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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 October 2009 (08.10.2009)

PCT

(10) International Publication Number
WO 2009/123630 A1

(51) **International Patent Classification**:
G06F 15/16 (2006.01)

(21) **International Application Number**:
PCT/US2008/059072

(22) **International Filing Date**:
2 April 2008 (02.04.2008)

(25) **Filing Language**: English

(26) **Publication Language**: English

(71) **Applicant** *(for all designated States except US)*: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) **Inventors; and**

(75) **Inventors/Applicants** *(for US only)*: **FOSTER, Joseph** [US/US]; Hewlett-Packard Development Company, L.P., 20555 Tomball Parkway, Houston, Texas 77070 (US). **ELLIOT, Robert** [US/US]; Hewlett-Packard Development Company, L.P., 20555 Tomball Parkway, Houston, Texas 77070 (US). **PLANK, Jeffrey** [US/US]; Hewlett-Packard Development Company, L.P., 20555 Tomball Parkway, Houston, Texas 77070 (US).

(74) **Agents**: **JONES, Kevin** et al.; Hewlett-Packard Company, Intellectual Property Administration, Mail Stop 35, P.O. Box 272400, Fort Collins, Texas 80527 (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, 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, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, 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,

[Continued on next page]

(54) **Title**: DISK DRIVE DATA ENCRYPTION

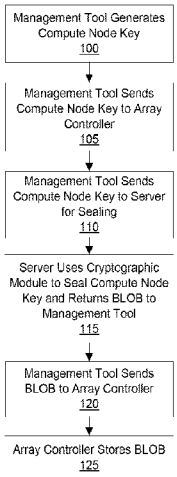

FIG. 1A

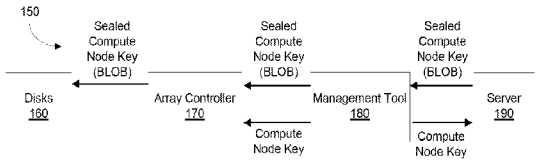

FIG. 1B

(57) **Abstract**: Embodiments include methods, apparatus, and systems for storage device data encryption. One method includes encrypting data on a storage device with a key and then transmitting the key to a cryptographic module that encrypts the key to form a Binary Large Object (BLOB). The BLOB is transmitted to an array controller that is coupled to the storage device which stores the BLOB.

WO 2009/123630 A1

MC, MT, NL, NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

— *with international search report (Art. 21(3))*

DISK DRIVE DATA ENCRYPTION

BACKGROUND

[001] Disk drives receive read and write requests from host computers in many storage systems. For security, data stored on the disk drives needs to be encrypted. In order to secure the data, important issues arise as to how to secure the data being stored and where to store the cryptographic keys.

[002] In some storage systems, the host computer sends both the cryptographic key and the data to the drive. In this operation, the host controls both the cryptographic keys and data flow.

[003] In other storage systems, the cryptographic keys are stored on a central key server. Timely and secure management of a large number of cryptographic keys in a single server is a difficult task. For instance, problems can arise since many different users or hosts need to be authenticated at the server in order to access the cryptographic keys. Further, if the cryptographic key is stored within the server, then the server that supports the corresponding disk array controller must be aware of the specific requirements associated with storing each cryptographic key.

[004] In still other storage systems, the cryptographic key is stored in the disk array controller. If the array controller fails, however, then the stored cryptographic key becomes trapped within the failed controller and the associated data stored on the disks is not accessible.

BRIEF DESCRIPTION OF THE DRAWINGS

[005] Figure 1A is a flow diagram for storing a cryptographic key in a disk drive in accordance with an exemplary embodiment of the present invention.

[006] Figure 1B is a block diagram of a system for storing a cryptographic key in a disk drive in accordance with an exemplary embodiment of the present invention.

[007] Figure 2A is a flow diagram for retrieving a cryptographic key from a disk drive in accordance with an exemplary embodiment of the present invention.

[008] Figure 2B is a block diagram of a system for retrieving a cryptographic key from a disk drive in accordance with an exemplary embodiment of the present invention.

[009] Figure 3A is another example of a flow diagram for storing a cryptographic key in a disk drive in accordance with an exemplary embodiment of the present invention.

[0010] Figure 3B is a block diagram of a system for storing a cryptographic key in a disk drive in accordance with an exemplary embodiment of the present invention.

[0011] Figure 4A is another example of a flow diagram for retrieving a cryptographic key from a disk drive in accordance with an exemplary embodiment of the present invention.

[0012] Figure 4B is a block diagram of a system for retrieving a cryptographic key from a disk drive in accordance with an exemplary embodiment of the present invention.

[0013] Figure 5 is a storage system for using encryption in accordance with an exemplary embodiment of the present invention.

[0014] Figure 6 is a computer or storage system for using encryption in accordance with an exemplary embodiment of the present invention.

DETAILED DESCRIPTION

[0015] Embodiments in accordance with the present invention are directed to apparatus, systems, and methods for automating cryptographic key management in storage systems. One exemplary embodiment securely acquires, delivers, and stores cryptographic keys used to encrypt and decrypt data in storage devices, such as disk drives. Further, exemplary embodiments protect removable media (such as hard disk drives) from unauthorized access when the removable media is moved or placed in an unauthorized appliance.

[0016] In one exemplary embodiment, a cryptographic key is provided from a disk controller to a key server. The cryptographic key is used to encrypt and decrypt data on the disk drive. Upon receiving the cryptographic key, the key server adds data to the cryptographic key and encrypts it to form a Binary Large Object (BLOB). The BLOB, now being sealed or encrypted, is then sent back to the controller which directs the BLOB to a disk drive for storage.

[0017] In order to unseal the BLOB, the disk controller transmits the sealed BLOB to the key server. Upon receiving the BLOB, the key server unseals the BLOB and returns the original cryptographic key back to the controller and disk drive. Data can now be read from and written to the disk drive using the cryptographic key.

[0018] Since the disk drive actually stores the BLOB, the disk drive can migrate from one controller to another while still providing access to the stored data. After a migration, the disk drive provides the BLOB back to the key server, and the key server returns to the original cryptographic key to the disk drive.

[0019] In one embodiment, the key server includes a cryptographic module (such as a Trusted Platform Module, TPM) to generate the BLOB. The key server does not permanently store the BLOB, but sends it to the disk drive for storage. As such, the key server is not burdened with storing state information (such as keys and BLOBS) for numerous disk drives. Instead, storage of the BLOBS occurs on the respective disk

drives, while the key server is used to convert to seal and unseal the BLOBS once received.

[0020] Each key server has a unique identity or identification. In one embodiment, this unique identity is used to seal and unseal the cryptographic keys received from the array controllers. Since an identity is unique to a specific key server, only the key server that sealed the BLOB can later unseal the BLOB and extract the original key. As an example, each key server can store various unique keys (for example, each customer or host having one or more unique keys) that are used to seal and unseal data for specific storage devices.

[0021] One embodiment stores a compute node key BLOB or key server key BLOB in the metadata area of the disk drive. For example, the BLOB for binding or sealing a RAID (Redundant Array of Independent/Inexpensive Disks) controller compute node key using the server TPM is stored in the non-user accessible metadata area of the disk drive. The metadata area of the disk drive is a section of the disk drive that is separate from application data and not accessible by users.

[0022] Figure 1A shows a flow diagram for storing/sealing a cryptographic key in a storage device, and Figure 1B shows a block diagram of a system 150 for storing/sealing the cryptographic key in the storage device in accordance with an exemplary embodiment of the present invention. For illustration, the storage device is shown as a disk drive, and the cryptographic key is shown as a compute node key.

[0023] In one embodiment, a volume access policy defines a set of rules or policies that for accessing the disk drives. If the volume access policy is set to include compute node, the encrypted disk data can only be properly decrypted if the RAID controller is in the appropriate server. If the volume access policy is set to include compute node configuration, the encrypted disk data can only be properly decrypted if the RAID controller is in the appropriate server with the server appropriately configured. The server and server configuration is verified with the use of the Trusted Platform Module BLOB sealing mechanism. There can be one or more compute node keys per RAID

controller (for example, multiple compute node keys exist in a blade environment). The design implementation allows for a different compute node key per storage volume.

[0024] In one embodiment, the compute node key does not leave the cryptographic boundary in plain text and is not stored in non-volatile memory. Instead, the key is encrypted before leaving the cryptographic boundary. The compute node key is regenerated by the server Trusted Platform Module on every power up initialization process.

[0025] According to block 100, a management tool 180 generates the compute node key. For example, the management tool (implemented in hardware and/or software) generates a 256 bit compute node key using a random number generator.

[0026] According to block 105, the management tool 180 sends the compute node key to the array controller 170. Once received, the array controller 170 adds the compute node key to the volume access policy.

[0027] According to block 110, the management tool 180 then sends the compute node key to the server 190 for sealing. By way of example, the management tool 180 transports the compute node key to a cryptographic module (such as a Trusted Platform Module) in the server 190 and requests that compute node key is sealed to create a BLOB.

[0028] According to block 115, the server 190 uses the cryptographic module to seal or encrypt the compute node key to create a BLOB. The sealed compute node key (now an encrypted BLOB) is returned to the management tool 180. By way of example, a TPM seals the compute node key and returns the resulting BLOB to the management tool.

[0029] According to block 120, the management tool 180 sends the BLOB to the array controller 170. In one embodiment, this exchange does not need to be encrypted because the BLOB itself does not reveal any cryptographic secrets.

[0030] According to block 125, the array controller 170 provides the BLOB to the disk drive 160 where the BLOB is stored. For example, array controller 170 stores the sealed compute node key in the metadata area volume specific information area for the appropriate volume(s).

[0031] Figure 2A shows a flow diagram for retrieving/unsealing a cryptographic key in a storage device, and Figure 2B shows a block diagram of a system 250 for retrieving/unsealing the cryptographic key in the storage device in accordance with an exemplary embodiment of the present invention. Figure 2B shares like reference numerals with Figure 1B.

[0032] According to block 200, software code 260 executing on a host requests the array controller 170 for the sealed compute node key (i.e., BLOB). For illustration, the code is shown as executing option ROM code.

[0033] According to block 205, the array controller 170 retrieves the BLOB from the disk drive 160 and returns it to the host. For example, the RAID controller retrieves the appropriate sealed compute node key from the disk drive and returns it from the RAID controller NVRAM.

[0034] According to block 210, the executing option ROM code 260 presents the BLOB to a cryptographic module in the server 190 for unsealing. For example, the executing option ROM code transmits the BLOB to the Server Trusted Platform Module requesting that the compute node key be unsealed.

[0035] According to block 215, the cryptographic module unseals the BLOB and returns the original compute node key back to the executing option ROM code. If the RAID controller 170 is in the appropriate server (with the server appropriately configured) the server Trusted Platform Module will return to the executing option ROM code the original compute node key.

[0036] According to block 220, the executing option ROM code 260 sends the compute node to the array controller 170. Data can now be encrypted or decrypted to and from the disk drive.

[0037] Exemplary embodiments can also be used with a key server and corresponding key server key. For example, if the volume access policy is set to include a key server key (such as a 256 bit key), the encrypted disk data can only be decrypted if the array controller can communicate with the key server (i.e. on the appropriate network). In one embodiment, there is one key server key per volume, and the key server key is generated by the key server. Depending on the key server policy all volumes can use the same key server key, or each volume key server key can be unique.

[0038] The key server key is associated to a specific volume based on the volumes LUN (Logical Unit Number) Identifier. In one embodiment, the volume LUN Identifier is persistent even after volume migration to a foreign controller ensuring that the encrypted volumes continue to be accessible even after a drive migration (assuming controller access policy did not include a compute node key of a different server).

[0039] The key server key is stored in volatile memory, except on the key server where the key is stored in non-volatile memory. When the key server key crosses the cryptographic boundary, it is first encrypted with another key, such as a transport key. By way of example, the transport key is created with an unauthenticated Diffie-Hellman exchange (IKEv2-SCSI) between the array controller and the management tool.

[0040] The transport key is created in response to the need to send a key across the cryptographic boundary. Once a transport key has been established it is store in volatile memory on the array controller. The transport key can be used multiple times for multiple keys or passwords crossing the cryptographic boundary. The generation of a new transport key (to replace an existing transport key) can be requested at any time by the management tool.

[0041] In one embodiment, the array controller does not communicate directly with the key server. Instead, the array controller communicates with the management tool which in turn communicates with the key server. During the communication exchange, the array controller does not authenticate itself (on purpose). If volume access is to be restricted to a specific array controller, then the controller access policy is set to include the local key.

[0042] If the array controller makes a request to the key server (with the help of the management tool) for a key server key and the key server declines (for any reason) after some reasonable time out and number of retries, the volume associated with that key is disabled.

[0043] If the volume access policy is set to include the key server key then the encrypted disk data can only be decrypted if the array controller can communicate with the key server (i.e. on the appropriate network). In one embodiment, there is one key server key per volume, and the key server key is generated by the key server. Depending on the key server policy all volumes can use the same key server key or each volume key server key can be unique.

[0044] In one embodiment, the key server key does not leave the cryptographic boundary in plain text and is not stored in non-volatile memory (it is encrypted before leaving the cryptographic boundary). The key server key is regenerated by the key server on every power up initialization process.

[0045] According to block 300, a management tool 380 generates the key server key. For example, the management tool (implemented in hardware and/or software) generates a 256 bit key server key using a random number generator.

[0046] According to block 305, the management tool 380 sends the key server key to the array controller 370. Once received, the array controller 370 adds the key server key to the volume access policy.

[0047] According to block 310, the management tool 380 then sends the key server key to the server 390 for sealing. By way of example, the management tool 380 transports the key server key to a cryptographic module (such as a Trusted Platform Module) in the server 390 and requests that key server key is sealed to create a BLOB.

[0048] According to block 315, the server 390 uses the cryptographic module to seal or encrypt the key server key to create a BLOB. The sealed key server key (now an encrypted BLOB) is returned to the management tool 380. By way of example, a TPM seals the key server key and returns the resulting BLOB to the management tool.

[0049] According to block 320, the management tool 380 sends the BLOB to the array controller 370. In one embodiment, this exchange does not need to be encrypted because the BLOB itself does not reveal any cryptographic secrets.

[0050] According to block 325, the array controller 370 provides the BLOB to the disk drive 360 where the BLOB is stored. For example, array controller 370 stores the sealed key server key in the metadata area of the volume specific information area for the appropriate volume(s).

[0051] Figure 4A shows a flow diagram for retrieving/unsealing a cryptographic key in a storage device, and Figure 4B shows a block diagram of a system 450 for retrieving/unsealing the cryptographic key in the storage device in accordance with an exemplary embodiment of the present invention. Figure 4B shares like reference numerals with Figure 3B.

[0052] According to block 400, software code 460 executing on a host requests the array controller 370 for the sealed key server key (i.e., BLOB). For illustration, the code is shown as executing option ROM code.

[0053] According to block 405, the array controller 370 retrieves the BLOB from the disk drive 360 and returns it to the host. For example, the RAID controller retrieves the

appropriate sealed key server key from the disk drive and returns it from the RAID controller NVRAM.

[0054] According to block 410, the executing option ROM code 460 presents the BLOB to a cryptographic module in the server 390 for unsealing. For example, the executing option ROM code transmits the BLOB the Server Trusted Platform Module requesting that key server key be unsealed.

[0055] According to block 415, the cryptographic module unseals the BLOB and returns the original key server key back to the executing option ROM code. If the RAID controller 370 is in the appropriate server (with the server appropriately configured) the server Trusted Platform Module will return to the executing option ROM code the original key server key.

[0056] According to block 420, the executing option ROM code 460 sends the key server to the array controller 370. Data can now be encrypted or decrypted to and from the disk drive.

[0057] Embodiments in accordance with the present invention are utilized in a variety of systems, methods, and apparatus. For illustration, exemplary embodiments are discussed in connection with a hard disk drive storage system. Exemplary embodiments, however, are applicable to other types of storage systems, such as storage devices using tape cartridges, optical disks, or movable media.

[0058] Fig. 5 is a block diagram of an exemplary distributed file or storage system 500 in accordance with an exemplary embodiment of the invention. By way of example, the system is a storage network and/or a storage area network (SAN) that includes a plurality of host computers 502 and one or more storage devices or arrays 503A, 503B that include one or more storage controllers 504 (shown by way of example as an array controller), and a plurality of storage devices 506 (shown by way of example as disk array 1 to disk array N).

[0059] The host computers 502 (shown as host 1 to host N) are coupled to the array controllers 504 through one or more fabrics or networks 510, and the storage devices or arrays 503 are coupled to the storage devices 506 through one or more fabrics or networks 511. A management tool 517 couples to the networks 510 for communication with the array controllers 504 and one or more servers 519, such as a key server. For instance, the hosts communicate with an array controller using a small computer system interface (SCSI) or other interface/commands over a Fibre Channel (FC). By way of example, networks 510 and 511 include one or more of the Ethernet, Fibre Channel (FC), Serial Attached SCSI (SAS), iSCSI, internet, local area network (LAN), wide area network (WAN), public and/or private networks, etc. Communications links 512 are shown in the figure to represent communication paths or couplings between the hosts, controllers, storage devices, management tool, and servers.

[0060] In one exemplary embodiment, the array controller 504 and disk arrays 506 are network attached devices providing random access memory (RAM) and/or disk space (for storage and as virtual RAM) and/or some other form of storage such as magnetic memory (example, tapes), micromechanical systems (MEMS), or optical disks, to name a few examples. Typically, the array controller and disk arrays include larger amounts of RAM and/or disk space and one or more specialized devices, such as network disk drives or disk drive arrays, (example, redundant array of independent disks (RAID)), high speed tape, magnetic random access memory (MRAM) systems or other devices, and combinations thereof. In one exemplary embodiment, the array controller 504 and disk arrays 506 are memory nodes that include one or more servers.

[0061] The storage controller 504 manages various data storage and retrieval operations. Storage controller 504 receives I/O requests or commands from the host computers 502, such as data read requests, data write requests, maintenance requests, etc. Storage controller 504 handles the storage and retrieval of data on the multiple disk arrays 506 and disk groups. In one exemplary embodiment, storage controller 504 is a separate device or may be part of a computer system, such as a server. Additionally, the

storage controller 504 may be located with, proximate, or a great geographical distance from the disk arrays 506 or from each other.

[0062] The array controller 504 includes numerous electronic devices, circuit boards, electronic components, etc. By way of example, the array controller 504 includes firmware 520, an input/output (I/O) scheduler 522, a buffer or queue 524 (for example, used to temporarily store the metadata structures during ownership transfer), one or more interfaces 526, one or more processors 528 (shown by way of example as a CPU, central processing unit), and memory 530 (including read and write cache). CPU 528 performs operations and tasks necessary to manage the various data storage and data retrieval requests received from host computers 502. For instance, processor 528 is coupled to a host interface 526A that provides bidirectional data communications to one or more host computers 502. Processor 528 is also coupled to an array interface 526B that provides bidirectional data communications to the disk arrays 506.

[0063] Memory 530 is also coupled to processor 528 and stores various information used by processor when carrying out its tasks. By way of example, memory 530 includes one or more of volatile memory, non-volatile memory, or a combination of volatile and non-volatile memory. The memory 530, for example, stores applications, data, control programs, algorithms (including software to implement or assist in implementing embodiments in accordance with the present invention), and other data associated with the storage device (example, state data such as mapping metadata, configuration metadata, and cached user data). The processor 528 communicates with memory 530, interfaces 526, and the other components via one or more buses 532.

[0064] In at least one embodiment, the storage devices are fault tolerant by using existing replication, disk logging, and disk imaging systems and other methods including, but not limited to, one or more levels of redundant array of inexpensive disks (RAID). Replication provides high availability when one or more of the disk arrays crash or otherwise fail. Further, in one exemplary embodiment, the storage devices provide memory in the form of a disk or array of disks where data items to be addressed are

accessed as individual blocks stored in disks (example, 512, 1024, 4096, etc. ... bytes each) or stripe fragments (4K, 16K, 32K, etc. ... each).

[0065] In one embodiment the storage devices 503A, 503B are disk arrays. Each disk array can have one or more controllers. For instance, an array has two controllers for redundancy. Further, the storage devices include both production disks and backup disks as discussed herein.

[0066] In one embodiment, storage devices 503A, 503B are physically located in a same data center. In another embodiment, the storage devices are located a great geographical distance apart in separate data centers. Further, although only two storage devices are shown, a SAN can include hundreds or thousands of such storage devices.

[0067] Figure 6 shows an exemplary embodiment of a computer or storage system 600. The system includes a computer 602 (such as key server) for using a cryptographic module 620. The computer 602 couples to at least one remote entity 654 via a network 652. The computer 602 may be, for example, a server, a desktop computer, a laptop computer or a mobile device. The computer 602 comprises a processor 640 coupled to at least one local entity 650. As used herein "local entities" refer to hardware/firmware/software entities that are internal to the computer 602 and "remote entities" refer to hardware/firmware/software entities that are external to the computer 602. Examples of local entities include but are not limited to an Operating System and peripherals such as a smartcard reader, a hard disk drive, network controller, and a graphics controller. Examples of remote entities include but are not limited to various portable and non-portable computers and/or electronic devices such as hosts, disk arrays, controllers, servers, main frame computers, distributed computing devices, laptops, and other electronic devices and systems whether such devices and systems are portable or non-portable.

[0068] The processor 640 also couples to a network interface 648 and memory 642 which stores an operating system (OS) 644 for the computer 602. As shown, the memory

642 may also store a Trusted Platform Module Software Stack 646 (TSS) which handles requests sent to a Trusted Platform Module (TPM) 620 coupled to the processor 640.

[0069] The TPM 620 is configured to provide cryptographic functions such as an RSA asymmetric algorithm for digital signature and for encryption, SHA-1 hashing, a Hash-based Message Authentication Code (HMAC) function, secure storage, random number generation, or other functions. The TPM 620 is implemented using software, firmware and/or hardware. The TPM components shown in Figure 6 have been generalized and are not all-inclusive. Also, TPM architectures and functions may possibly change over time as authorized by the Trusted Computing Group (TCG).

[0070] As shown in Figure 6, the TPM 620 comprises an input/output (I/O) interface 622 in communication with the processor 640. The I/O interface 622 couples to other TPM components such as cryptographic services 624, a random number source 626, asymmetric algorithms 628, storage 630 and Platform Configuration Registers (PCRs) 632. The cryptographic services 624 support functions such as hashing, digital signing, encryption and decryption. The random number source 626 generates random numbers for the cryptographic services 624. For example, in some embodiments, the cryptographic services 624 use random numbers to generate cryptographic keys. The asymmetric algorithms 628 enable the TPM 620 to perform asymmetric key operations. The storage 630 securely stores secrets (for example, cryptographic keys or other data) protected by the TPM 620. The PCRs 632 store information about the current state of the computer 602. For example, in some embodiments, the PCRs 632 store individual integrity measurements related to the computer 602 as well as sequences of integrity measurements.

[0071] Exemplary embodiments provide for a method to store the TPM generated BLOB independent of the server that housed the TPM (requiring no server specific support). As such, embodiments ensure access to the disk data even if the RAID controller that encrypted the data fails and is replaced by another encrypting RAID controller (i.e. the BLOB is not trapped or stored in the failed RAID controller, but stored in the drive).

Furthermore, exemplary embodiments do not require human intervention to perform a computer node key back up operation since the BLOB is stored by the RAID Controller on each drive of a multi-drive RAID set. In the event of a TPM migration from one server to another, the associated disk drives may also be migrated to the new server without the involvement of a cryptographic officer.

[0072] Exemplary embodiments protect the array controllers (both hardware and software based) from other RAID controllers. Where the RAID controller BLOB is stored is apparent because the feature set a specific controller will offer (i.e. the difficulty to migrate drives) and the fact that cryptographic applications usually disclose their key management architecture. Furthermore, since the BLOB is stored in the metadata portion of the drive, proof of authentication is the BLOB itself, rather than a certificate or private key.

[0073] Definitions:

[0074] As used herein and in the claims, the following words have the following definitions:

[0075] As used herein, a "BLOB" is encrypted data that is generated by a cryptographic module or processor, such as a TPM (for use in Protected Storage, or for saving context outside the TPM).

[0076] As used herein, the term "storage device" means any data storage device capable of storing data including, but not limited to, one or more of a disk array, a disk drive, a tape drive, a solid state drive, an optical drive, a SCSI device, or a Fibre Channel device. As used herein, a "disk array" or "array" is a storage system that includes plural disk drive, a cache, and controller. Arrays include, but are not limited to, networked attached storage (NAS) arrays, modular SAN arrays, monolithic SAN arrays, utility SAN arrays, and storage virtualization.

[0077] As used herein, "TPM" or "Trusted Platform Module" is a cryptographic processor implemented in accordance with the specifications defined in the TCG Trusted

Platform Module Specification. TPM provides various functions, such as secure generation of cryptographic keys, remote attestation, sealed storage, binding, and a hardware random number generator.

[0078] In one exemplary embodiment, one or more blocks in the flow diagrams are automated. In other words, apparatus, systems, and methods occur automatically. As used herein, the terms "automated" or "automatically" (and like variations thereof) mean controlled operation of an apparatus, system, and/or process using computers and/or mechanical/electrical devices without the necessity of human intervention, observation, effort and/or decision.

[0079] The flow diagrams in accordance with exemplary embodiments of the present invention are provided as examples and should not be construed to limit other embodiments within the scope of the invention. For instance, the blocks should not be construed as steps that must proceed in a particular order. Additional blocks/steps may be added, some blocks/steps removed, or the order of the blocks/steps altered and still be within the scope of the invention. Further, blocks within different figures can be added to or exchanged with other blocks in other figures. Further yet, specific numerical data values (such as specific quantities, numbers, categories, etc.) or other specific information should be interpreted as illustrative for discussing exemplary embodiments. Such specific information is not provided to limit the invention.

[0080] In the various embodiments in accordance with the present invention, embodiments are implemented as a method, system, and/or apparatus. As one example, exemplary embodiments are implemented as one or more computer software programs to implement the methods described herein. The software is implemented as one or more modules (also referred to as code subroutines, or "objects" in object-oriented programming). The location of the software will differ for the various alternative embodiments. The software programming code, for example, is accessed by a processor or processors of the computer or server from long-term storage media of some type, such as a CD-ROM drive, flash memory, or hard drive. The software programming code is

embodied or stored on any of a variety of known media for use with a data processing system or in any memory device such as semiconductor, magnetic and optical devices, including a disk, hard drive, CD-ROM, ROM, flash memory, etc. The code is distributed on such media, or is distributed to users from the memory or storage of one computer system over a network of some type to other computer systems for use by users of such other systems. Alternatively, the programming code is embodied in the memory and accessed by the processor using the bus. The techniques and methods for embodying software programming code in memory, on physical media, and/or distributing software code via networks are well known and will not be further discussed herein.

[0081] The above discussion is meant to be illustrative of the principles and various embodiments of the present invention. Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

CLAIMS

What is claimed is:

1) A method, comprising:

encrypting data on a storage device with a cryptographic key;

transmitting the cryptographic key to a cryptographic module that encrypts the key to form a Binary Large Object (BLOB);

transmitting the BLOB to an array controller that is coupled to the storage device; and

storing the BLOB on the storage device.

2) The method of claim 1 further comprising:

receiving a request to decrypt the data on the storage device;

in response to the request, transmitting the BLOB from the storage device to the cryptographic module that extracts the cryptographic key from the BLOB;

sending the cryptographic key, which is extracted from the BLOB, to the array controller;

decrypting the data on the storage device with the cryptographic key.

3) The method of claim 1 further comprising, storing the BLOB in an accessible metadata portion of the storage device.

4) The method of claim 1 further comprising, sealing the cryptographic key with a Trusted Platform Module (TPM).

5) The method of claim 1 further comprising, regenerating the cryptographic key on every power-up initialization process of the storage device.

6) The method of claim 1, wherein the storage device is a disk drive.

7) The method of claim 1 further comprising:

moving the storage device to couple to another array controller;

allowing the data to be read from the storage device after the BLOB is unsealed at a server.

8) A tangible computer readable medium having instructions for causing a computer to execute a method, comprising:

storing a Binary Large Object (BLOB) on a storage device;

transmitting the BLOB from the storage device to a cryptographic module that extracts an encrypted key from the BLOB;

transmitting the cryptographic key to a controller that is coupled to the storage device; and

using the cryptographic key to encrypt or decrypt data stored on the storage device.

9) The tangible computer readable medium of claim 8 further comprising, generating the key with a random number generator.

10) The tangible computer readable medium of claim 8 further comprising:

sending the key back to the server after the key is used to encrypt or decrypt the data stored on the storage device;

using a cryptographic module at the server to form the BLOB;

transmitting the BLOB to the storage device for storage.

11) The tangible computer readable medium of claim 8 further comprising, storing the BLOB in a secure section of the storage device that is separate from application data and not accessible by users.

12) The tangible computer readable medium of claim 8 further comprising, permanently storing the BLOB only in the storage device and not a server or the controller.

13) The tangible computer readable medium of claim 8 further comprising, storing the BLOB by the controller on each storage device in a multi-drive Redundant Array of Independent/Inexpensive Disks (RAID) set.

14) The tangible computer readable medium of claim 8 further comprising, using the controller to set and enforce authentication policy for the storage device without requiring the server to store the BLOB.

15) The tangible computer readable medium of claim 8 further comprising, using an identity unique to the server to seal the key and form the BLOB.

16) A storage system, comprising:

a cryptographic module that seals a key to form a Binary Large Object (BLOB);

an array controller in communication with the cryptographic module; and

a storage device storing the BLOB and connected to the array controller, wherein the storage device transmits the BLOB to the cryptographic module which extracts the key and sends the key to the array controller to encrypt or decrypt data stored on the storage device.

17) The storage system of claim 16, wherein the cryptographic module is a Trusted Platform Module (TPM) located in a server.

18) The storage system of claim 16 further comprising, a software management tool that generates the key with a random number generator.

19) The storage system of claim 16, wherein the BLOB is stored only in the storage device.

20) The storage system of claim 16, wherein data stored on the storage device is accessible after the array controller fails since the BLOB is not stored in the array controller but in the storage device.

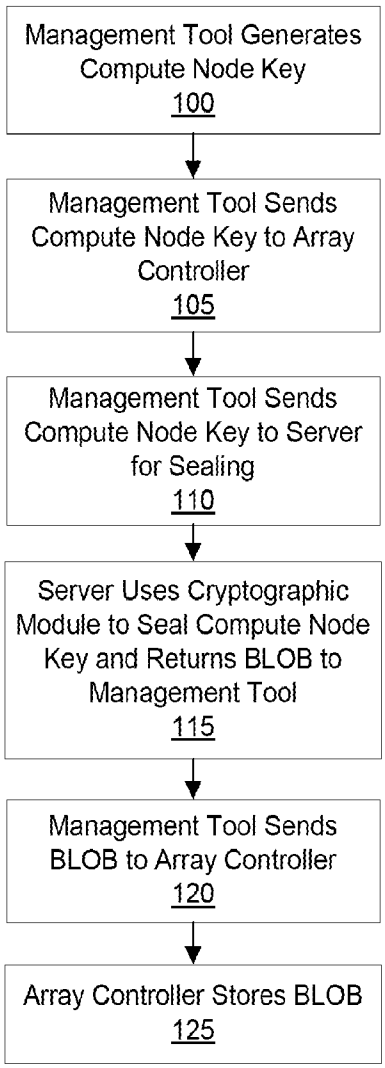
Management Tool Generates Compute Node Key
100
Management Tool Sends Compute Node Key to Array Controller
105
Management Tool Sends Compute Node Key to Server for Sealing
110
Server Uses Cryptographic Module to Seal Compute Node Key and Returns BLOB to Management Tool
115
Management Tool Sends BLOB to Array Controller
120
Array Controller Stores BLOB
125

FIG. 1A

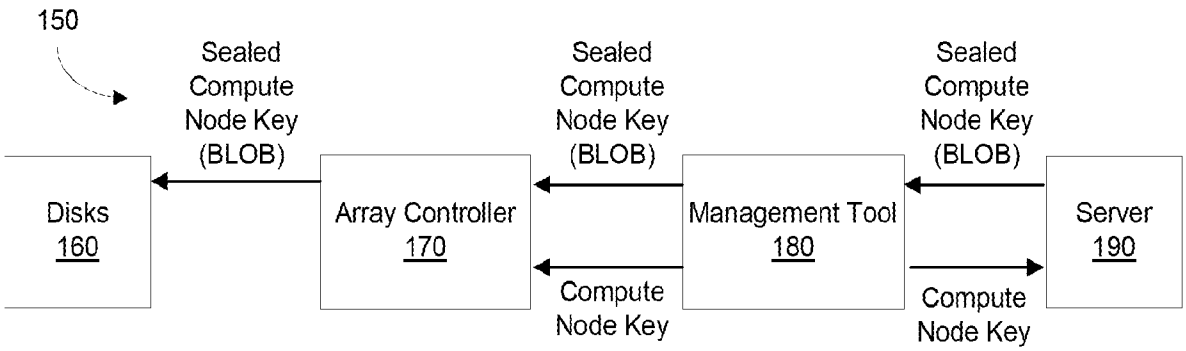
150
Sealed Compute Node Key (BLOB)
Sealed Compute Node Key (BLOB)
Sealed Compute Node Key (BLOB)
Disks
160
Array Controller
170
Management Tool
180
Server
190
Compute Node Key
Compute Node Key

FIG. 1B

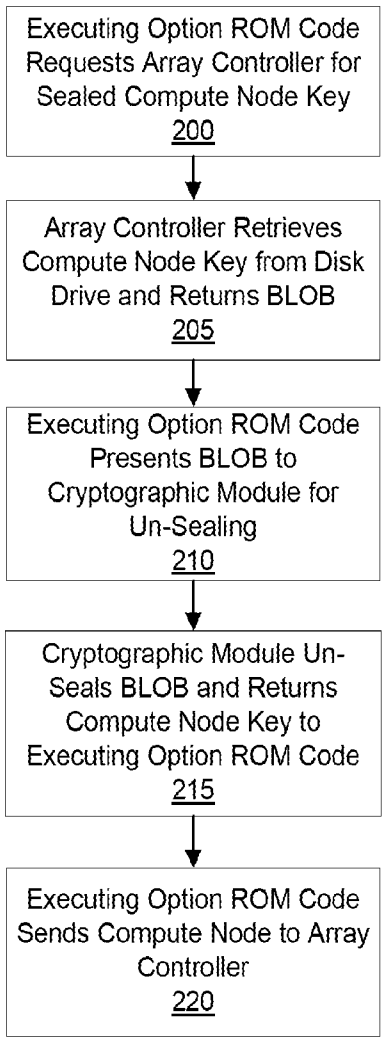
Executing Option ROM Code Requests Array Controller for Sealed Compute Node Key
200
Array Controller Retrieves Compute Node Key from Disk Drive and Returns BLOB
205
Executing Option ROM Code Presents BLOB to Cryptographic Module for Un-Sealing
210
Cryptographic Module Un-Seals BLOB and Returns Compute Node Key to Executing Option ROM Code
215
Executing Option ROM Code Sends Compute Node to Array Controller
220

FIG. 2A

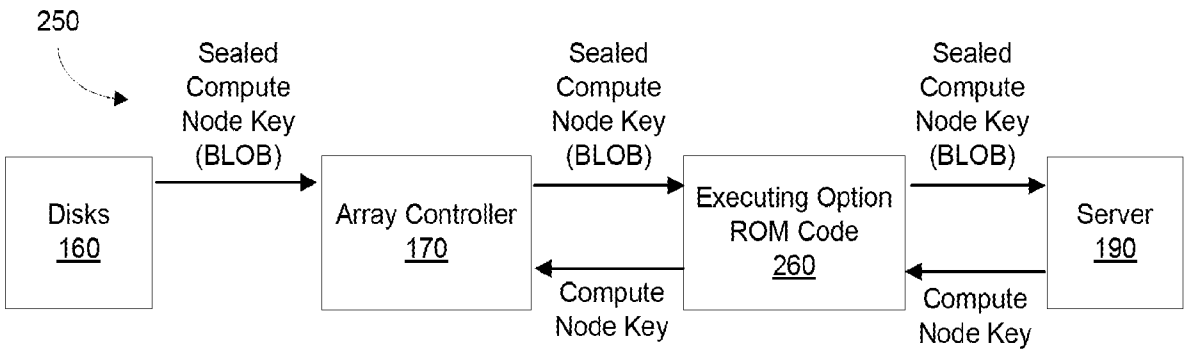
250
Sealed Compute Node Key (BLOB)
Sealed Compute Node Key (BLOB)
Sealed Compute Node Key (BLOB)
Disks
160
Array Controller
170
Executing Option ROM Code
260
Server
190
Compute Node Key
Compute Node Key

FIG. 2B

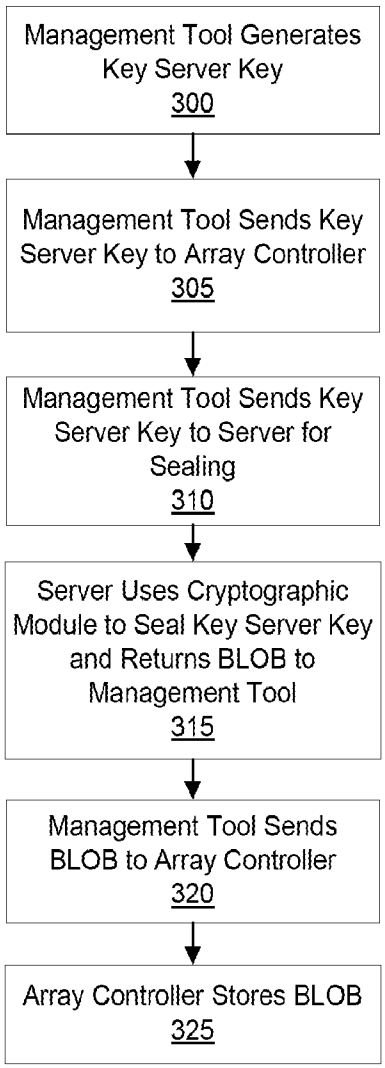
Management Tool Generates Key Server Key
300
Management Tool Sends Key Server Key to Array Controller
305
Management Tool Sends Key Server Key to Server for Sealing
310
Server Uses Cryptographic Module to Seal Key Server Key and Returns BLOB to Management Tool
315
Management Tool Sends BLOB to Array Controller
320
Array Controller Stores BLOB
325

FIG. 3A

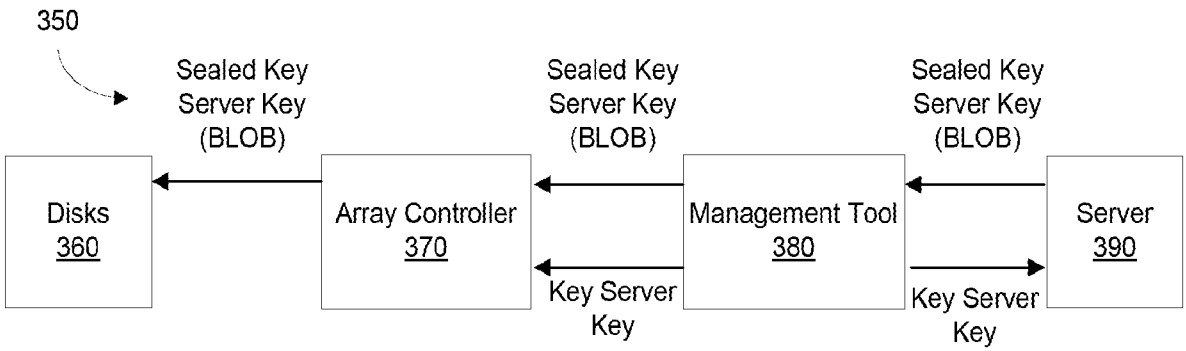
350
Sealed Key Server Key (BLOB)
Sealed Key Server Key (BLOB)
Sealed Key Server Key (BLOB)
Disks
360
Array Controller
370
Management Tool
380
Server
390
Key Server Key
Key Server Key

FIG. 3B

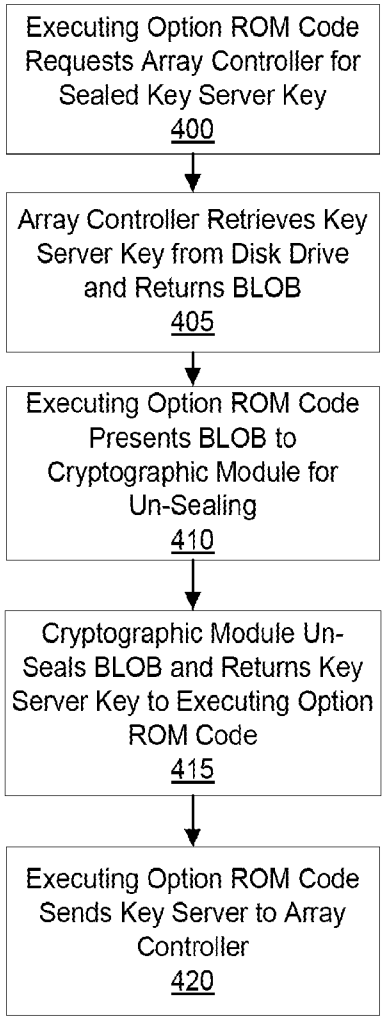
Executing Option ROM Code Requests Array Controller for Sealed Key Server Key
400
Array Controller Retrieves Key Server Key from Disk Drive and Returns BLOB
405
Executing Option ROM Code Presents BLOB to Cryptographic Module for Un-Sealing
410
Cryptographic Module Un-Seals BLOB and Returns Key Server Key to Executing Option ROM Code
415
Executing Option ROM Code Sends Key Server to Array Controller
420

FIG. 4A

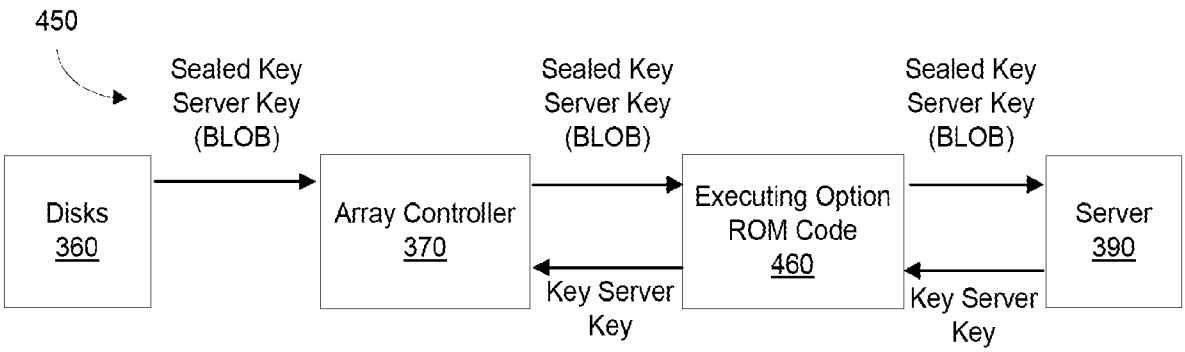
450
Sealed Key Server Key (BLOB)
Sealed Key Server Key (BLOB)
Sealed Key Server Key (BLOB)
Disks
360
Array Controller
370
Executing Option ROM Code
460
Server
390
Key Server Key
Key Server Key

FIG. 4B

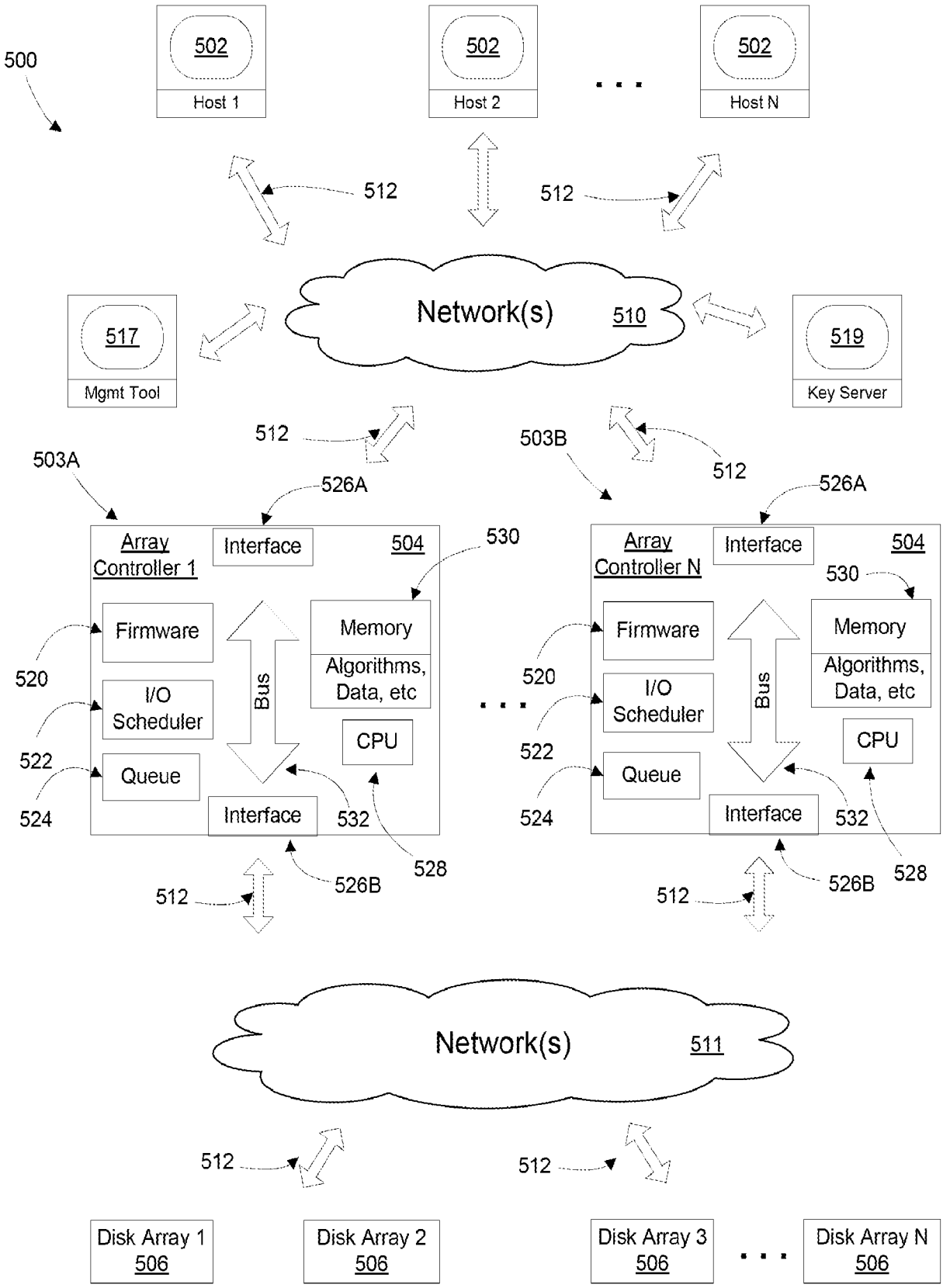
500
502
Host 1
502
Host 2
502
Host N
512
512
Network(s)
510
517
Mgmt Tool
519
Key Server
512
503A
526A
503B
512
526A
Array Controller 1
Interface
504
530
Array Controller N
Interface
504
530
Firmware
Memory
Algorithms, Data, etc
I/O Scheduler
Bus
CPU
Queue
Interface
520
522
524
532
528
526B
512
Network(s)
511
512
512
Disk Array 1 506
Disk Array 2 506
Disk Array 3 506
Disk Array N 506

FIG. 5

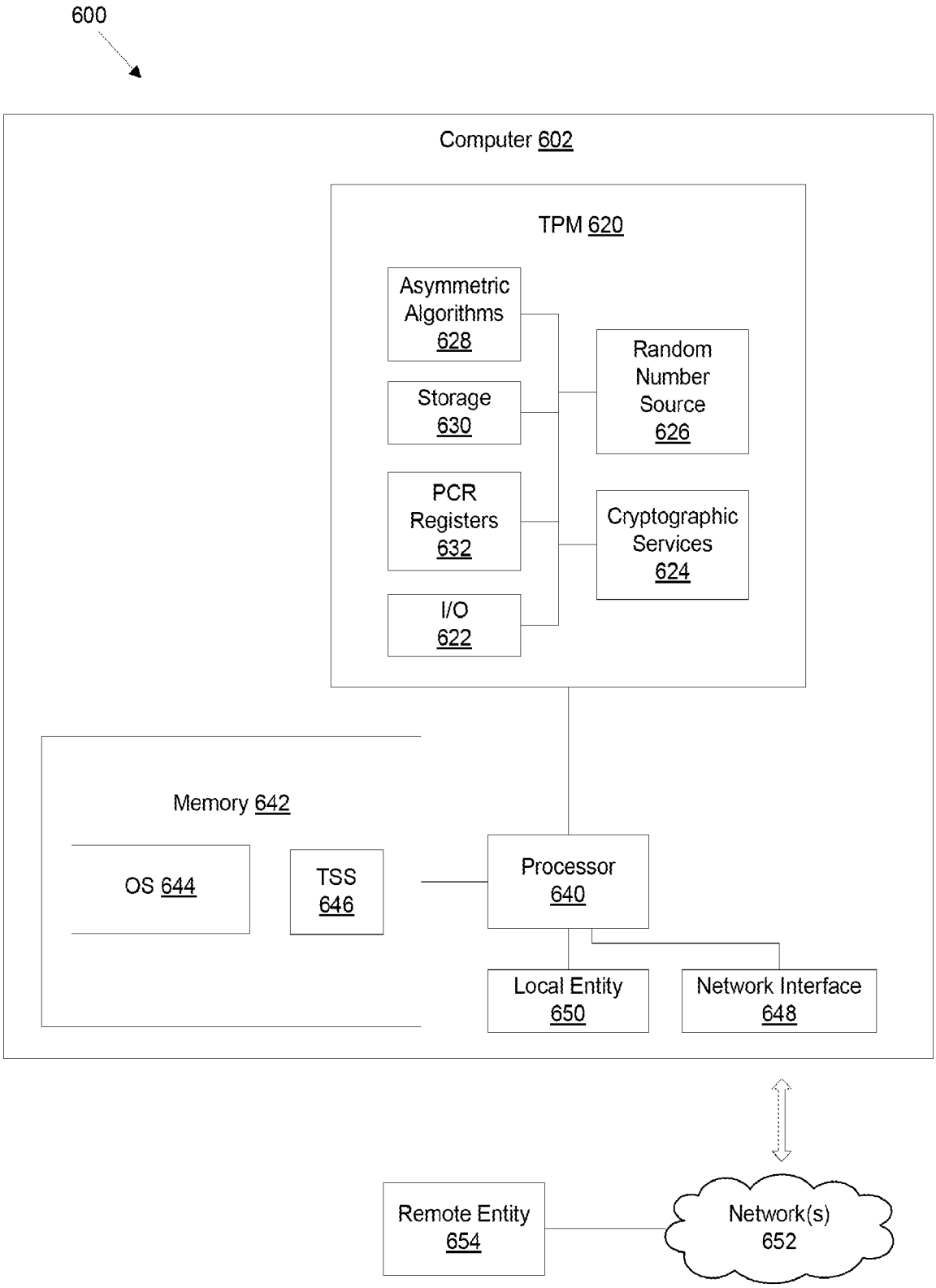
600
Computer 602
TPM 620
Asymmetric Algorithms 628
Storage 630
PCR Registers 632
I/O 622
Random Number Source 626
Cryptographic Services 624
Memory 642
OS 644
TSS 646
Processor 640
Local Entity 650
Network Interface 648
Remote Entity 654
Network(s) 652

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2008/059072

A. CLASSIFICATION OF SUBJECT MATTER

G06F 15/16(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC :G06F 12/14, G06F 15/16, H04L 9/00, H04L 9/32,

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

Korean Utility models and applications for Utility models since 1975

Japanese Utility models and applications for Utility models since 1975

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

e-KIPASS(KIPO internal)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2005/0138360 A1 (CHANDRA H. KAMALAKANTHA) Jun. 23, 2005. See Paragraph 27 - Paragraph 50; Paragraph 55 - Paragraph 57; Figures 1,4.	1-20
A	US 2003/0028664 A1 (KAIJUN TAN et al.) Feb. 6, 2003. See Paragraph 34 - Paragraph 36; Paragraph 38 - Paragraph 48; Paragraph 51 - Paragraph 58;Figures 1-5.	1-20
A	US 2004/0117625 A1 (DAVID W. GRAWROCK) Jun. 17, 2004. See Paragraph 20 - Paragraph 32; Paragraph 34 - Paragraph 40; Figures 2,4,5.	1-20
A	US 2008/0022412 A1 (DAVID C. CHALLENER) Jan. 24, 2008. See Paragraph 16 - Paragraph 26; Figures 1,2.	1-20

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:

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

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

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 MARCH 2009 (24.03.2009)	**25 MARCH 2009 (25.03.2009)**

Name and mailing address of the ISA/KR

Korean Intellectual Property Office
Government Complex-Daejeon, 139 Seonsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea

Facsimile No. 82-42-472-7140

Authorized officer

PARK, SANG HYUN

Telephone No. 82-42-481-8263

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2008/059072

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2005-0138360 A1	23.06.2005	AU 2004-313091 A1	21.07.2005
		CA 2547154 A1	21.07.2005
		EP 1698103 A1	06.09.2006
		WO 2005-067202 A1	21.07.2005
US 2003-0028664 A1	06.02.2003	NONE	
US 2004-0117625 A1	17.06.2004	AU 2003-290767 A1	29.07.2004
		CN 1514571 A	21.07.2004
		EP 1573468 A2	14.09.2005
		KR 10-2005-0085678 A	29.08.2005
		TW 255122 B	11.05.2006
		WO 2004-061628 A2	22.07.2004
		WO 2004-061628 A3	27.01.2005
US 2008-0022412 A1	24.01.2008	NONE	