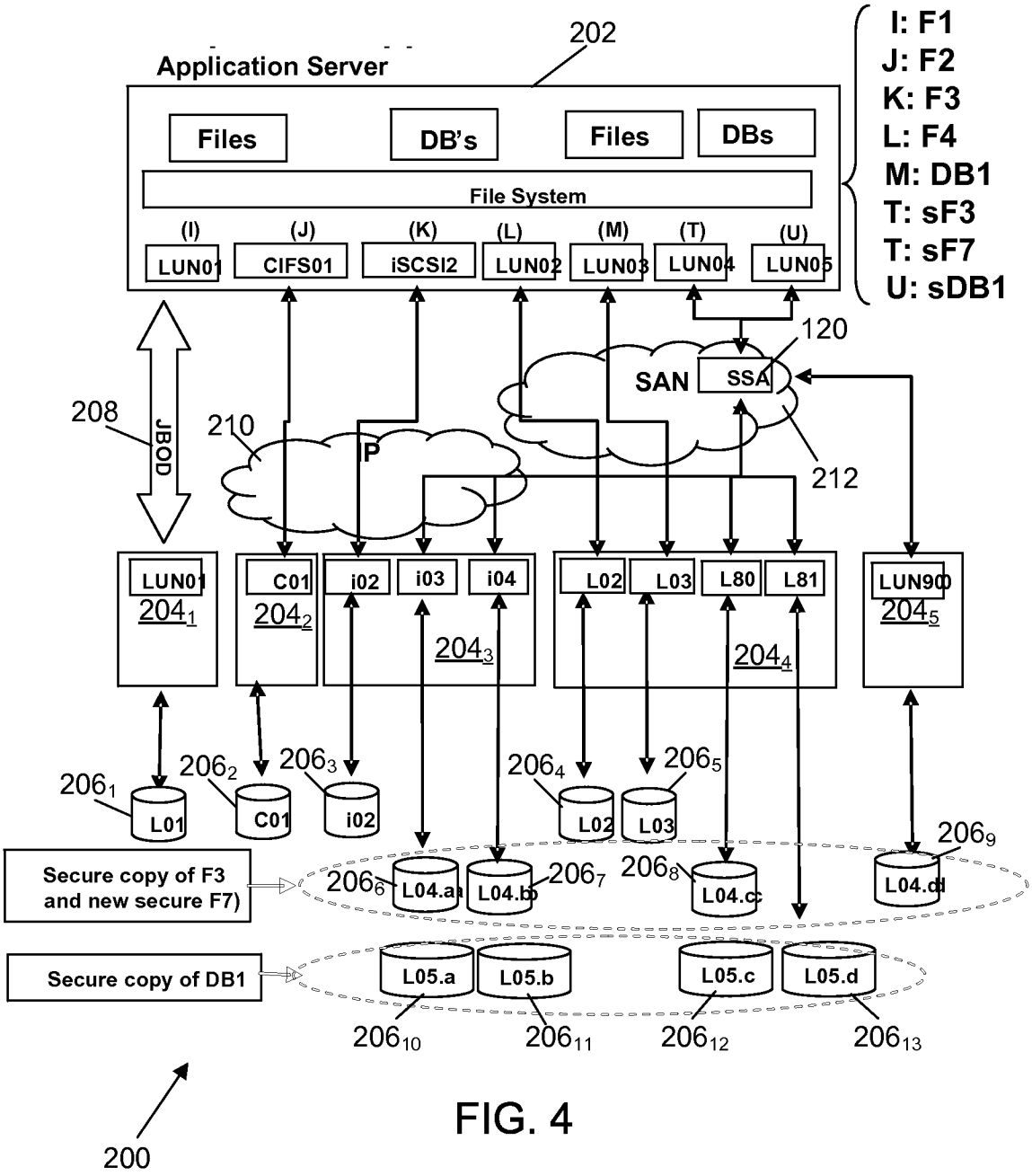
202
Application Server
Files
DB's
Files
DBs
File System
(I)
(J)
(K)
(L)
(M)
(T)
(U)
LUN01
CIFS01
iSCSI2
LUN02
LUN03
LUN04
LUN05
I: F1
J: F2
K: F3
L: F4
M: DB1
T: sF3
T: sF7
U: sDB1
120
SAN
SSA
208
JBOD
210
IP
212
LUN01
2041
C01
2042
i02
i03
i04
2043
L02
L03
L80
L81
2044
LUN90
2045
2061
2062
2063
2064
2065
2069
L01
C01
i02
L02
L03
L04.d
Secure copy of F3
and new secure F7)
2066
L04.a
L04.b
2067
2068
L04.c
Secure copy of DB1
L05.a
L05.b
L05.c
L05.d
20610
20611
20612
20613
200

FIG. 4

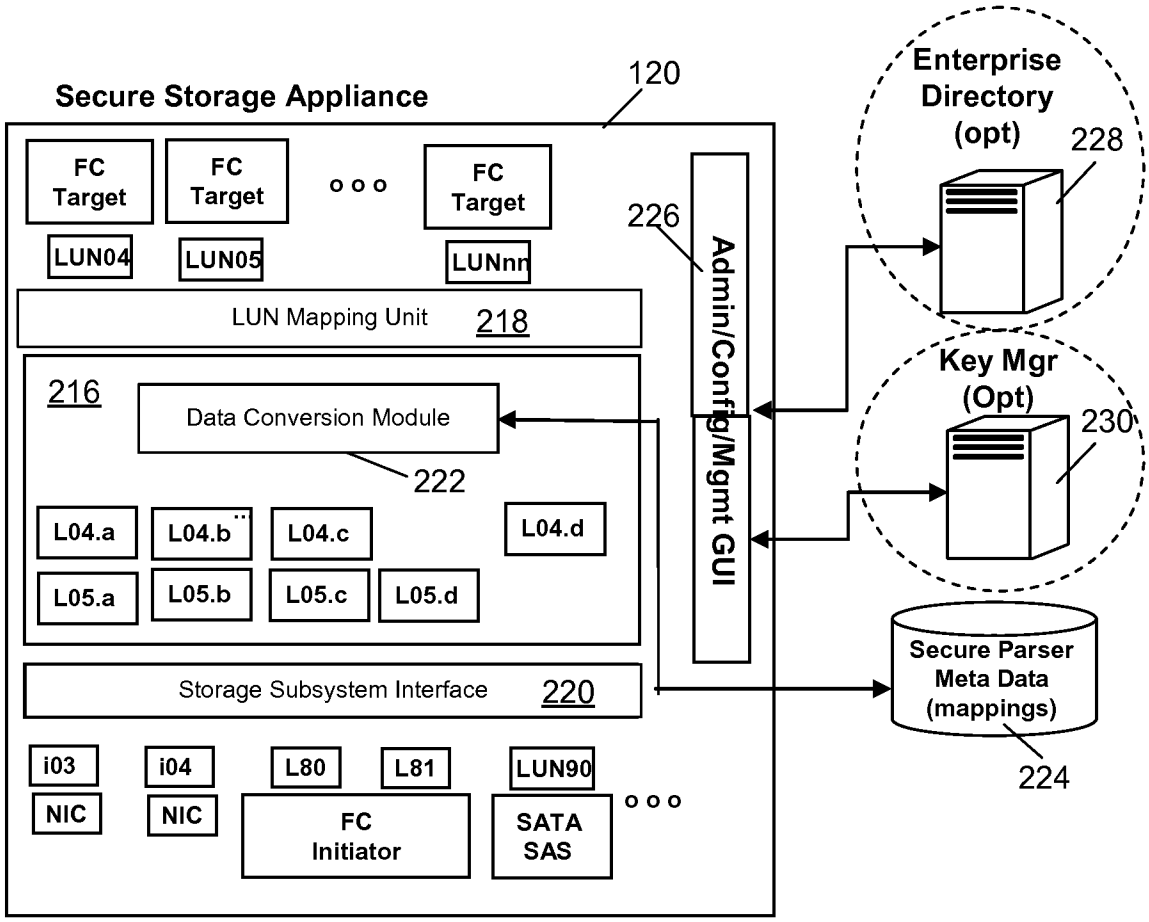
Secure Storage Appliance
120
FC Target
FC Target
o o o
FC Target
LUN04
LUN05
LUNnn
LUN Mapping Unit 218
216
Data Conversion Module
222
L04.a
L04.b
L04.c
L04.d
L05.a
L05.b
L05.c
L05.d
Storage Subsystem Interface 220
i03
i04
L80
L81
LUN90
NIC
NIC
FC Initiator
SATA SAS
o o o
226
Admin/Config/Mgmt GUI
Enterprise Directory (opt)
228
Key Mgr (Opt)
230
Secure Parser Meta Data (mappings)
224

FIG. 5

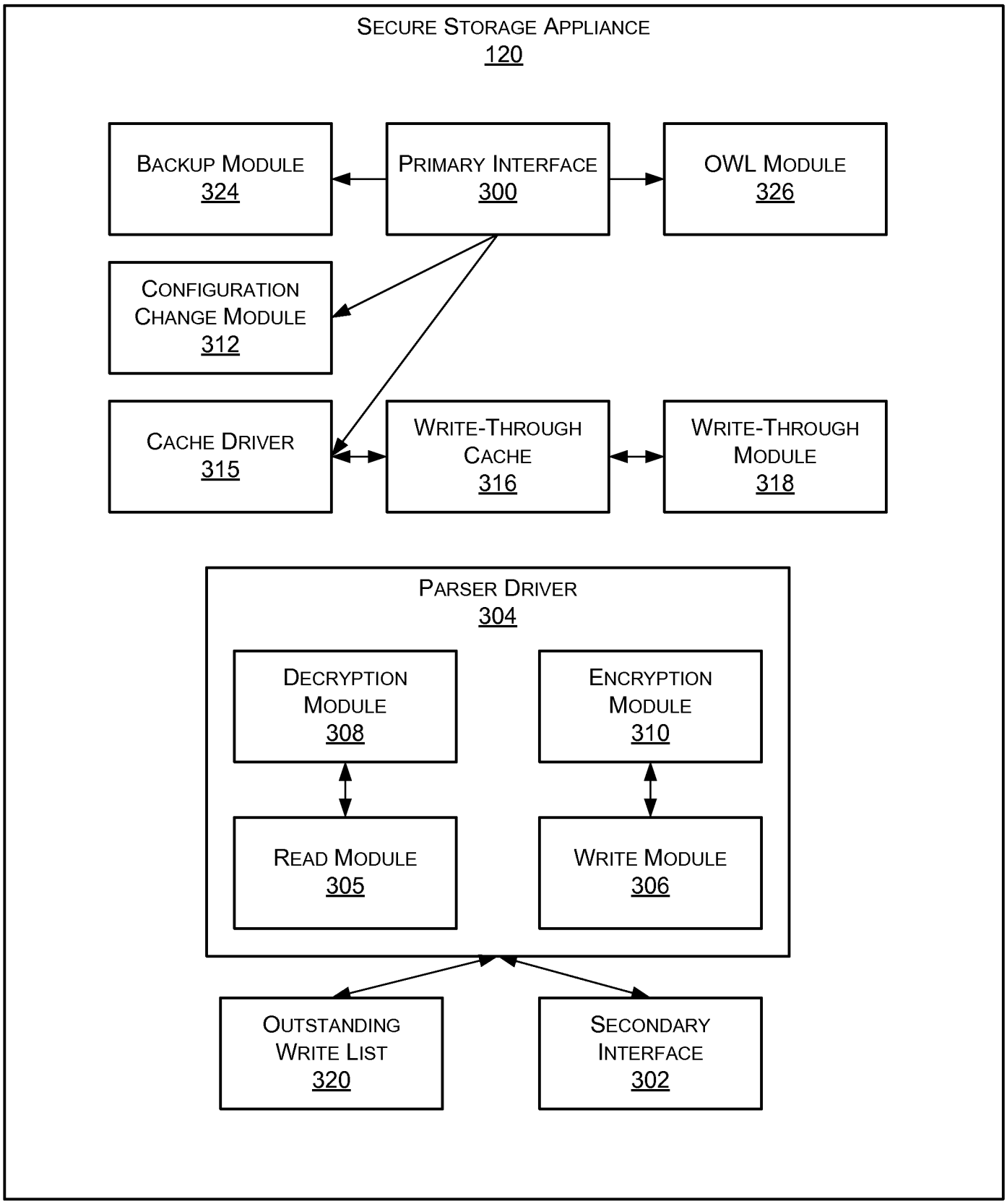
SECURE STORAGE APPLIANCE
120
BACKUP MODULE
324
PRIMARY INTERFACE
300
OWL MODULE
326
CONFIGURATION CHANGE MODULE
312
CACHE DRIVER
315
WRITE-THROUGH CACHE
316
WRITE-THROUGH MODULE
318
PARSER DRIVER
304
DECRYPTION MODULE
308
ENCRYPTION MODULE
310
READ MODULE
305
WRITE MODULE
306
OUTSTANDING WRITE LIST
320
SECONDARY INTERFACE
302

FIG. 6

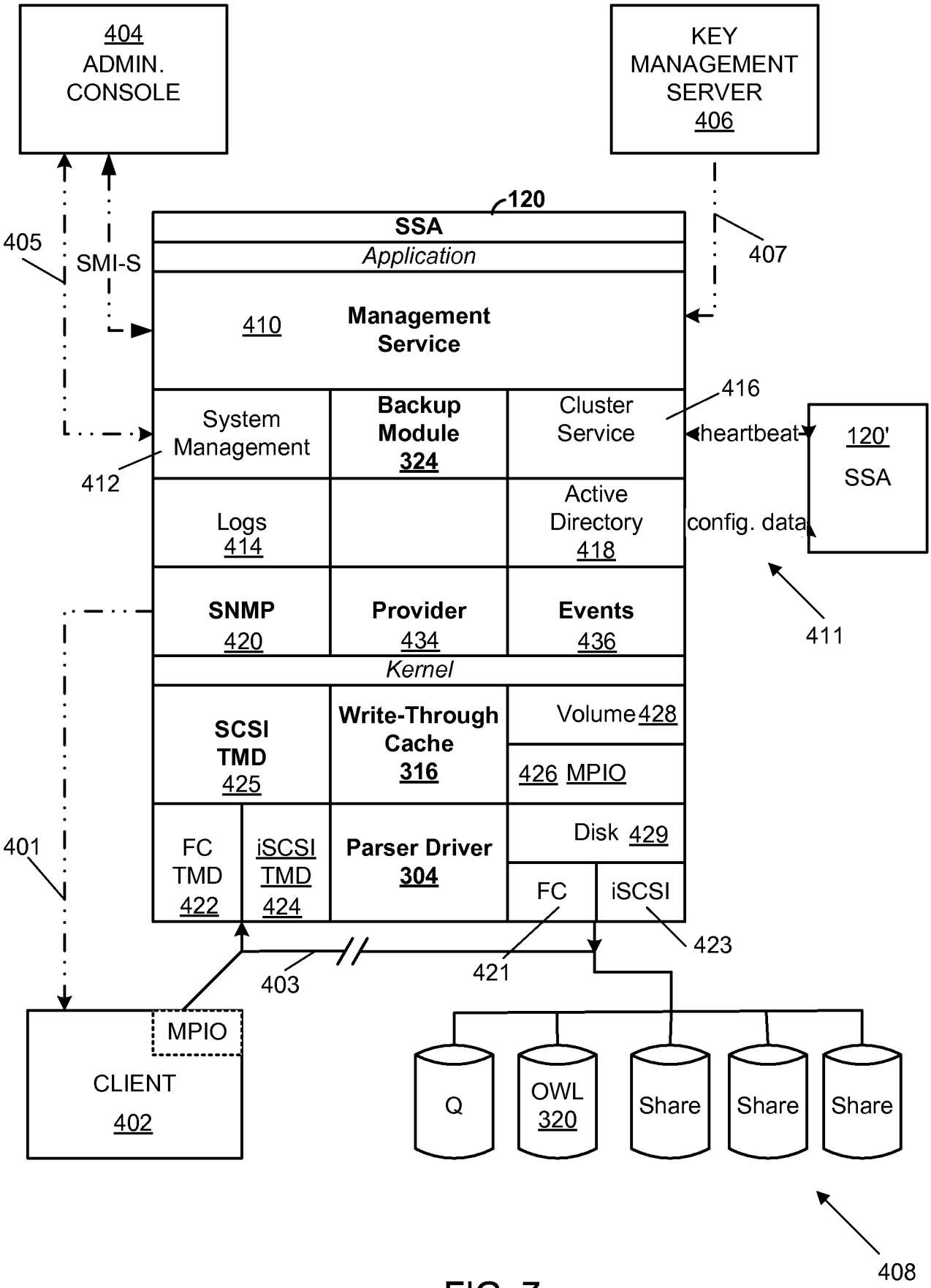
404
ADMIN.
CONSOLE
KEY
MANAGEMENT
SERVER
406
120
SSA
Application
405
SMI-S
407
410
Management
Service
416
System
Management
412
Backup
Module
324
Cluster
Service
heartbeat
120'
SSA
Logs
414
Active
Directory
418
config. data
411
SNMP
420
Provider
434
Events
436
Kernel
SCSI
TMD
425
Write-Through
Cache
316
Volume 428
426 MPIO
401
FC
TMD
422
iSCSI
TMD
424
Parser Driver
304
Disk 429
FC
iSCSI
423
403
421
MPIO
CLIENT
402
Q
OWL
320
Share
Share
Share
408

FIG. 7

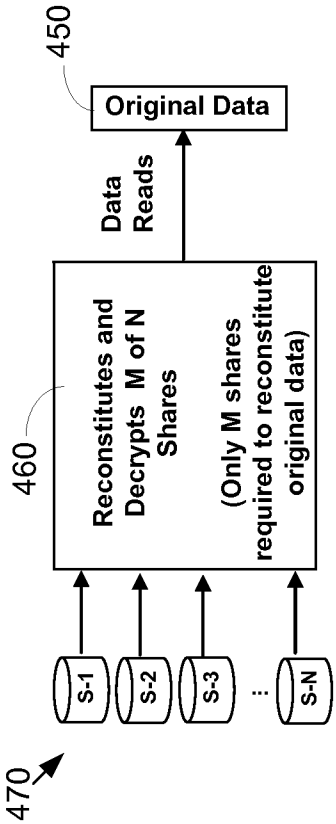
450
Original Data
Data Reads
460
Reconstitutes and Decrypts M of N Shares
(Only M shares required to reconstitute original data)
S-1
S-2
S-3
S-N
470

FIG. 9

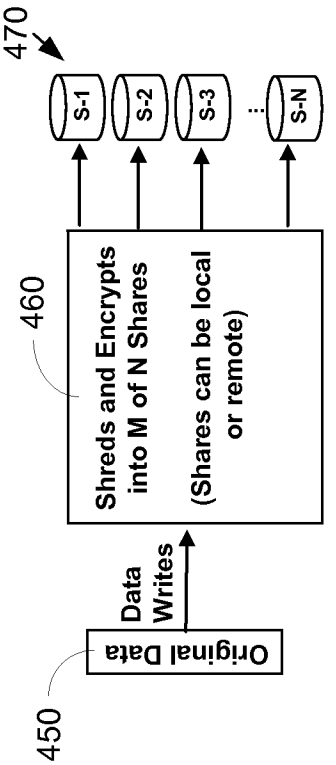
470
S-1
S-2
S-3
S-N
460
Shreds and Encrypts into M of N Shares
(Shares can be local or remote)
Data Writes
Original Data
450

FIG. 8

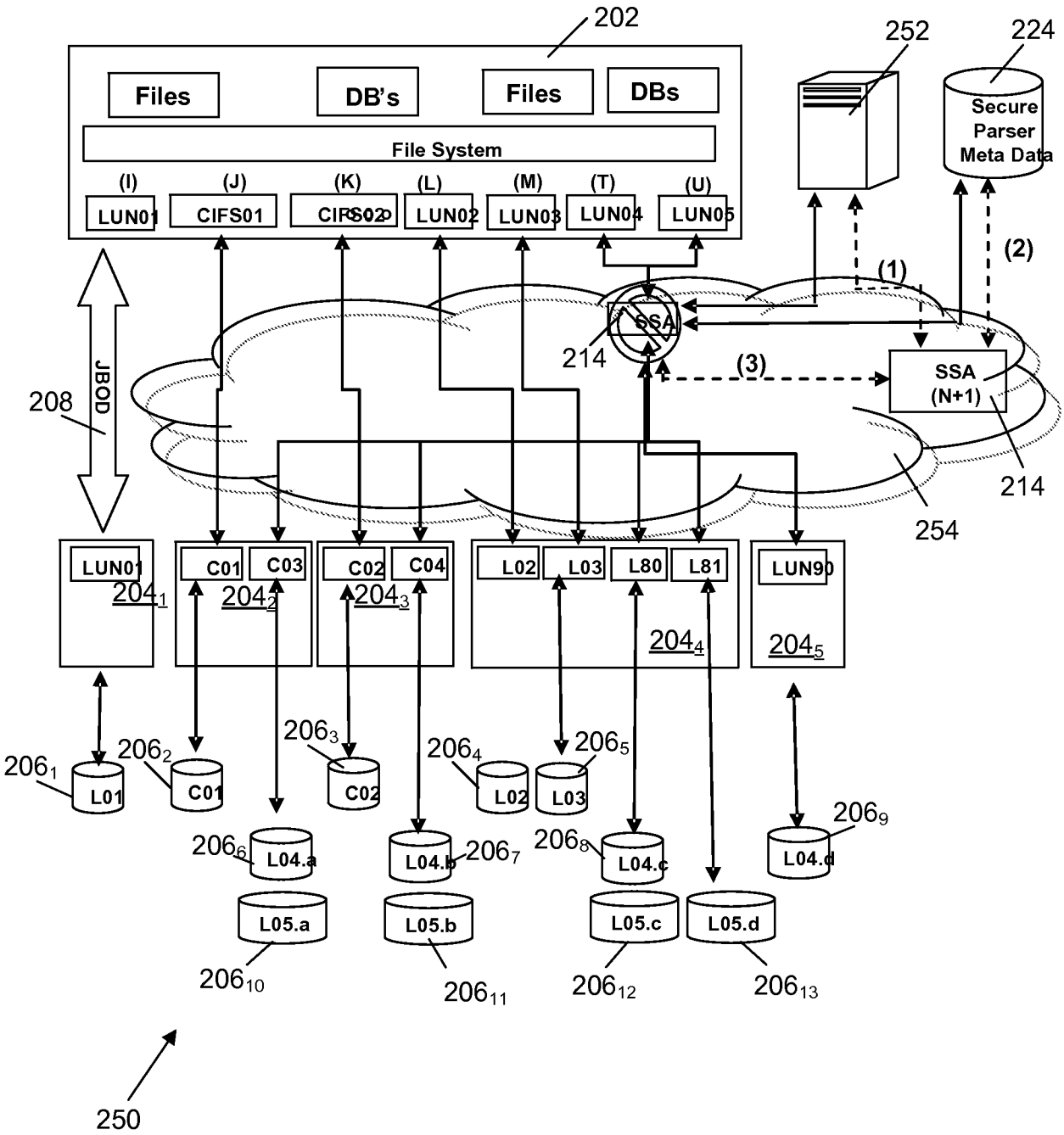
202
Files
DB's
Files
DBs
File System
(I)
(J)
(K)
(L)
(M)
(T)
(U)
LUN01
CIFS01
CIFS02
LUN02
LUN03
LUN04
LUN05
252
224
Secure
Parser
Meta Data
(1)
(2)
(3)
SSA
214
SSA
(N+1)
214
254
208
JBOD
LUN01
C01
C03
C02
C04
L02
L03
L80
L81
LUN90
204_1
204_2
204_3
204_4
204_5
206_1
L01
206_2
C01
206_3
C02
206_4
L02
206_5
L03
206_6
L04.a
206_7
L04.b
206_8
L04.c
206_9
L04.d
L05.a
L05.b
L05.c
L05.d
206_{10}
206_{11}
206_{12}
206_{13}
250

FIG. 10

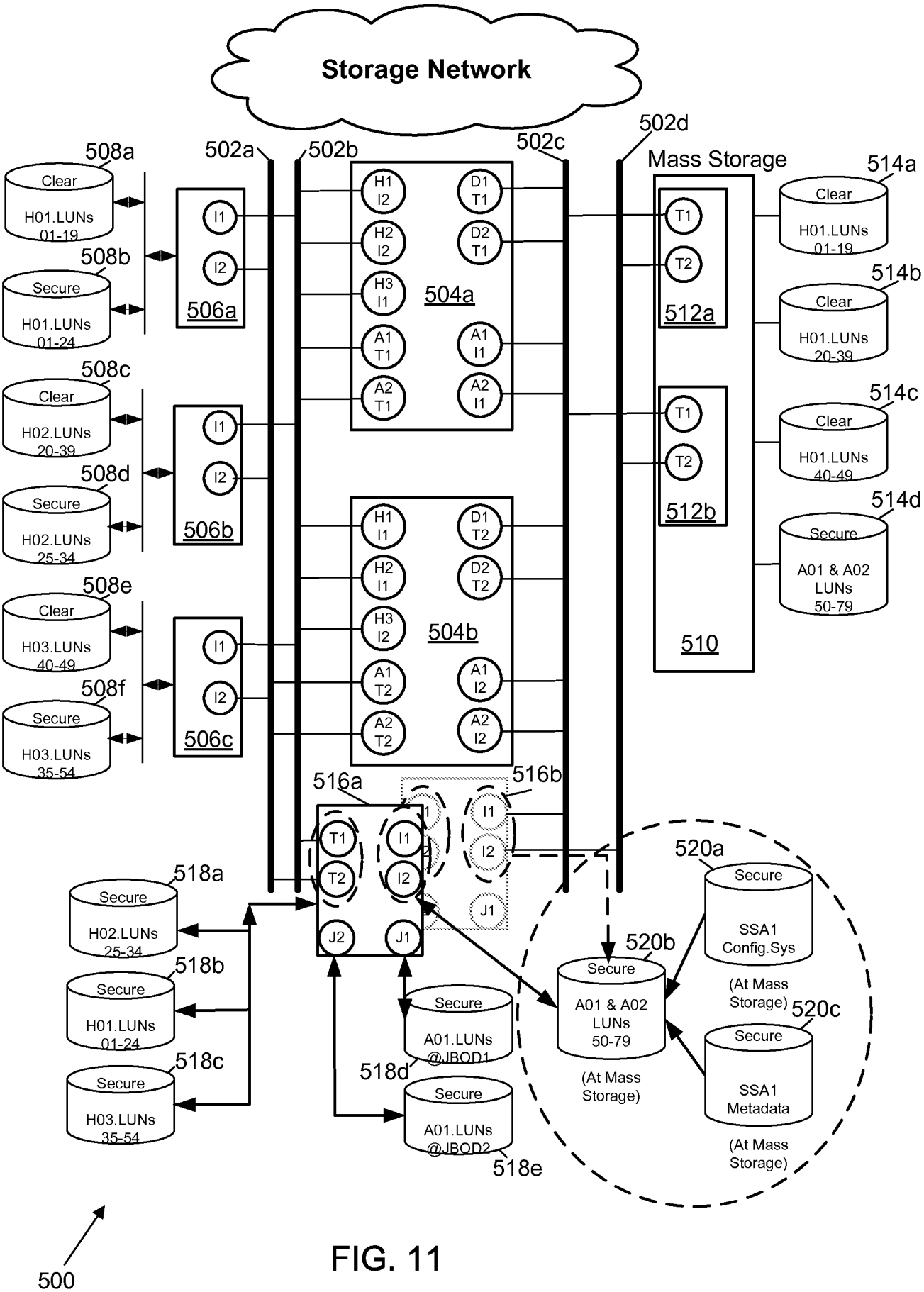
Storage Network
502a
502b
502c
502d
Mass Storage
508a
Clear
H01.LUNs 01-19
508b
Secure
H01.LUNs 01-24
508c
Clear
H02.LUNs 20-39
508d
Secure
H02.LUNs 25-34
508e
Clear
H03.LUNs 40-49
508f
Secure
H03.LUNs 35-54
I1
I2
506a
506b
506c
H1 I2
H2 I2
H3 I1
A1 T1
A2 T1
D1 T1
D2 T1
A1 I1
A2 I1
504a
H1 I1
H2 I1
H3 I2
A1 T2
A2 T2
D1 T2
D2 T2
A1 I2
A2 I2
504b
T1
T2
512a
512b
510
514a
Clear
H01.LUNs 01-19
514b
Clear
H01.LUNs 20-39
514c
Clear
H01.LUNs 40-49
514d
Secure
A01 & A02 LUNs 50-79
516a
516b
J1
J2
518a
Secure
H02.LUNs 25-34
518b
Secure
H01.LUNs 01-24
518c
Secure
H03.LUNs 35-54
518d
Secure
A01.LUNs @JBOD1
518e
Secure
A01.LUNs @JBOD2
520a
Secure
SSA1 Config.Sys
(At Mass Storage)
520b
Secure
A01 & A02 LUNs 50-79
(At Mass Storage)
520c
Secure
SSA1 Metadata
(At Mass Storage)
500

FIG. 11

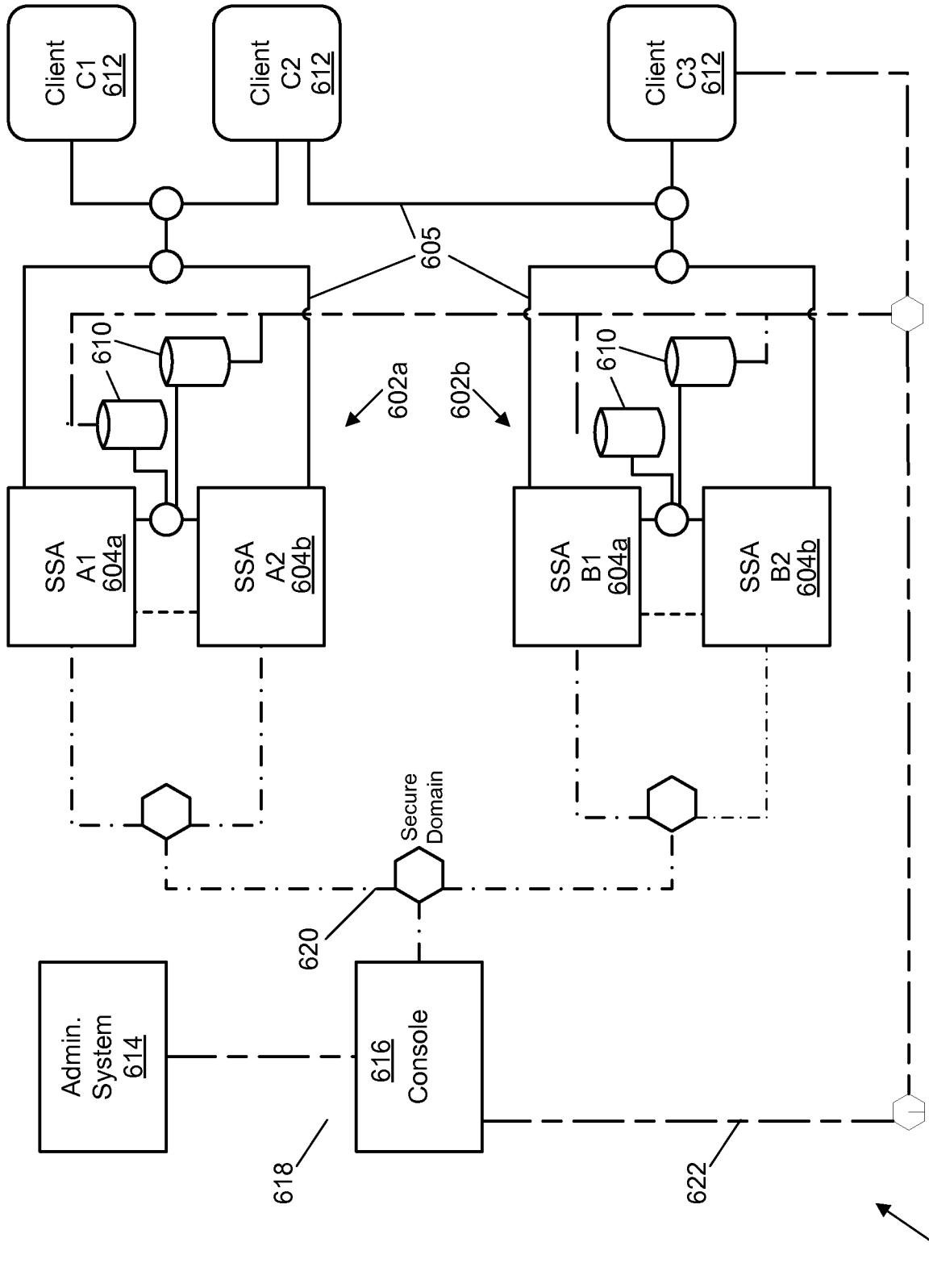
Client C1 612
Client C2 612
Client C3 612
605
610
610
602a
602b
SSA A1 604a
SSA A2 604b
SSA B1 604a
SSA B2 604b
Secure Domain
620
Admin. System 614
616 Console
618
622
600

FIG. 12

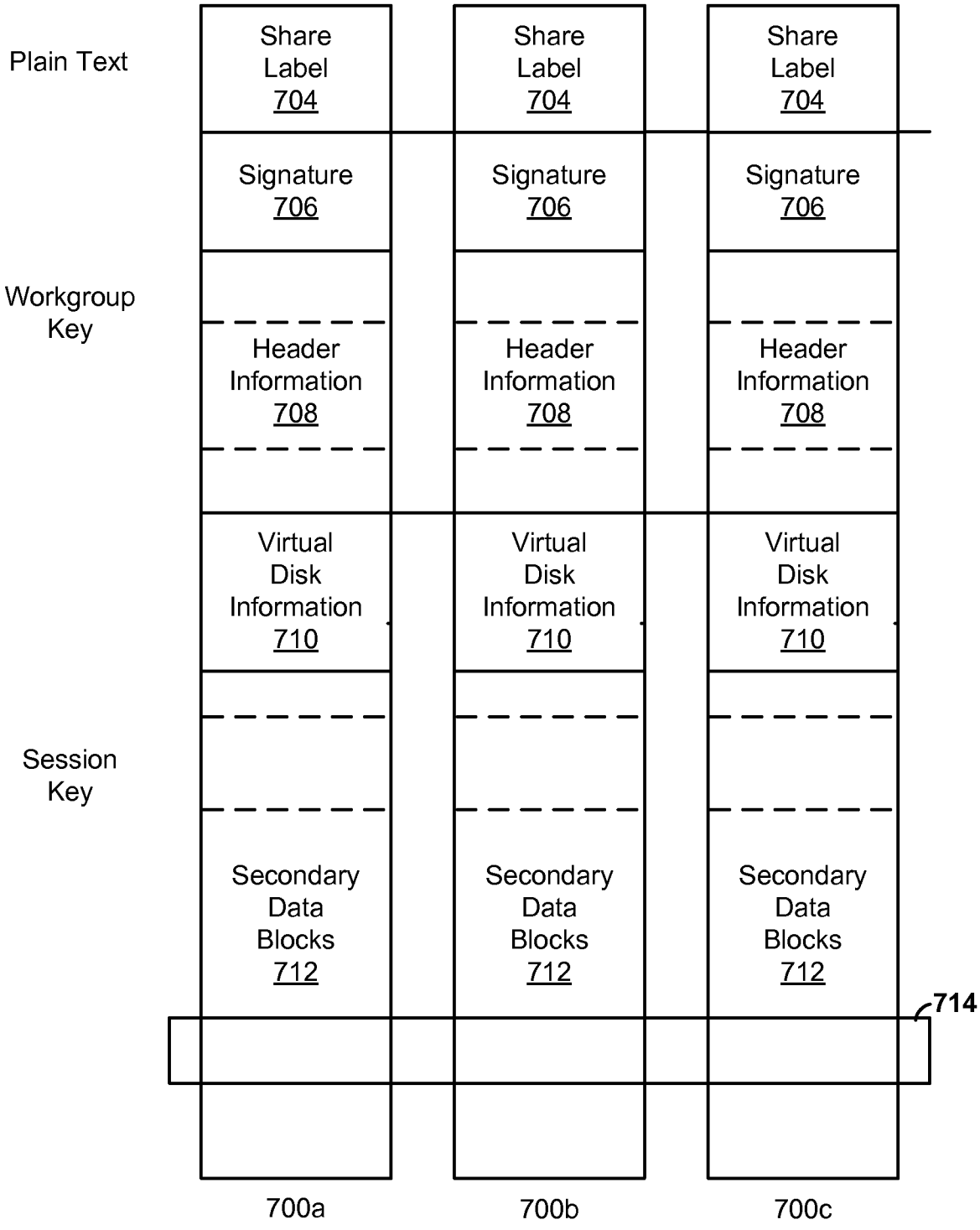
Plain Text
Workgroup Key
Session Key
Share Label 704
Signature 706
Header Information 708
Virtual Disk Information 710
Secondary Data Blocks 712
714
700a
700b
700c

FIG. 13

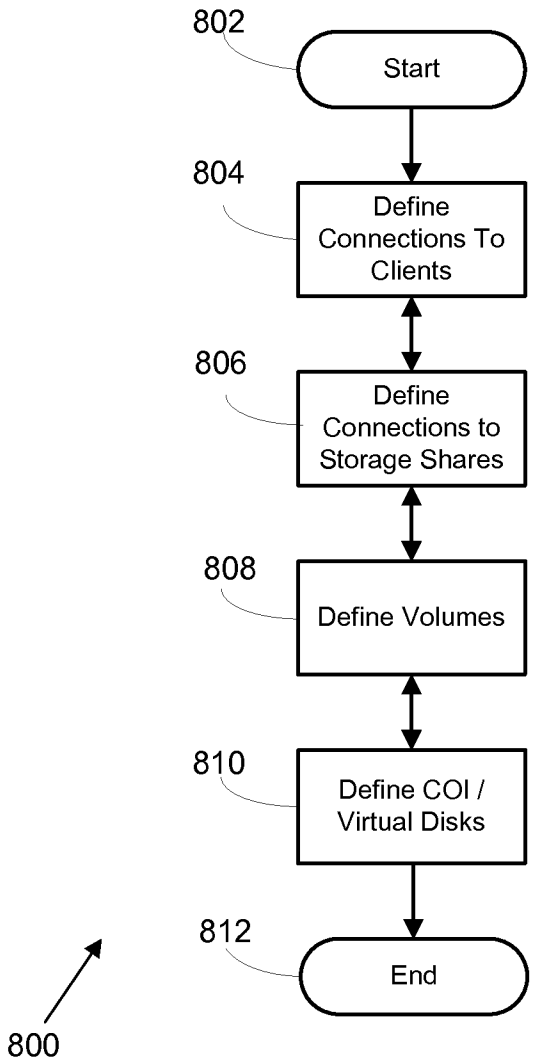
802
Start
804
Define Connections To Clients
806
Define Connections to Storage Shares
808
Define Volumes
810
Define COI / Virtual Disks
812
End
800

FIG. 14

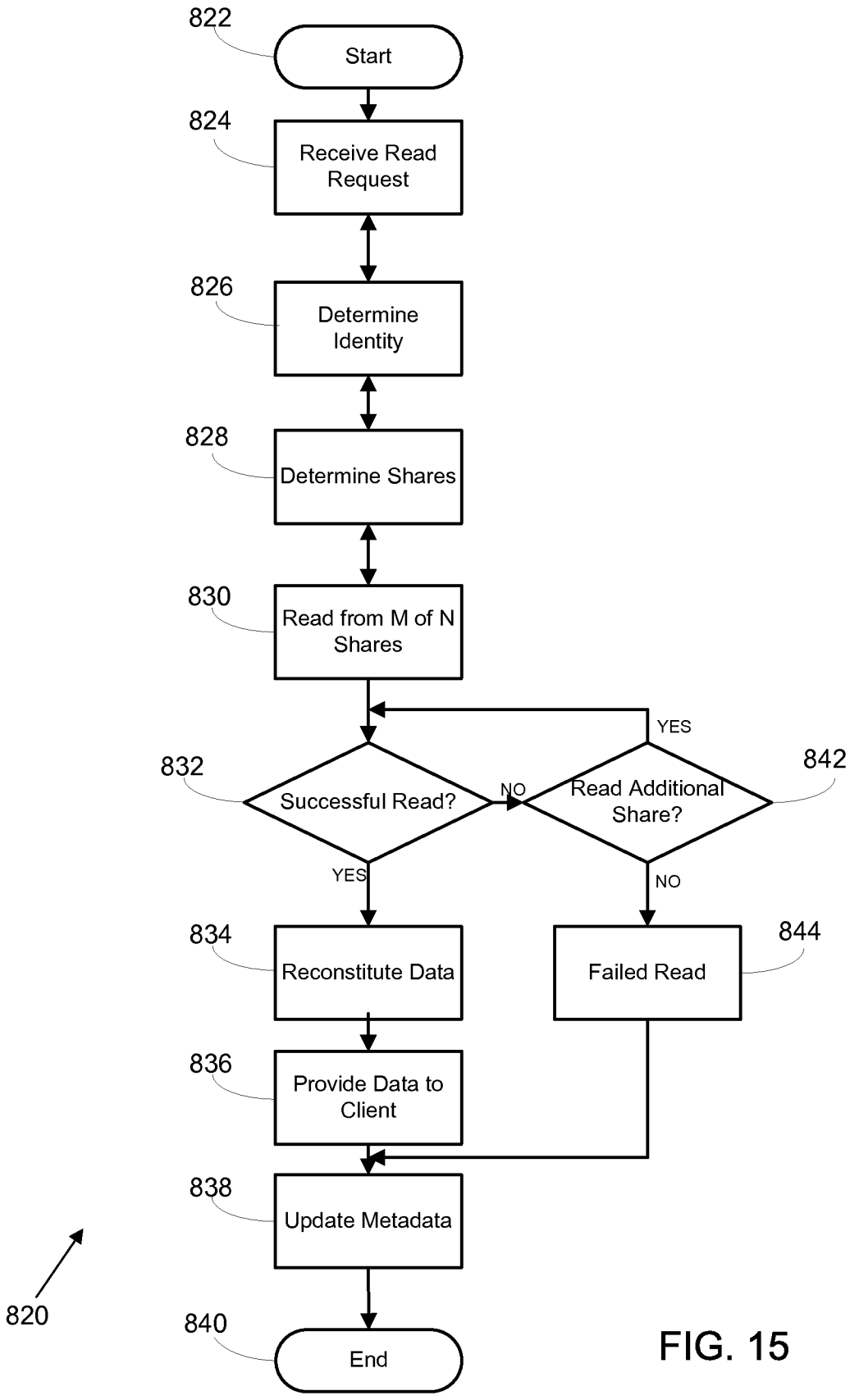
822
Start
824
Receive Read Request
826
Determine Identity
828
Determine Shares
830
Read from M of N Shares
832
Successful Read?
NO
YES
842
Read Additional Share?
YES
NO
834
Reconstitute Data
844
Failed Read
836
Provide Data to Client
838
Update Metadata
840
End
820

FIG. 15

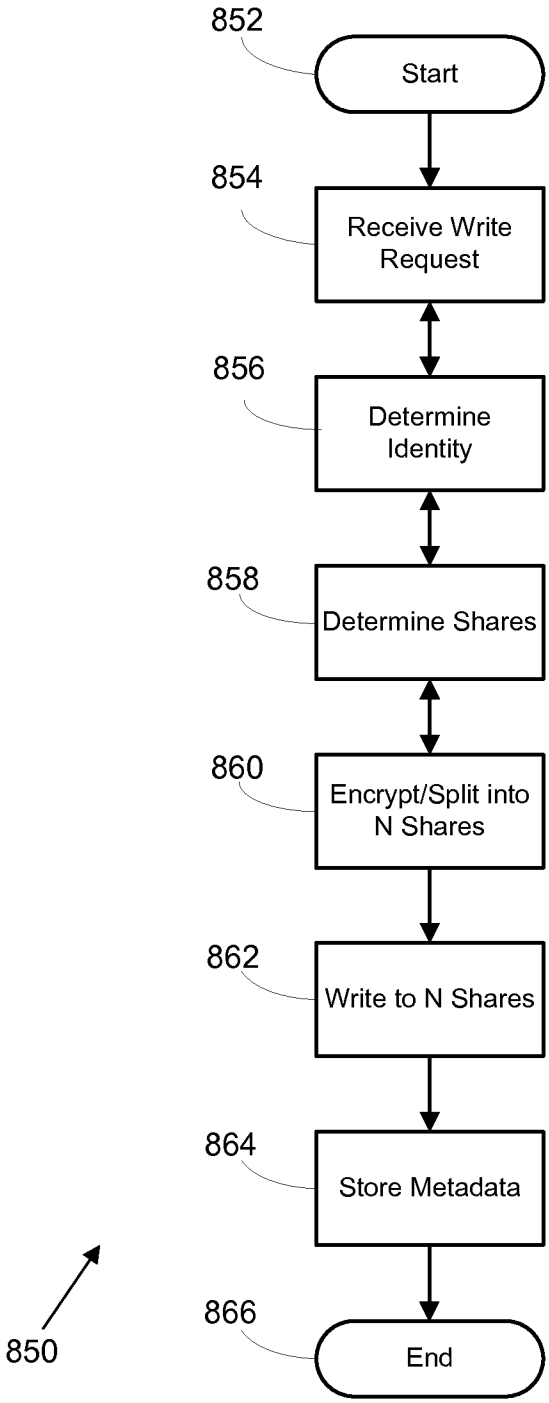
852
Start
854
Receive Write Request
856
Determine Identity
858
Determine Shares
860
Encrypt/Split into N Shares
862
Write to N Shares
864
Store Metadata
866
End
850

FIG. 16

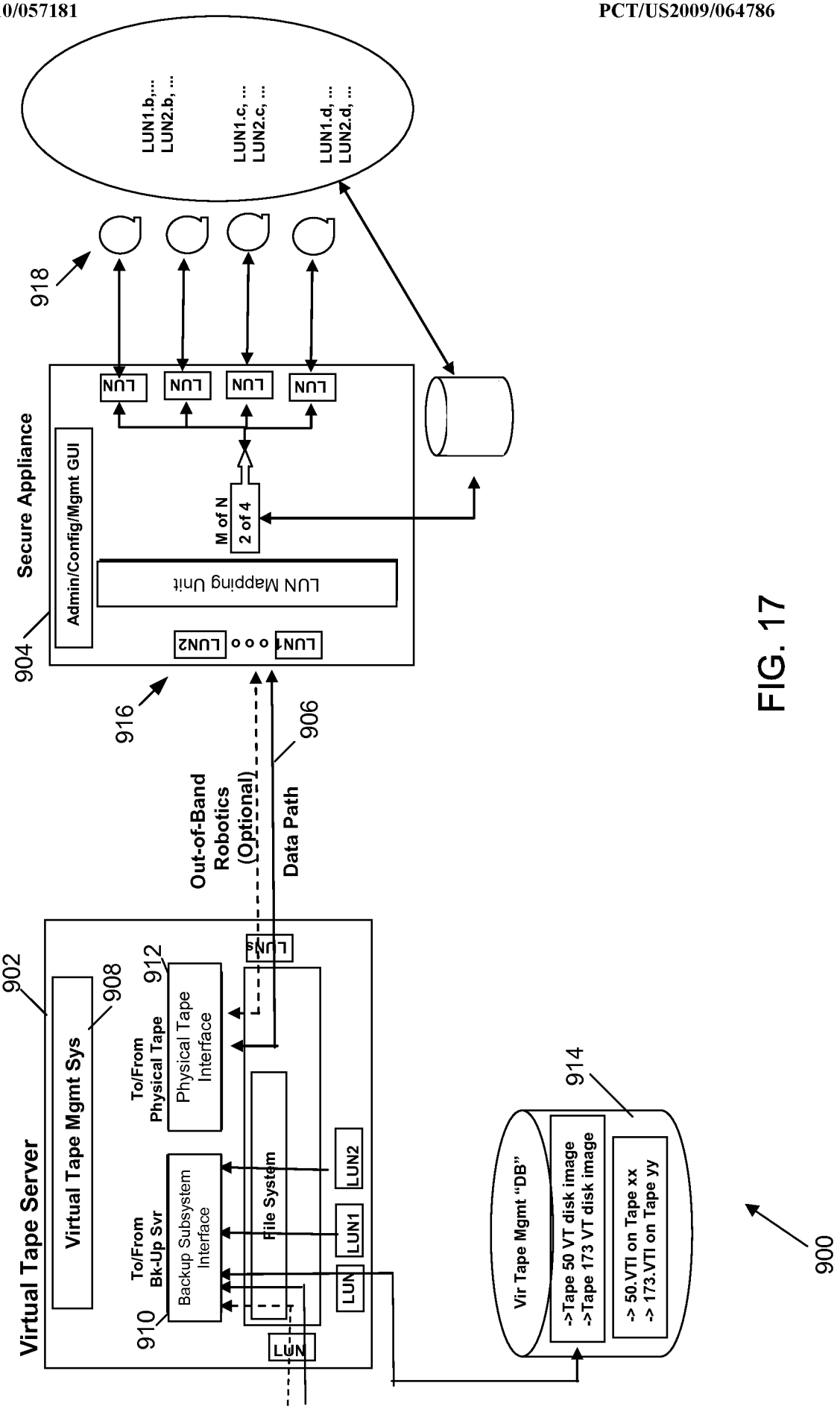
LUN1.b,...
LUN2.b, ...
LUN1.c, ...
LUN2.c, ...
LUN1.d, ...
LUN2.d, ...
918
LUN
LUN
LUN
LUN
Secure Appliance
Admin/Config/Mgmt GUI
M of N
2 of 4
LUN Mapping Unit
LUN1
LUN2
904
916
Out-of-Band
Robotics
(Optional)
Data Path
906
902
Virtual Tape Server
Virtual Tape Mgmt Sys
908
912
To/From
Physical Tape
Physical Tape
Interface
LUNs
To/From
Bk-Up Svr
Backup Subsystem
Interface
910
File System
LUN2
LUN1
LUN
LUN
914
Vir Tape Mgmt "DB"
->Tape 50 VT disk image
->Tape 173 VT disk image
-> 50.VTI on Tape xx
-> 173.VTI on Tape yy
900

FIG. 17

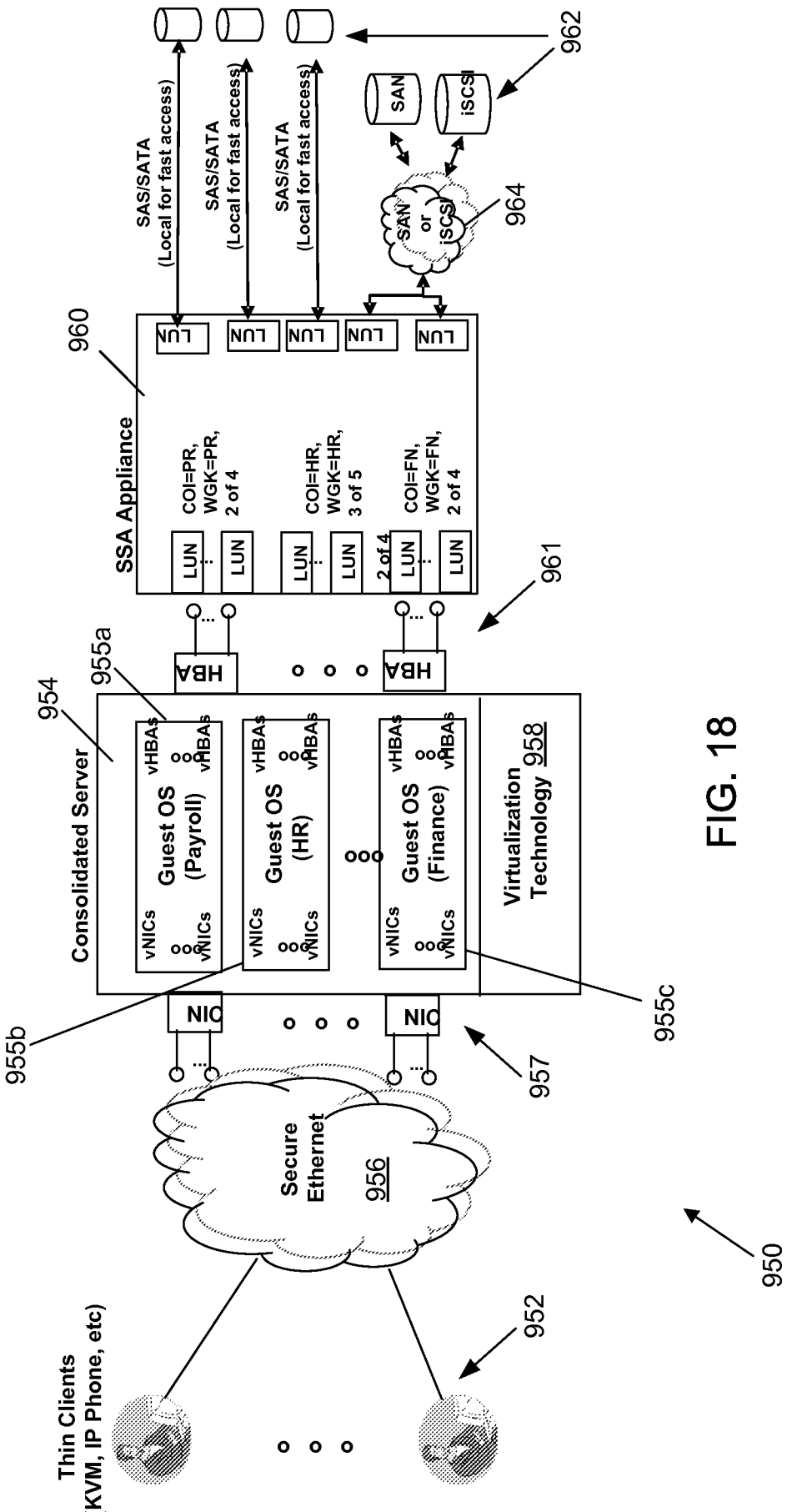
962
SAS/SATA (Local for fast access)
SAS/SATA (Local for fast access)
SAS/SATA (Local for fast access)
SAN
iSCSI
SAN or iSCSI
964
960
SSA Appliance
LUN
LUN
LUN
LUN
LUN
COI=PR, WGK=PR, 2 of 4
COI=HR, WGK=HR, 3 of 5
2 of 4
COI=FN, WGK=FN, 2 of 4
961
HBA
HBA
955a
954
Consolidated Server
vHBAs
Guest OS (Payroll)
vNICs
Guest OS (HR)
Guest OS (Finance)
Virtualization Technology 958
955b
955c
NIC
NIC
957
Secure Ethernet 956
950
952
Thin Clients (KVM, IP Phone, etc)

FIG. 18

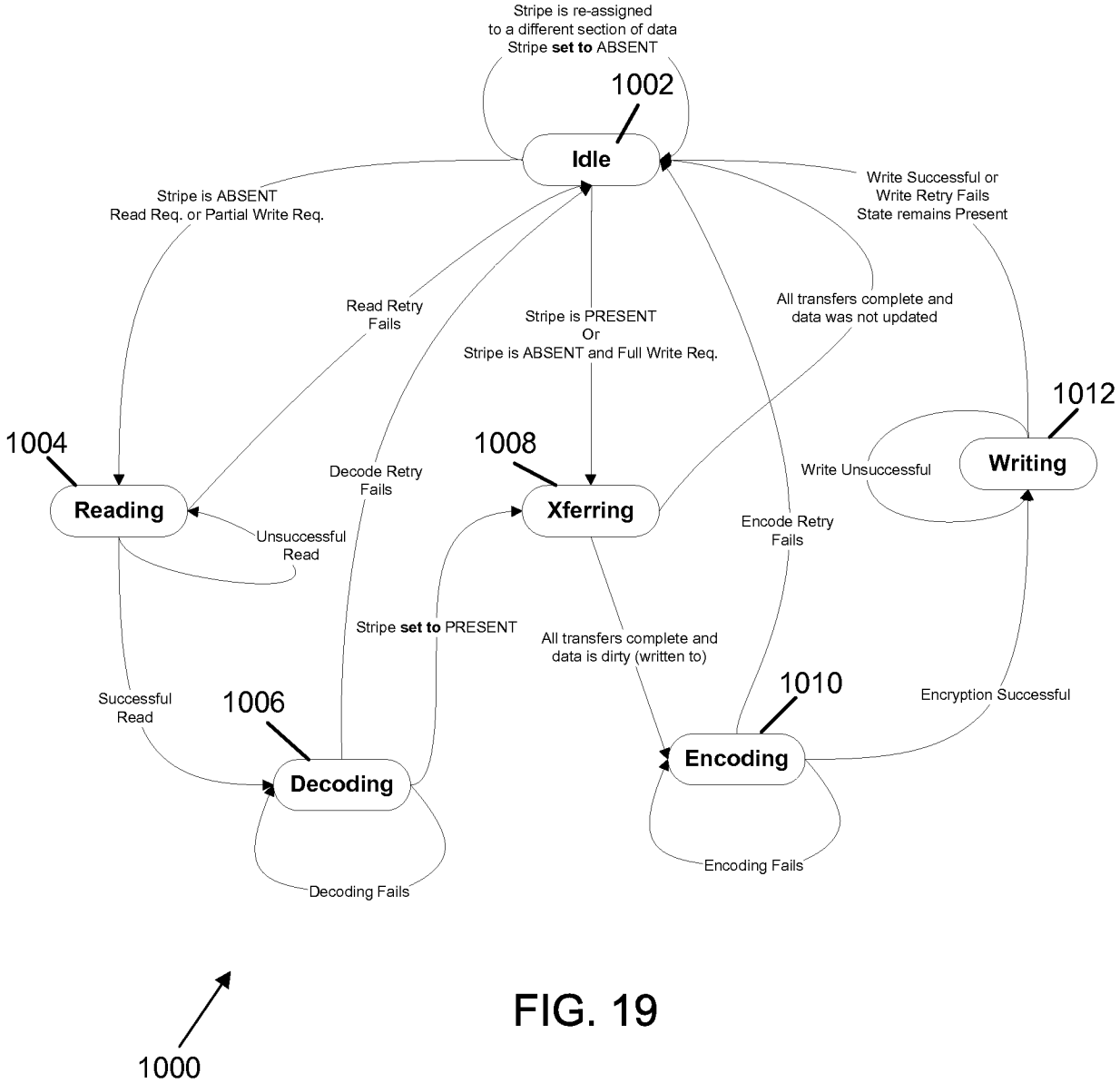
Stripe is re-assigned
to a different section of data
Stripe **set to** ABSENT
1002
Idle
Stripe is ABSENT
Read Req. or Partial Write Req.
Write Successful or
Write Retry Fails
State remains Present
Read Retry
Fails
Stripe is PRESENT
Or
Stripe is ABSENT and Full Write Req.
All transfers complete and
data was not updated
1012
1004
1008
Reading
Xferring
Writing
Decode Retry
Fails
Write Unsuccessful
Unsuccessful
Read
Encode Retry
Fails
Stripe **set to** PRESENT
All transfers complete and
data is dirty (written to)
Successful
Read
1006
1010
Encryption Successful
Decoding
Encoding
Decoding Fails
Encoding Fails
1000

FIG. 19

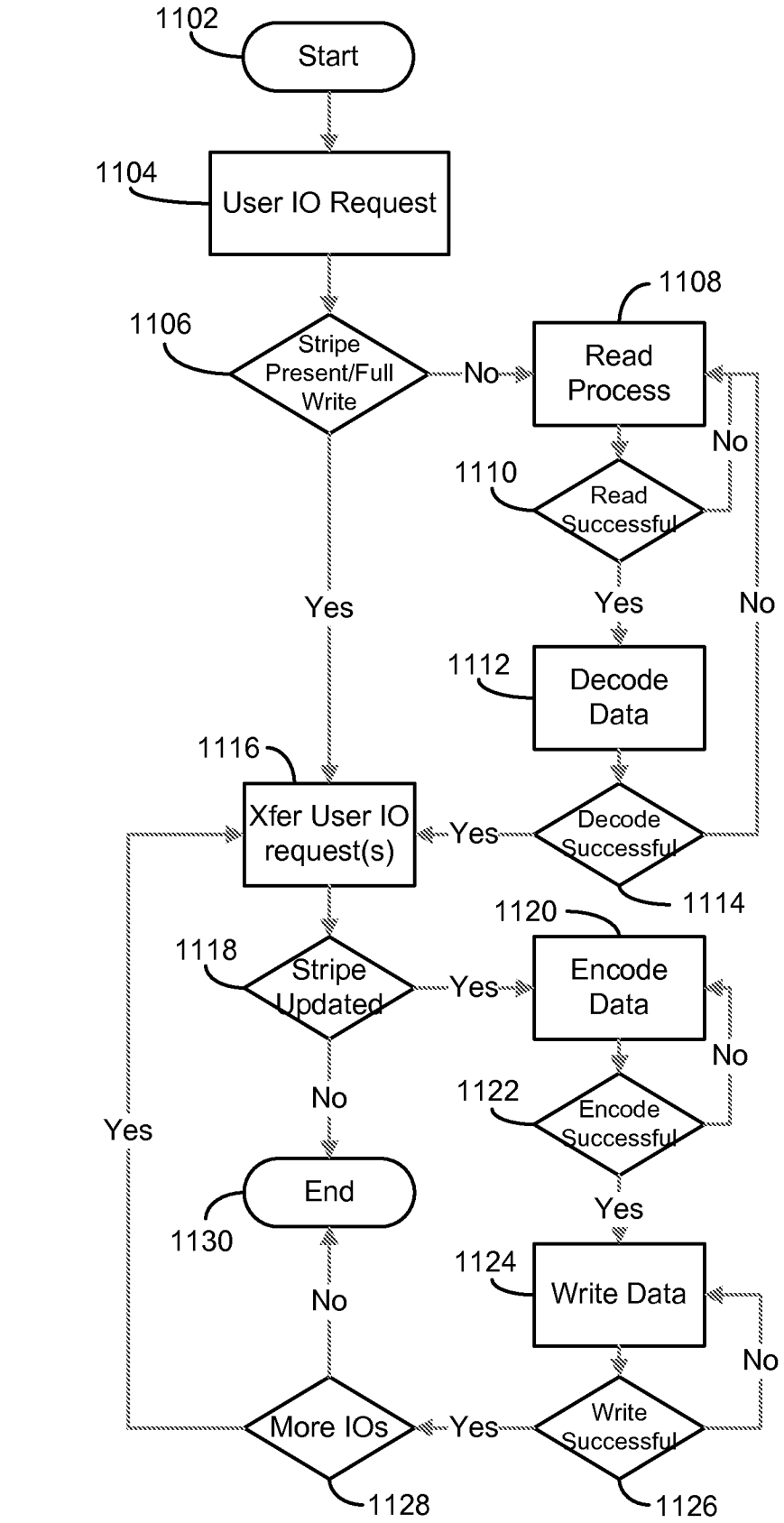
1102
Start
1104
User IO Request
1106
Stripe Present/Full Write
No
1108
Read Process
No
1110
Read Successful
Yes
No
Yes
1112
Decode Data
1116
Xfer User IO request(s)
Yes
Decode Successful
1114
1120
1118
Stripe Updated
Yes
Encode Data
No
1122
Encode Successful
No
Yes
Yes
End
1130
1124
Write Data
No
No
More IOs
Yes
Write Successful
1128
1126
1100

FIG. 20

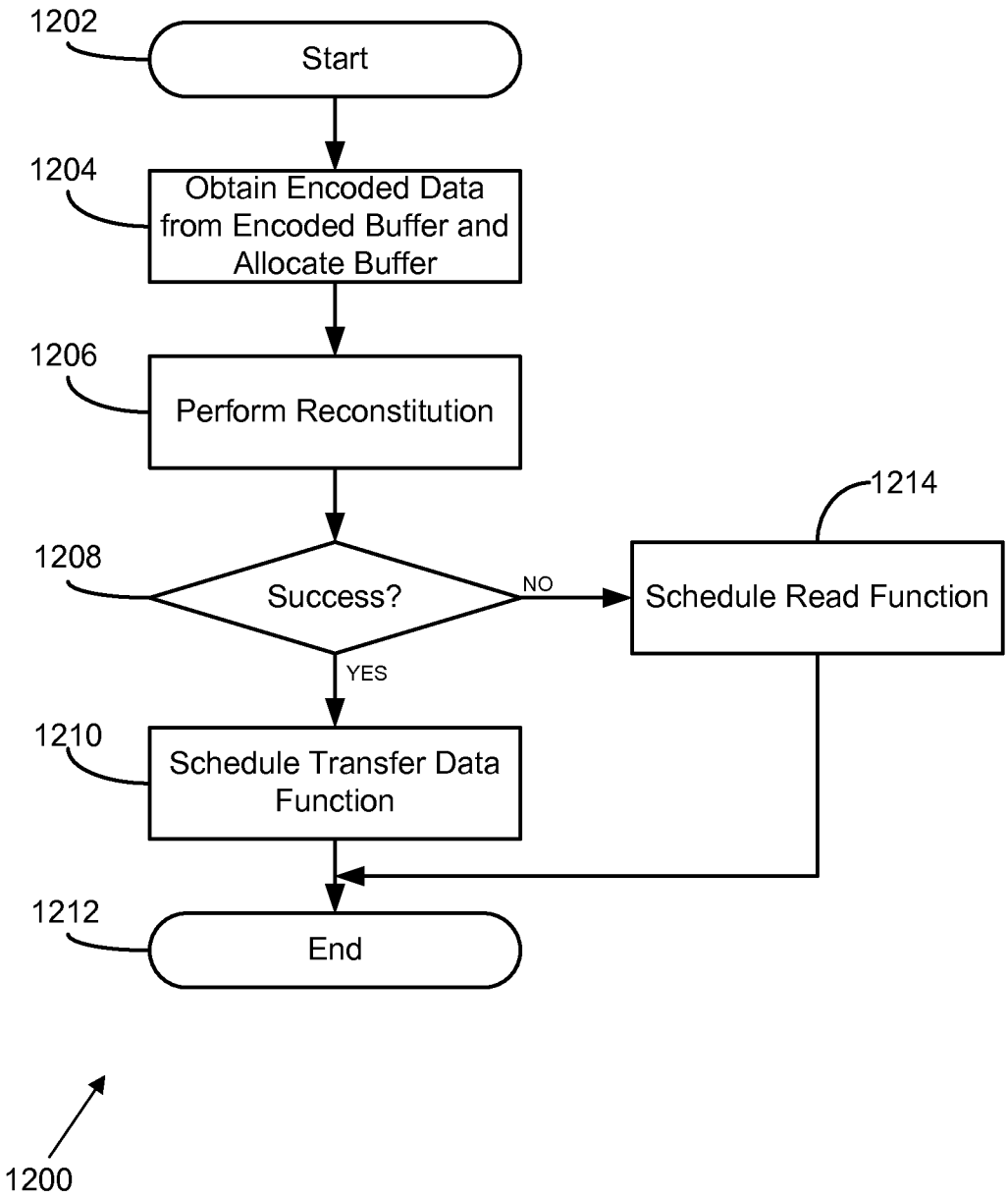
1202
Start
1204
Obtain Encoded Data from Encoded Buffer and Allocate Buffer
1206
Perform Reconstitution
1208
Success?
NO
YES
1214
Schedule Read Function
1210
Schedule Transfer Data Function
1212
End
1200

FIG. 21

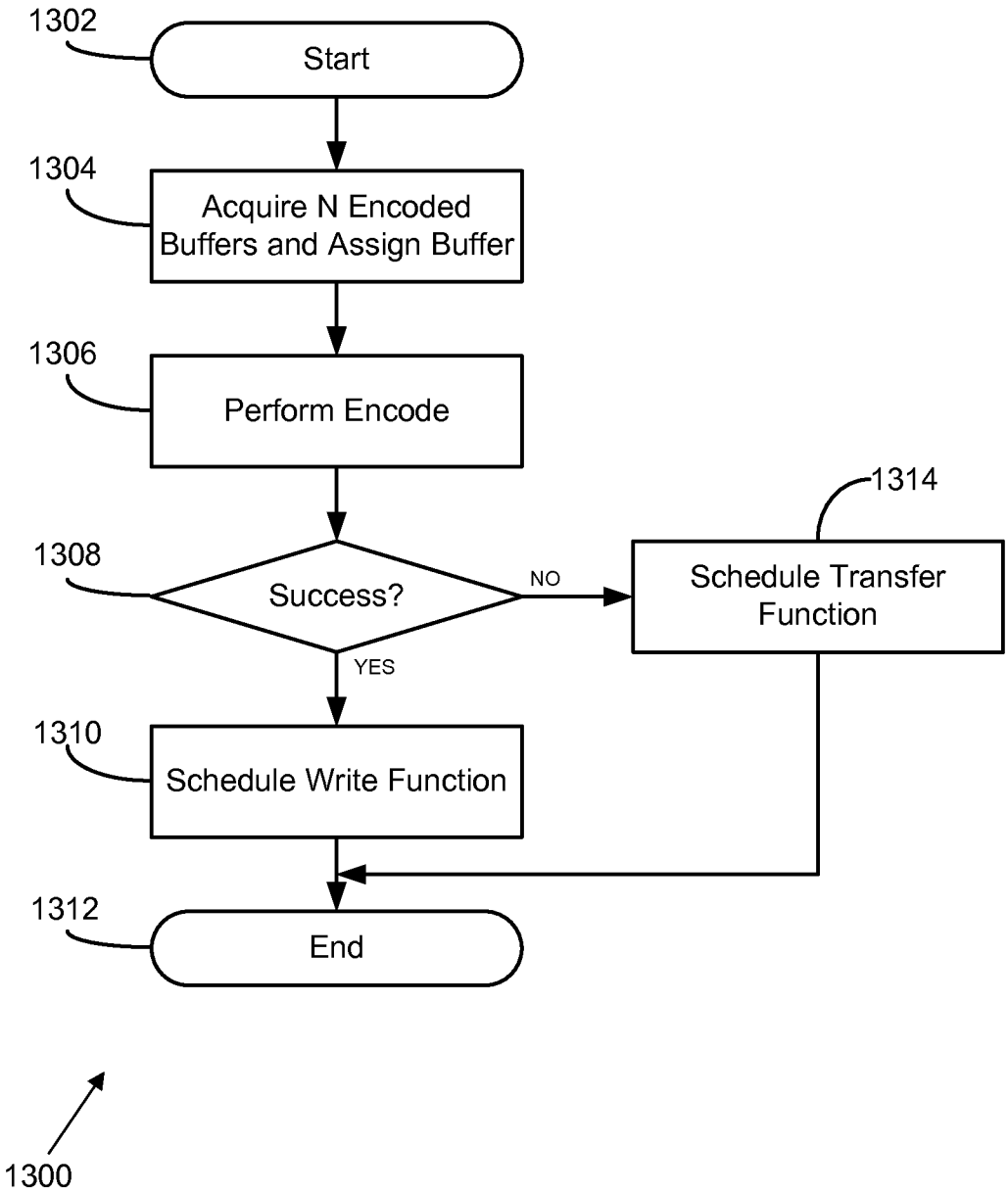
1302
Start
1304
Acquire N Encoded Buffers and Assign Buffer
1306
Perform Encode
1308
Success?
NO
YES
1314
Schedule Transfer Function
1310
Schedule Write Function
1312
End
1300

FIG. 22

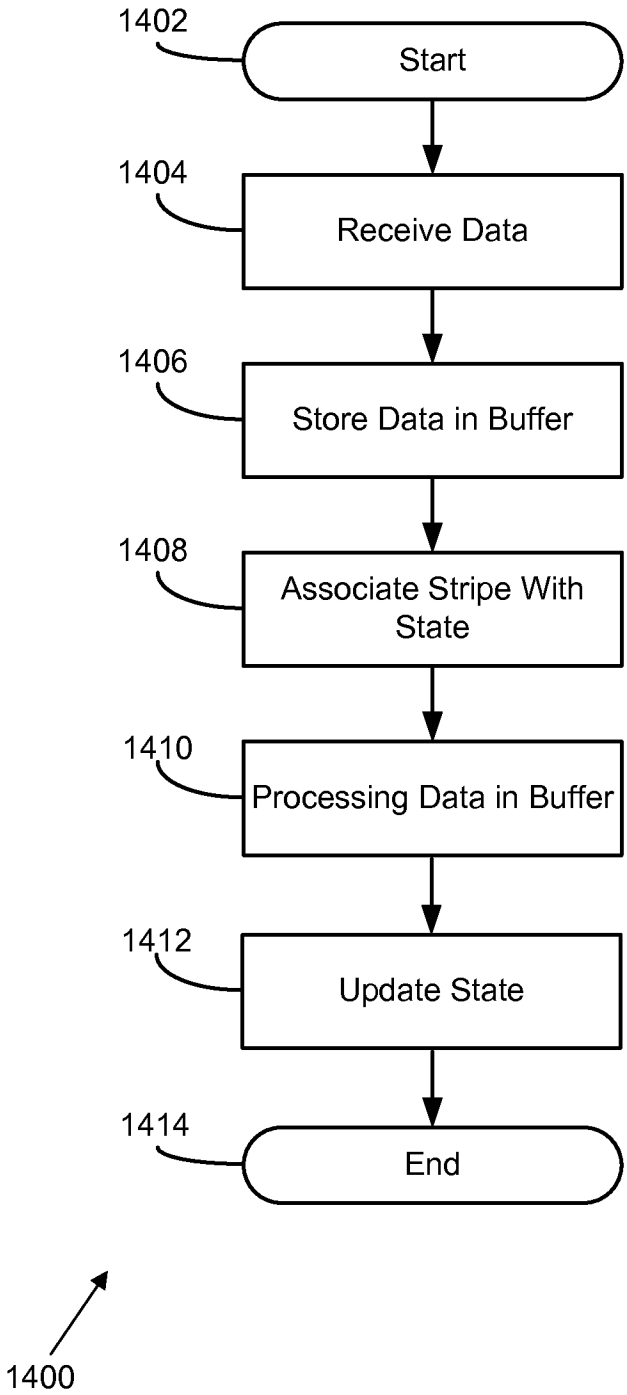
1402
Start
1404
Receive Data
1406
Store Data in Buffer
1408
Associate Stripe With State
1410
Processing Data in Buffer
1412
Update State
1414
End
1400

FIG. 23

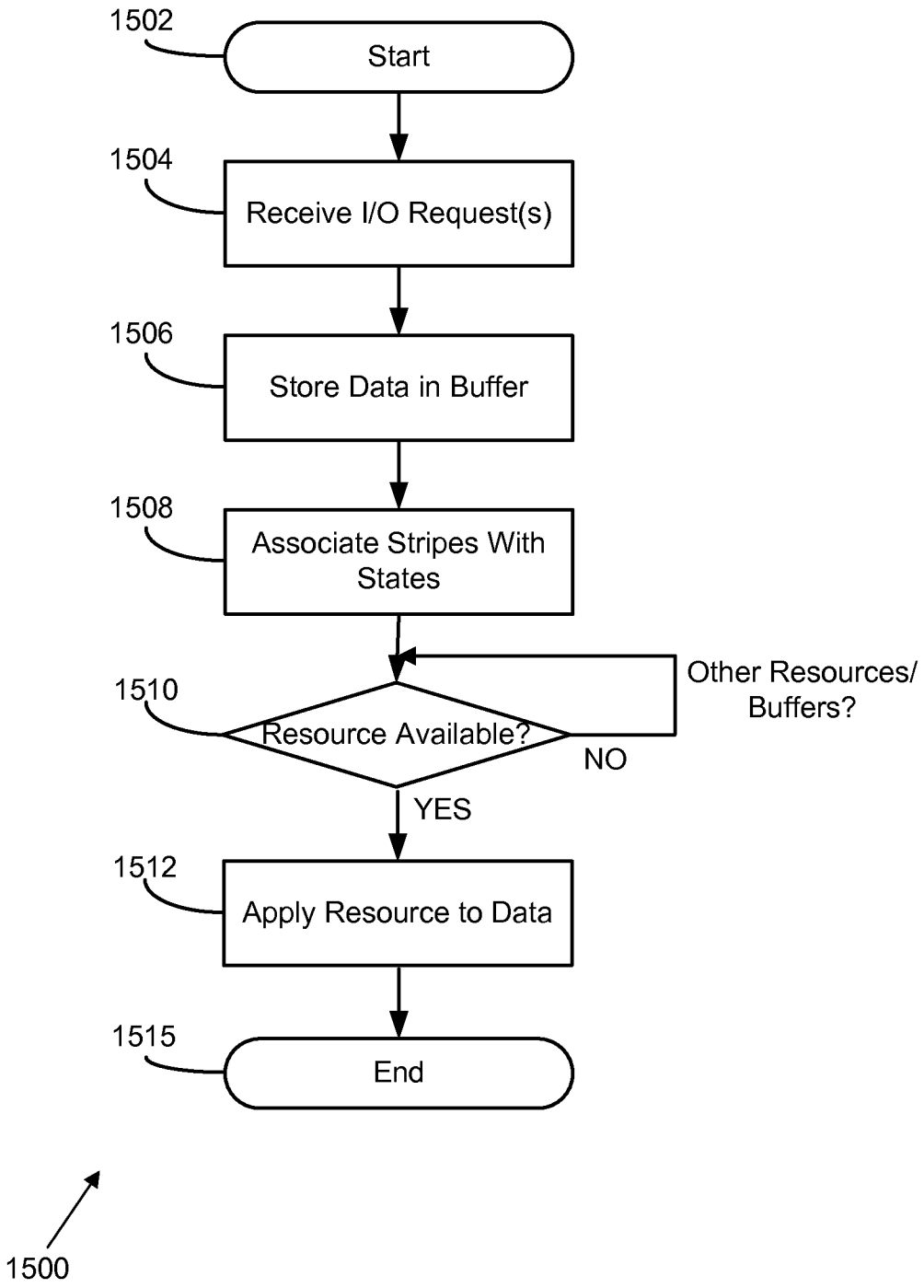
1502
Start
1504
Receive I/O Request(s)
1506
Store Data in Buffer
1508
Associate Stripes With States
1510
Resource Available?
Other Resources/ Buffers?
NO
YES
1512
Apply Resource to Data
1515
End
1500

FIG. 24