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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 September 2015 (24.09.2015)

WIPO | PCT

(10) International Publication Number
WO 2015/142970 A1

(51) International Patent Classification:
G06F 21/74 (2013.01) *G06F 21/78* (2013.01)

(21) International Application Number:
PCT/US2015/021125

(22) International Filing Date:
18 March 2015 (18.03.2015)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
14/221,105 20 March 2014 (20.03.2014) US

(71) Applicant: **MICROSOFT TECHNOLOGY LICENSING, LLC** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **BASMOV, Innokentiy**; c/o Microsoft Technology Licensing, LLC, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **NYSTRÖM, Magnus Bo Gustaf**; c/o Microsoft Technology Licensing, LLC, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **FERGUSON, Niels T.**; c/o Microsoft Technology Licensing, LLC, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (US). **SEMENKO, Alex M.**; c/o Microsoft Technology Licensing, LLC, LCA - International Patents (8/1172), One Microsoft Way, Redmond, Washington 98052-6399 (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, BN, 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, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, 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,

[Continued on next page]

(54) Title: RAPID DATA PROTECTION FOR STORAGE DEVICES

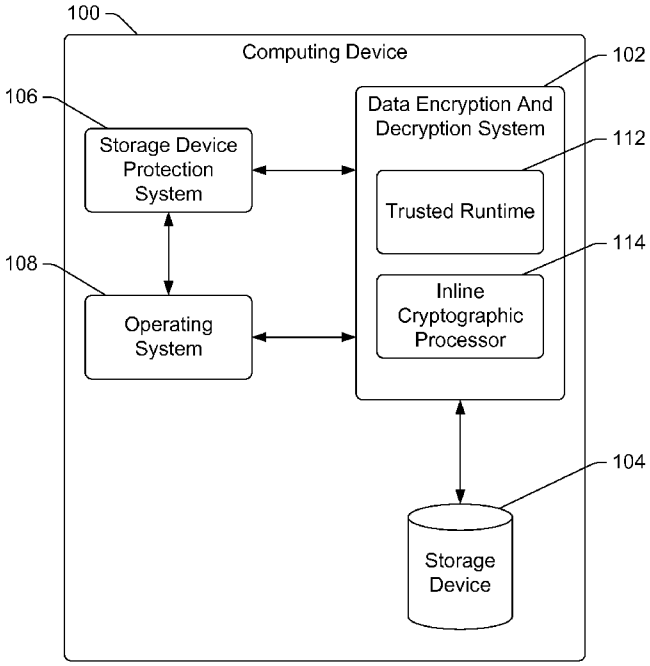

Fig. 1

(57) Abstract: A computing device uses a data encryption and decryption system that includes a trusted runtime and an inline cryptographic processor. The trusted runtime provides a trusted execution environment, and the inline cryptographic processor provides decryption and encryption of data in-line with storage device read and write operations. When a portion (e.g., partition) of a storage device is defined, the trusted runtime generates an encryption key and provides the encryption key to the inline cryptographic processor, which uses the encryption key to encrypt data written to the portion and decrypt data read from the portion. Access to the portion can be subsequently protected by associating the key with authentication credentials of a user or other entity. The trusted runtime protects the encryption key based on an authentication key associated with the authentication credentials, allowing subsequent access to the encryption key only in response to the proper authentication credentials being provided.

WO 2015/142970 A1

GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, 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, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

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

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

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

RAPID DATA PROTECTION FOR STORAGE DEVICES

BACKGROUND

[0001] Computers have become increasingly commonplace and used in various different settings, which has led to situations in which users desire to have data stored on computer storage devices encrypted. These situations can be a result of a user's personal desires, a corporate policy dictating that the user's computer must encrypt corporate data stored on the personal computer's storage device, and so forth. However, waiting for an entire storage device to be encrypted (which can be on the order of several minutes or hours, depending on the size of the storage device) can be frustrating for users, leading to poor user experiences.

SUMMARY

[0002] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0003] In accordance with one or more aspects, a method is implemented in a trusted runtime of a computing device that is isolated from other programs in the computing device. A request to generate a key for a portion of a storage device is received at the trusted runtime. In response to the request to generate the key, the key is generated and persisted using the trusted runtime across power cycles of the computing device. An inline cryptographic processor of the computing device is also provisioned with the key so that subsequent writes to the portion of the storage device are encrypted based on the key. Additionally, in response to a request that protection of the portion be enabled, an indication that protection of the portion is enabled is returned despite at least part of the portion being unencrypted.

[0004] In accordance with one or more aspects, a computing device includes an inline cryptographic processor and a trusted runtime that is isolated from an operating system of the computing device. The trusted runtime is configured to generate a key for a portion of a storage device, persist the key across power cycles of the computing device, and return, in response to a request that protection of the portion of the storage device be enabled, an indication that protection of the portion of the storage device is enabled despite one or more parts of the portion of the storage device being unencrypted. The inline cryptographic processor is configured to receive the key from the trusted runtime, encrypt

subsequent writes to the portion of the storage device based on the key, and decrypt subsequent reads from the portion of the storage device based on the key.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The detailed description is described with reference to the accompanying figures. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The use of the same reference numbers in different instances in the description and the figures may indicate similar or identical items. Entities represented in the figures may be indicative of one or more entities and thus reference may be made interchangeably to single or plural forms of the entities in the discussion.

[0006] Fig. 1 is a block diagram illustrating an example computing device implementing the rapid data protection for storage devices in accordance with one or more embodiments.

[0007] Fig. 2 illustrates an example system implementing the rapid data protection for storage devices in accordance with one or more embodiments.

[0008] Fig. 3 is a flowchart illustrating an example process for generating an encryption key for a newly defined portion of a storage device in accordance with one or more embodiments.

[0009] Fig. 4 is a flowchart illustrating an example process for enabling protection on a portion of a storage device in accordance with one or more embodiments.

[0010] Fig. 5 is a flowchart illustrating an example process for locking a protected portion of a storage device in accordance with one or more embodiments.

[0011] Fig. 6 is a flowchart illustrating an example process for unlocking a protected portion of a storage device in accordance with one or more embodiments.

[0012] Fig. 7 is a flowchart illustrating an example process for disabling protection of a portion of a storage device in accordance with one or more embodiments.

[0013] Fig. 8 is a flowchart illustrating an example process for revealing an encryption key for recovery or migration of a portion in accordance with one or more embodiments.

[0014] Fig. 9 illustrates an example system that is representative of one or more systems and/or devices that may implement the various techniques described herein.

DETAILED DESCRIPTION

[0015] Rapid data protection for storage devices is discussed herein. A computing device uses a data encryption and decryption system that includes a trusted runtime and an inline cryptographic processor. The trusted runtime provides a trusted or protected

execution environment, and the inline cryptographic processor provides decryption and encryption of data in-line with storage device read and write operations. The rapid data protection discussed herein works with various portions of a storage device, such as partitions of a storage device. When a portion of a storage device is created, the trusted runtime generates an encryption key for the portion and provides the encryption key to the inline cryptographic processor. The inline cryptographic processor uses the encryption key to encrypt data written to the portion and decrypt data read from the portion, but does not persist the encryption key across power cycles of the computing device. The trusted runtime persists the encryption key securely across power cycles of the computing device, allowing the operating system to request use of the encryption key to encrypt and decrypt data stored on the portion, but not revealing the encryption key to the operating system.

[0016] Access to the portion can be subsequently protected by associating the key with particular authentication credentials. The authentication credentials of a user or other entity are provided to the trusted runtime, which protects the encryption key based on the authentication credentials (e.g., encrypts the encryption key using an authentication key of the user or other entity). The trusted runtime subsequently persists the protected encryption key. Access to the encryption key can thus subsequently be obtained by the trusted runtime only in response to the proper authentication credentials being provided, thus restricting access to the data stored on the portion to a user or entity that provides the proper authentication credentials.

[0017] Thus, upon creation of the portion, data written to the portion is encrypted and decrypted using the encryption key. This encryption and decryption occurs independently of any instruction or request from a user to have data on the portion encrypted. In response to a subsequent user request to have data on the portion encrypted, access to the encryption key is gated or restricted based on the authentication credentials of the user. Further, in response to such a user request, an indication can be rapidly returned that the data on the portion is encrypted or protected; the indication can be returned without waiting for the portion to be encrypted because any data previously written to the portion is already encrypted, and data subsequently written to the portion will also be encrypted using the encryption key. However, only a user able to provide the correct authentication credentials can authorize the inline cryptographic processor to gain access to the encryption key, and thus allow the inline cryptographic processor to subsequently decrypt data stored on the portion and encrypt data for storage on the portion.

[0018] Fig. 1 is a block diagram illustrating an example computing device 100 implementing the rapid data protection for storage devices in accordance with one or more embodiments. Computing device 100 can be a variety of different types of devices, such as a physical device or a virtual device. For example, computing device 100 can be a physical device such as a desktop computer, a server computer, a laptop or netbook computer, a tablet or notepad computer, a mobile station, an entertainment appliance, a set-top box communicatively coupled to a display device, a television or other display device, a cellular or other wireless phone, a game console, an automotive computer, and so forth. Computing device 100 can also be a virtual device, such as a virtual machine running on a physical device. A virtual machine can be run on any of a variety of different types of physical devices (e.g., any of the various types listed above). Thus, computing device 100 may range from a full resource device with substantial memory and processor resources (e.g., personal computers, game consoles) to a low-resource device with limited memory and/or processing resources (e.g., traditional set-top boxes, hand-held game consoles).

[0019] Computing device 100 includes a data encryption and decryption system 102, a storage device 104, a storage device protection system 106, and an operating system 108. The data encryption and decryption system 102 manages encryption and decryption of data written to and read from the storage device 104. The storage device 104 can be implemented using a variety of different technologies, such as a flash memory device, a magnetic disk, an optical disc, combinations thereof, and so forth. Although a single storage device 104 is illustrated in Fig. 1, it should be noted that the computing device 100 can include any number of storage devices.

[0020] The data encryption and decryption system 102 includes a trusted runtime 112 and an inline cryptographic processor 114. The trusted runtime provides a trusted or protected execution environment, and provides keys to the inline cryptographic processor 114, as discussed in more detail below. The inline cryptographic processor 114 decrypts and encrypts data in-line with device read and write operations from and to the storage device 104, as discussed in more detail below.

[0021] The storage device protection system 106 manages authentication of credentials, and configuration of the trusted runtime 112 based on the authenticated credentials (e.g., provision of an authentication key to the trusted runtime 112 based on the authenticated credentials). The operating system 108 manages the operation of various hardware devices in the computing device 100, and the running of additional programs on

the computing device 100. The operating system 108, as well as other programs running on the computing device 100, can read from and write to the storage device 104 via the data encryption and decryption system 102.

[0022] The storage device 104 is illustrated as being included as part of the computing device 100. For example, the storage device 104 can be an internal storage device coupled to an internal bus of the computing device 100. By way of another example, the storage device 104 can be an internal storage device implemented as one or more chips on a same circuit board as chips implementing one or more of the systems 102, 106, and 108, can be an internal storage device implemented in the same chip as one or more of the systems 102, 106, and 108, and so forth.

[0023] The storage device 104 can alternatively be external to the computing device 100 and coupled to the computing device 100 in a variety of different wired and/or wireless manners. For example, the storage device 104 can be coupled to the computing device 100 via a Universal Serial Bus (USB) connection, a wireless USB connection, an IEEE 1394 connection, an external SATA (eSATA) connection, a Bluetooth connection, and so forth. The storage device 104 can be designed to be coupled to different computing devices (concurrently or at different times). In one or more embodiments, the storage device 104 is a removable device, such as being designed to be easily coupled to and decoupled from the computing device 100 and transported to other computing devices. An example of such a removable storage volume is a thumb drive or USB flash memory device. Alternatively, the storage device 104 can take other forms, such as being a network storage device that is coupled to computing device 100 via a network (e.g., the Internet, a local area network (LAN), a cellular or other phone network, an intranet, a storage area network (SAN), network-attached storage (NAS), a personal area network (PAN), other public and/or proprietary networks, combinations thereof, and so forth).

[0024] The storage device protection system 106 can be implemented in a variety of different manners. Although illustrated as separate from the operating system 108, the storage device protection system 106 can be implemented at least in part in the operating system 108. The storage device protection system 106 can additionally or alternatively be implemented in other manners, such as implemented as part of a pre operating system environment, as part of a pre-boot execution environment, and so forth.

[0025] Fig. 2 illustrates an example system 200 implementing the rapid data protection for storage devices in accordance with one or more embodiments. The system 200 is implemented as part of the computing device 100 of Fig. 1. The system 200 includes the

storage device protection system 106, the trusted runtime 112, the inline cryptographic processor 114, and the storage device 104. The techniques discussed herein are discussed with reference to a portion of the storage device 104. A portion of the storage device 104 refers to at least a particular logical or physical part of the storage device. A portion of the storage device 104 can refer to part of the storage device 104 referred to as a partition and that is treated (e.g., displayed in folders or directories) as if it were a separate, physical storage device. However, the techniques discussed herein can be analogously applied at other granularities and used with various other portions of storage devices. For example, rather than partitions, the techniques discussed herein can be applied analogously to portions of the storage device that are files, folders or directories, entire storage devices, collections of multiple storage devices, and so forth.

[0026] The storage device protection system 106 includes an authentication module 202 and a trusted runtime configuration module 204. The trusted runtime 112 can be configured to allow the inline cryptographic processor 114 to decrypt data on the storage device 104 only if valid authentication credentials to access the data are provided, as discussed in more detail below. The authentication module 202 authenticates the authentication credentials provided by an entity and determines whether the provided authentication credentials are valid. In one or more embodiments, the entity is a user of the system 200, although the entity can alternatively be other devices, components, modules, and so forth. The storage device protection system 106 operates as a key management system, making an authentication key corresponding to valid authentication credentials available to the trusted runtime 112.

[0027] The authentication credentials can take various different forms, such as knowledge of a secret phrase (e.g., a password), a private key corresponding to a certificate, a temporal secret (e.g., a one-time password), and so forth. The authentication module 202 can maintain or otherwise obtain a record of various data to compare against the provided authentication credentials in order to determine whether the provided authentication credentials are valid, can apply any of a variety of processes or algorithms to the provided authentication credentials to determine whether the provided authentication credentials are valid, and so forth.

[0028] In response to the authentication module 202 determining that the provided authentication credentials are valid, the storage device protection system 106 obtains an authentication key associated with the entity that provided the authentication credentials. The authentication key can be obtained in various manners, such as provided by the

storage device protection system 106, retrieved from a store maintained by the storage device protection system 106, received from the entity that provided the authentication credentials (e.g., from a user's smartcard), generated from the authentication credentials or other information provided by the entity that provided the authentication credentials, and so forth.

[0029] The trusted runtime configuration module 204 communicates with the trusted runtime 112 through a set of commands supported by the trusted runtime 112, providing various commands to the trusted runtime 112 to perform various operations. These operations can include, for example, enabling protection on a portion of the storage device 104, locking a protected portion of the storage device 104, unlocking a portion of the storage device 104, disabling protection of the storage device 104, and recovery and migration of keys. Some of these commands can include providing the authentication key obtained by the storage device protection system 106 to the trusted runtime 112, as discussed in more detail below.

[0030] The trusted runtime 112 includes a key generation module 206, a key protection module 208, an inline cryptographic processor configuration module 210, and a persistent key store 212. The trusted runtime 112 is an environment that allows execution of programs in the trusted runtime 112 in isolation from an operating system running on the same device. The trusted runtime isolates execution of programs in memory (preventing the operating system from accessing memory space used by the isolated programs when executing), in processor (preventing the operating system from accessing registers or other state of the processor when executing the isolated programs), and in storage (preventing the operating system or any external parties from accessing storage locations used by the isolated programs during operation as well as while the computing device is turned off). Thus, the trusted runtime 112 allows modules to execute as programs on a computing device that are isolated from and independent of the operating system and any other programs (e.g., not executing in the trusted runtime 112) executing on the computing device. The operating system or other programs executing on the computing device can only access programs in the trusted runtime using an interface (e.g., a set of interface commands) exposed by the trusted runtime 112.

[0031] The trusted runtime 112 can be implemented in a variety of different manners. In one or more embodiments, the trusted runtime 112 is implemented as an ARM TrustZone on ARM architecture processors of the computing device. Alternatively, the trusted runtime 112 can be implemented in other manners, such as by a separate processor

of the computing device (e.g., a processor or processor core separate from the processor or processor core on which the operating system executes). Alternatively, the trusted runtime can be implemented by an additional secure mode provided or facilitated by a hypervisor or other virtual machine manager of the computing device.

[0032] The key generation module 206 generates encryption keys to be used to encrypt data on portions of the storage device 104. The key generation module 206 can generate encryption keys using any of a variety of public and/or proprietary techniques. In one or more embodiments, the key generation module 206 generates a different encryption key for each portion of the storage device 104, and can maintain these keys in the persistent key store 212. The keys maintained in the persistent key store 212 are maintained as associated with the respective portions of the storage device 104 for which they were generated. One or more (e.g., multiple (x)) keys can be maintained in the persistent key store 212, such as one key for each portion of the storage device 104. The persistent key store 212 can be implemented in a variety of different manners, and allows the keys in the store 212 to be persisted across power cycles of the computing device. A power cycle of the computing device refers to the computing device being powered off and then powered on again, or the computing device being rebooted or hardware reset. Thus, after the computing device implementing the system 200 has been powered off and again powered on, or after the computing device has been rebooted, the keys remain in the persistent key store 212.

[0033] The key protection module 208 protects (e.g., encrypts) the keys in the persistent key store 212 under certain circumstances, such as when protection on a portion of the storage device 104 is enabled as discussed in more detail below. The inline cryptographic processor configuration module 210 communicates with the inline cryptographic processor 114, providing encryption keys to the inline cryptographic processor 114 as discussed in more detail below.

[0034] In one or more embodiments, an operating system (e.g., operating system 108 of Fig. 1) can communicate with the trusted runtime 112 through a set of commands supported by the trusted runtime 112. These commands can allow certain operations to be performed, such as locking a protected portion of the storage device. In one or more embodiments, commands that include an authentication key being provided to the trusted runtime 112 are provided by the storage device protection system 106, whereas other commands (e.g., locking a protected portion of the storage device) that need not have an authentication key provided to the trusted runtime can be provided by the operating system

108 or the storage device protection system 106. It should be noted that, regardless of whether a command is received from the operating system 108 or from the storage device protection system 106, the trusted runtime 112 does not disclose the encryption keys generated by the key generation module 206 to the operating system 108 or the storage device protection system 106.

[0035] Alternatively, rather than maintaining protected keys in the persistent key store 212, the key protection module 208 can return the protected encryption key to the operating system. The operating system can optionally store the protected encryption key so that the protected encryption key is persisted across power cycles of the computing device. The key protection module 208 knows the key used to protect (e.g., encrypt) the encryption key and does not reveal the key to the operating system. The operating system therefore does not have access to the encryption key, but the operating system can provide the protected encryption key to the key protection module 208, which can in turn retrieve the encryption key from the protected key. Thus, the trusted runtime 112 can leverage the operating system to store one or more protected keys rather than (or alternatively in addition to) maintaining the protected keys in the persistent key store 212.

[0036] The inline cryptographic processor 114 includes an encryption and decryption module 214, a management module 216, and an encryption key store 218. Encryption keys are received from the trusted runtime 112 and stored as encryption keys in the encryption key store 218. One or more (e.g., multiple (y)) encryption keys can be stored in the encryption key store 218. In one or more embodiments, the encryption key store 218 includes one encryption key for each portion of the storage device 104. Once stored in the encryption key store 218, an encryption key is maintained in the encryption key store 218 until a command is received (e.g., from the trusted runtime 112) to delete the encryption key, or until a power cycle occurs (e.g., the computing device is powered down, the computing device is rebooted, and so forth). Although the encryption key is not persisted in the encryption key store 218, after a power cycle occurs the trusted runtime 112 can again provide encryption keys to the inline cryptographic processor for inclusion in the encryption key store 218 (optionally only after being obtained from a protected encryption key in the persistent key store 212, as discussed in more detail below).

[0037] The inline cryptographic processor 114 receives and responds to I/O requests 220. The I/O requests 220 are from an operating system (e.g., the operating system 108 of Fig. 1), or alternatively other programs running on the computing device implementing the system 200. The encryption and decryption module 214 encrypts data received as part of

an I/O request 220 prior to the data being written to a portion of the storage device 104. The encryption and decryption module 214 also decrypts data read from a portion of the storage device 104 prior to the data being returned to the requester as part of a read I/O request 220.

[0038] In embodiments in which the encryption key store 218 stores multiple encryption keys, an indication of the encryption key to be used by the inline cryptographic processor 114 to encrypt or decrypt data for an I/O request 220 can be provided as part of the I/O request 220. The indication can take various different forms, such as an identifier of an encryption key (e.g., provided by the trusted runtime when the portion is created or a portion is unlocked, as discussed in more detail below), an indication of the portion to which the request applies, and so forth. In one or more embodiments, this identifier of an encryption key is an identifier of a register of the inline cryptographic processor in which the encryption key is stored.

[0039] Alternatively, no such indication of the encryption key to be used by the inline cryptographic processor 114 need be provided. For example, the inline cryptographic processor 114 or the trusted runtime 112 can maintain a record of which encryption key is used for which portion of the storage device 104, in which case no such indication need be provided. By way of another example, if the encryption key store 218 includes a single encryption key, then no such indication need be provided.

[0040] It should be noted that although an indication of an encryption key to use to encrypt and decrypt content can be received by the inline cryptographic processor 114, and the encryption key is provided to the inline cryptographic processor 114 by the trusted runtime 112, the inline cryptographic processor 114 does not disclose the encryption keys to devices or programs external to the inline cryptographic processor 114. Under particular conditions, such as when swapping encryption keys in and out of the encryption key store 218 as discussed in more detail below, a protected version of an encryption key may be disclosed by the inline cryptographic processor 114, but other than under such particular conditions the encryption keys (protected or unprotected) are not disclosed by the inline cryptographic processor 114.

[0041] The management module 216 manages operation of the inline cryptographic processor 114, optionally including managing communications with the trusted runtime 112. In one or more embodiments, the trusted runtime 112 communicates with the management module 216 of the inline cryptographic processor 114 via another module or component, using secure (e.g., encrypted) communication channels. For example, the

trusted runtime 112 can provide encryption keys to the management module 216, which stores the encryption keys in the encryption key store 218. In other embodiments, the trusted runtime 112 can communicate with the inline cryptographic processor 114 directly, such as by storing encryption keys directly in registers of the encryption key store 218. The management module 216 also manages disclosing a protected version of an encryption key under the particular conditions discussed above.

[0042] The inline cryptographic processor 114 is implemented at least in part in hardware (e.g., as one or more hardware chips), and is in-line with the regular read and write operations for the storage device 104. Thus, all data written to a portion of the storage device 104 can be encrypted by the inline cryptographic processor 114, and all data read from a portion of the storage device can be decrypted by the inline cryptographic processor 114. Generally, the inline cryptographic processor 114 is cryptographic hardware with one or more registers that can store encryption keys, and that allows an operating system or other programs to reference and use the registers (e.g., to have data encrypted), but does not allow an operating system or other programs to extract the contents of the registers.

[0043] In one or more embodiments, the inline cryptographic processor 114 is implemented as part of a storage controller for the storage device 104. Alternatively, the inline cryptographic processor 114 can be implemented in other manners, such as in a video path (e.g., a video card or other hardware) for video recording or playback, an audio path (e.g., an audio card or other hardware) for audio recording or playback, in a communications path (e.g., a network card or other hardware) for sending or receiving data to another device, and so forth.

[0044] The encryption and decryption module 214 can encrypt and decrypt data based on an encryption key in the encryption key store 218 using any of a variety of public and/or proprietary techniques. In one or more embodiments, the encryption and decryption module 214 uses symmetric key cryptography, with the same key (the encryption key) being used for encryption and decryption. Alternatively, the encryption and decryption module 214 can use asymmetric key cryptography, with different keys being used for encryption and decryption. In such embodiments, although only the encryption key is illustrated in the encryption key store 218, the key generation module 206 generates both the encryption key and the decryption key, and the encryption key store 218 maintains both the encryption key and the decryption key. The encryption and decryption module 214 thus uses the encryption key from the store 218 when encrypting data to be written to

a portion of the storage device 104, and uses the decryption key from the store 218 when decrypting data read from a portion of the storage device 104. Thus, it should be noted that in discussions herein referring to decryption using the encryption key, such references are to a separate decryption key in situations where asymmetric key cryptography is being used.

[0045] In one or more embodiments, the encryption key store 218 can store up to a particular number (*y*) of encryption keys, although the storage device 104 may include more than that particular number of portions. In such situations, encryption keys are swapped in and out of the encryption key store 218 as appropriate so that the encryption key identified by an I/O request 220 is included in the encryption key store 218. The swapping of encryption keys can be implemented in various manners. For example, if an encryption key identified by an I/O request 220 is not included in the encryption key store 218, the management module 216 can request and receive the identified encryption key from the trusted runtime 112.

[0046] By way of another example, the management module 216 can have an encryption key protected (e.g., encrypted using a secret key known only to the inline cryptographic processor 114) and the protected encryption key returned to the operating system. If an encryption key identified by an I/O request 220 is not included in the encryption key store 218, the management module 216 can request and receive a protected version of the identified encryption key from the operating system, then decrypt and store the encryption key in the encryption key store 218. Thus, the operating system itself can participate in the swapping of encryption keys. In such situations, the management module 216 can optionally change the secret key used to protect the encryption key after a particular amount of time elapses or in response to a particular event (e.g., the computing device is powered down or rebooted). Thus, after the particular amount of time elapses or the particular event occurs, the protected key maintained by the operating system is no longer usable by the inline cryptographic processor 114 because the key has changed.

[0047] The use of the trusted runtime 112 and the inline cryptographic processor 114 allows the system 200 to generate a different key for each portion created on the storage device 104, and encrypt/decrypt all data stored in the portion as the data is written/read. While portions can be created very early in the lifetime of the storage device 104, and the storage device 104 can be handled by potentially untrusted parties, the portion encryption key is never exposed to such parties even if they have physical access to the storage device

104 and/or have administrative privileges on the operating system installed on the computing device implementing the system 200.

[0048] The system 200 supports various different operations, as mentioned above. These operations include, for example, generating an encryption key for a newly defined portion of the storage device 104, enabling protection on a portion of the storage device 104, locking a protected portion of the storage device 104, unlocking a portion of the storage device 104, disabling protection of the storage device 104, and recovery and migration of keys. These operations are discussed in additional detail as follows.

Generating Encryption Keys for Newly Created Portions

[0049] Portion creation refers to the creation or definition of a new portion of the storage device 104. One or more portions can be created on the storage device 104. These portions can be created in response to a request from a user of the computing device, an administrator of the computing device, another device or module, and so forth. The creation of a portion includes various acts, including establishing and maintaining various data structures describing the portion, such as which parts of the storage device (e.g., which sectors of a disk) are included in the portion. The portion can be created using any of a variety of public and/or proprietary techniques. Additionally, as part of the creation of the portion, the trusted runtime is requested to generate an encryption for the newly created (also referred to as newly defined) portion.

[0050] Fig. 3 is a flowchart illustrating an example process 300 for generating an encryption key for a newly defined portion of a storage device in accordance with one or more embodiments. Process 300 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 300 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 300 is an example process for generating an encryption key for a newly defined portion of a storage device; additional discussions of generating an encryption key for a newly defined portion of a storage device are included herein with reference to different figures.

[0051] In process 300, a request to generate an encryption key for a newly defined portion of a storage device is received (act 302). The request can be received in the form of a command, and can be received from various different modules. For example, the request can be received from the operating system 108 of Fig. 1, the storage device protection system 106 of Fig. 1 or Fig. 2, and so forth. The request can be received, for example, from a program or module creating or defining the newly defined portion of the

storage device. This portion can be created or defined in any of a variety of manners using various public and/or proprietary techniques.

[0052] In response to the received request, a new encryption key is generated (act 304). The new encryption key can be generated in any of a variety of manners, as discussed above.

[0053] Additionally, the newly generated encryption key is persisted using the trusted runtime (act 306). The newly generated encryption key can be persisted in different manners. In one or more embodiments, the newly generated encryption key is maintained in a persistent key store of the trusted runtime. Alternatively, the encryption key can be persisted in other manners. For example, an encrypted encryption key can be generated (e.g., generated by encrypting the encryption key using a secret key known only to the trusted runtime) and the encrypted encryption key provided to the operating system. The operating system can thus maintain the encrypted encryption key, relieving the trusted runtime of the need to maintain the encryption key.

[0054] Regardless of how the newly generated encryption key is persisted, the encryption key is persisted across power cycles. For example, the persistent key store of the trusted runtime can persist the encryption key across power cycles as discussed above. By way of another example, the operating system can manage storage of the encrypted encryption key in a nonvolatile memory that persists the encrypted encryption key across power cycles.

[0055] The newly generated encryption key is also provided to the inline cryptographic processor (act 308). Providing the newly generated encryption key to the inline cryptographic processor provisions the inline cryptographic processor with the newly generated encryption key, allowing the inline cryptographic processor to encrypt data written to the portion and decrypt data read from the portion based on the newly generated encryption key. The manner in which the encryption key is used to encrypt and decrypt data varies based on the manner in which encryption and decryption is performed by the inline cryptographic processor.

[0056] An identifier of the newly generated encryption key is returned to the requester (act 310). The identifier can take any of a variety of different forms, but allows different encryption keys (e.g., used for different portions) to be identified. Thus, if the inline cryptographic processor is storing multiple encryption keys, one for each of multiple portions of a storage device, the correct encryption key for a particular portion can be identified. For example, the operating system can issue read and write requests that

include an identifier of the particular encryption key to use. It should be noted that this identifier of an encryption key merely allows the operating system to identify to the inline cryptographic processor which encryption key to use; the identifier does not provide or otherwise allow the operating system to obtain the encryption key.

[0057] Thus, all data written to the portion is encrypted by the inline cryptographic processor, and all data read from the portion is decrypted by the inline cryptographic processor. No component, module, or device other than the trusted runtime and the inline cryptographic processor have or can gain access to the newly generated encryption key, and neither the trusted runtime nor the inline cryptographic processor will reveal the encryption key to any component, module, or device external to the trusted runtime and the inline cryptographic processor as discussed above.

[0058] It should be noted that after creation of the portion, access to the data stored in the portion is unrestricted. Although the data stored in the portion is encrypted, no authentication credentials need be provided to the trusted runtime in order to read the encrypted data.

[0059] The process 300 can be repeated any number of times to create new portions on the storage device.

Enabling Protection on a Portion

[0060] Enabling protection on a portion refers to associating the portion with particular authentication credentials, allowing only authenticated entities to access the portion. For example, a user can request to have the portion protected, and as a result have the portion associated with the authentication credentials of the user, allowing access to the data stored on the portion (and allowing new data to be stored on the portion) only if the authentication credentials of the user are provided.

[0061] Fig. 4 is a flowchart illustrating an example process 400 for enabling protection on a portion of a storage device in accordance with one or more embodiments. Process 400 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 400 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 400 is an example process for enabling protection on a portion of a storage device; additional discussions of enabling protection on a portion of a storage device are included herein with reference to different figures.

[0062] In process 400, a request to enable protection on a portion of a storage device is received (act 402). This request is also referred to as a request to encrypt the portion of the storage device. The request can be received in the form of a command, and can be received from various different modules. In one or more embodiments, the request is received from the storage device protection system 106 of Fig. 1 or Fig. 2, although the request can alternatively be received from other modules such as the operating system 108 of Fig. 1.

[0063] In response to the request, the encryption key for the portion is protected based on an authentication key (act 404). The authentication key is the authentication key associated with user credentials provided to the storage device protection system 106. In one or more embodiments, the authentication key is the authentication key associated with an entity (e.g., a user) currently logged into a computing device implementing the system 200 of Fig. 2 at the time the request is received in act 402. Alternatively, the authentication key can be the authentication key associated with another entity providing user credentials to the computing device implementing the system 200 of Fig. 2.

[0064] The encryption key can be protected based on the authentication key (also referred to as wrapping the encryption key with the authentication key) in various manners. In one or more embodiments, an encryption algorithm uses the authentication key as the key to encrypt the encryption key. Any of a variety of public and/or proprietary encryption algorithms can be used to encrypt the encryption key. Once encrypted, the encryption key can be retrieved (decrypted) using the authentication key and an appropriate decryption algorithm, but cannot be retrieved (or it is computationally very difficult to be retrieved) without the authentication key.

[0065] The authentication key to use to protect the encryption key can be obtained by the trusted runtime in a variety of different manners. In one or more embodiments, the authentication key is provided by the storage device protection system 106 as part of the request in act 402. Alternatively, the authentication key can be provided by the storage device protection system 106, or alternatively another module or device, in other manners or at other times.

[0066] The protected encryption key for the portion is maintained (act 406). The protected encryption key for the portion can be maintained in different manners. In one or more embodiments, the protected encryption key is maintained in the persistent key store of the trusted runtime. The encryption key for the portion in the persistent key store of the trusted runtime is replaced with the protected encryption key for the portion, so the trusted

runtime no longer maintains the encryption key in the clear – the unprotected encryption key is removed and only the protected encryption key is maintained. Thus, without being provided the authentication key, the trusted runtime can no longer determine the encryption key for the portion of the storage device. Alternatively, the protected encryption key can be maintained in other manners. For example, the protected encryption key can be provided to the operating system, and the encryption key removed (e.g., deleted) from the persistent key store. Thus, without being provided the authentication key, the trusted runtime can no longer determine the encryption key for the portion of the storage device.

[0067] The trusted runtime also has the protected encryption key persisted across power cycles of the computing device. The trusted runtime can have the protected encryption key persisted across power cycles of the computing device in different manners, such as by storing the protected encryption key in the persistent key store of the trusted runtime, or by providing the protected encryption key to the operating system.

[0068] An indication that the portion is protected is also returned to the requester (act 408). This indication can also be referred to as an indication that the portion is encrypted. The indication can be returned rapidly to the requester – the trusted runtime need not wait for data already written to the portion to be encrypted before returning the indication because all data written to the portion is encrypted by the inline cryptographic processor. Although parts of the portion may not yet be encrypted, those unencrypted parts are parts to which no data has been written yet (if data had been written to those parts, it would have been encrypted by the inline cryptographic processor).

[0069] After protection is enabled on the portion, the authentication key is needed in order for the trusted runtime to derive the encryption key and provide the encryption key to the inline cryptographic processor. The encryption key is no longer available to the trusted runtime other than via the protected encryption key (e.g., the encryption key has been replaced in the trusted runtime with the protected encryption key), so the trusted runtime cannot retrieve the encryption key without the authentication key.

Locking a Protected Portion

[0070] Locking a protected portion refers to preventing access to a portion for which protection was previously enabled. For example, at some point in time after a user requests to enable protection of a portion, the user can request to have the portion locked. As a result, access to the portion is prevented, so that data stored on the portion cannot be decrypted, and new encrypted data cannot be written to the portion.

[0071] Fig. 5 is a flowchart illustrating an example process 500 for locking a protected portion of a storage device in accordance with one or more embodiments. Process 500 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 500 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 500 is an example process for locking a protected portion of a storage device; additional discussions of locking a protected portion of a storage device are included herein with reference to different figures.

[0072] In process 500, a request to lock a protected portion of a storage device is received (act 502). The request can be received in the form of a command, and can be received from various different modules. In one or more embodiments, the request is received from the storage device protection system 106 of Fig. 1 or Fig. 2, although the request can alternatively be received from other modules such as the operating system 108 of Fig. 1. The request can be initiated by a user or other module, and can be an explicit request (e.g., a request that the protected portion be locked because the user desires it) or an implicit request (e.g., a request that the protected portion be locked because the user is logging off of his or her account on the computing device).

[0073] In one or more embodiments, storage device protection system 106 or the operating system 108 issues the request in act 502 as part of a process of preventing access to the portion. The storage device protection system 106 or the operating system 108 can also perform additional acts in preventing access to the portion, including stopping sending I/O requests to the portion, dismounting the portion, and so forth.

[0074] In response to the request, the encryption key is removed from the inline cryptographic processor (act 504). The encryption key can be removed from the inline cryptographic processor by the trusted runtime in various manners, such as issuing a command to the inline cryptographic processor to delete the encryption key from the encrypted key store of the inline cryptographic processor, the trusted runtime writing a new value (e.g., all zeroes, all ones, etc.) to the register of the inline cryptographic processor where the encryption key is stored, and so forth.

[0075] Additionally, any unprotected copies of the encryption key are removed from the trusted runtime (act 506). Situations can arise in which the trusted runtime, after providing an encryption key to the inline cryptographic processor, keeps a copy of the encryption key for some amount of time (e.g., for a few minutes, or longer to facilitate

swapping of encryption keys in the inline cryptographic processor). If any unprotected copies of the encryption key remain in the trusted runtime, those copies are deleted in act 506. It should be noted, however, that if the protected encryption key is stored in the persistent key store, the protected encryption key in the persistent key store is not deleted.

[0076] After a protected portion is locked, neither the trusted runtime nor the inline cryptographic processor has an unprotected copy of the encryption key. The trusted runtime may have (or obtain) a protected copy of the encryption key, and the authentication key is needed in order for the trusted runtime to derive the encryption key and again provide the encryption key to the inline cryptographic processor.

Unlocking a Portion

[0077] Unlocking a protected portion refers to allowing access to a previously locked portion. For example, at some point in time after a portion is locked, the user can request to have the portion unlocked. As a result, access to the portion is again allowed so that data stored on the portion can be decrypted, and data can be encrypted and written to the portion.

[0078] Fig. 6 is a flowchart illustrating an example process 600 for unlocking a protected portion of a storage device in accordance with one or more embodiments. Process 600 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 600 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 600 is an example process for unlocking a protected portion of a storage device; additional discussions of unlocking a protected portion of a storage device are included herein with reference to different figures.

[0079] In process 600, a request to unlock a portion of a storage device is received (act 602). The request can be received in the form of a command, and can be received from various different modules. In one or more embodiments, the request is received from the storage device protection system 106 of Fig. 1 or Fig. 2, although the request can alternatively be received from other modules such as the operating system 108 of Fig. 1. The request can be initiated by a user or other module, and can be an explicit request (e.g., a request that the portion be unlocked because the user desires it) or an implicit request (e.g., a request that the portion be unlocked because the user is logging into his or her account on the computing device).

[0080] In response to the request, the encrypted key for the portion is obtained from the protected encryption key for the portion (act 604). The encryption key for the portion is obtained based on both the protected encryption key and an authentication key. The protected encryption key for the portion is maintained in the persistent key store, or can be received from the operating system, as discussed above. The authentication key is the authentication key associated with user credentials provided to the storage device protection system 106. In one or more embodiments, the authentication key is the authentication key associated with an entity (e.g., a user) currently logged into a computing device implementing the system 200 of Fig. 2 at the time the request is received in act 402. Alternatively, the authentication key can be the authentication key associated with another entity providing user credentials to the computing device implementing the system 200 of Fig. 2.

[0081] The encryption key can be obtained from the protected encryption key in a variety of different manners, based at least in part on the manner in which the encryption key was previously protected. In one or more embodiments, a decryption algorithm uses the authentication key as the key to decrypt the encryption key. Any of a variety of public and/or proprietary encryption algorithms can be used to decrypt the encryption key.

[0082] The authentication key to use to obtain the encryption key can be obtained by the trusted runtime in a variety of different manners. In one or more embodiments, the authentication key is provided by the storage device protection system 106 as part of the request in act 602. Alternatively, the authentication key can be provided by the storage device protection system 106, or alternatively another module or device, in other manners or at other times.

[0083] The obtained encryption key is provided to the inline cryptographic processor (act 606). Providing the obtained encryption key to the inline cryptographic processor provisions the inline cryptographic processor with the obtained encryption key, allowing the inline cryptographic processor to encrypt data written to the portion and decrypt data read from the portion based on the obtained encryption key. The manner in which the encryption key is used to encrypt and decrypt data varies based on the manner in which encryption and decryption is performed by the inline cryptographic processor.

[0084] An identifier of the obtained encryption key is returned to the requester (act 608). The identifier can take any of a variety of different forms, but allows different encryption keys (e.g., used for different portions) to be identified, as discussed above. It should be noted that this identifier of the encryption key merely allows the operating

system to identify to the inline cryptographic processor which encryption key to use; the identifier does not provide or otherwise allow the operating system to obtain the encryption key.

[0085] After the portion is unlocked, the inline cryptographic processor is provisioned with the encryption key for the portion. Data can be encrypted and written to the portion, as well as read from the portion and decrypted. Although the inline cryptographic processor has an unprotected copy of the encryption key, no other device, module, or component external to the inline cryptographic processor and the trusted runtime have access to the encryption key. In one or more embodiments, the trusted runtime discards all unprotected copies of the encryption key once it has been provisioned in the inline cryptographic processor. Alternatively, the trusted runtime may maintain a copy of the encryption key for a particular amount of time.

Disabling Protection of a Portion

[0086] Disabling protection of a portion refers to disassociating the portion with particular authentication credentials, no longer restricting access to the portion to only authenticated entities. For example, a user can request to have protection of the portion disabled, and as a result have the portion disassociated with the authentication credentials of the user, allowing access to the data stored on the portion (and allowing new data to be stored on the portion) without regard for what (if any) authentication credentials of the user are provided.

[0087] Fig. 7 is a flowchart illustrating an example process 700 for disabling protection of a portion of a storage device in accordance with one or more embodiments. Process 700 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 700 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 700 is an example process for disabling protection of a portion of a storage device; additional discussions of disabling protection of a portion of a storage device are included herein with reference to different figures.

[0088] In process 700, a request to disable protection of a portion of a storage device is received (act 702). The request can be received in the form of a command, and can be received from various different modules. In one or more embodiments, the request is received from the storage device protection system 106 of Fig. 1 or Fig. 2, although the

request can alternatively be received from other modules such as the operating system 108 of Fig. 1.

[0089] In response to the request, the encryption key for the portion is obtained from the protected encryption key for the portion (act 704). The encryption key for the portion is obtained based on both the protected encryption key and an authentication key. The protected encryption key for the portion is maintained in the persistent key store, or can be received from the operating system, as discussed above. The authentication key is the authentication key associated with user credentials provided to the storage device protection system 106. In one or more embodiments, the authentication key is the authentication key associated with an entity (e.g., a user) currently logged into a computing device implementing the system 200 of Fig. 2 at the time the request is received in act 402. Alternatively, the authentication key can be the authentication key associated with another entity providing user credentials to the computing device implementing the system 200 of Fig. 2.

[0090] The encryption key can be obtained from the protected encryption key in a variety of different manners, based at least in part on the manner in which the encryption key was previously protected. In one or more embodiments, a decryption algorithm uses the authentication key as the key to decrypt the encryption key. Any of a variety of public and/or proprietary encryption algorithms can be used to decrypt the encryption key.

[0091] The authentication key to use to obtain the encryption key can be obtained by the trusted runtime in a variety of different manners. In one or more embodiments, the authentication key is provided by the storage device protection system 106 as part of the request in act 702. Alternatively, the authentication key can be provided by the storage device protection system 106, or alternatively another module or device, in other manners or at other times.

[0092] Alternatively, in one or more embodiments the trusted runtime may choose to have a copy of the unprotected key in its memory while the portion is unlocked. In such embodiments, the trusted runtime need not decrypt the protected encryption key in order to obtain the unprotected encryption key in act 704.

[0093] The encryption key obtained in act 704 is persisted using the trusted runtime (act 706). The encryption key can be persisted in different manners, analogous to the discussion above regarding act 306 of Fig. 3. For example, the encryption key can be maintained in a persistent key store of the trusted runtime, or an encrypted encryption key can be generated (e.g., generated by encrypting the encryption key using a secret key

known only to the trusted runtime) and the encrypted encryption key provided to the operating system. The encryption key is persisted across power cycles, as discussed above.

[0094] Additionally, the protected encryption key is removed (act 708). In situations in which the protected encryption key is maintained in the persistent key store, the obtained encryption key effectively replaces the protected encryption key in the persistent key store, for example overwriting the protected encryption key. Thus, the protected encryption key is removed from the persistent key store. Alternatively, in situations in which the protected encryption key is returned to and maintained by the operating system, the trusted runtime notifies the operating system to delete the protected encryption key. Thus, the protected encryption key is removed from the operating system.

[0095] After protection of the portion is disabled, access to the data stored in the portion is unrestricted. Although the data stored in the portion is encrypted, no authentication credentials need be provided to the trusted runtime in order to read the encrypted data. In one or more embodiments, the data in the portion is deleted or protected in some other manner prior to disabling protection of the portion.

[0096] As discussed above, even when protection of the portion is enabled and the portion is unlocked, the encryption key is not known to the operating system or the storage device protection system 106. Protection of the portion can thus be disabled, and the computing device transferred to a different user without a risk that the encryption keys in the trusted runtime or inline cryptographic processor are known to a previous user of the computing device or the storage device protection system used by the previous user of the computing device.

Portion Recovery and Migration

[0097] Maintaining the encryption key or the protected encryption key in the trusted runtime makes the encryption key, and thus the portion, inaccessible if the storage device is moved to a different computing device. Portion recovery and migration refers to allowing the trusted runtime to reveal a protected encryption key, or the unprotected encryption key, in order to allow recovery or migration of the portion. For example, a user can request to have the storage device including the portion migrated to a different computing device, and have the protected encryption key revealed in order to do so. By way of another example, a user can request to have the encryption key revealed to an administrator or other device for recovery purposes, in case the user loses his or her authentication credentials.

[0098] Fig. 8 is a flowchart illustrating an example process 800 for revealing an encryption key for recovery or migration of a portion in accordance with one or more embodiments. Process 800 is carried out by a trusted runtime of a computing device, such as trusted runtime 112 of Figs. 1 and 2, and can be implemented in software, firmware, hardware, or combinations thereof. Process 800 is shown as a set of acts and is not limited to the order shown for performing the operations of the various acts. Process 800 is an example process for revealing an encryption key for recovery or migration of a portion; additional discussions of revealing an encryption key for recovery or migration of a portion are included herein with reference to different figures.

[0099] In process 800, a request for migration or recovery of an encryption key of a portion of a storage device is received (act 802). The request can be received in the form of a command, and can be received from various different modules. In one or more embodiments, the request is received from the storage device protection system 106 of Fig. 1 or Fig. 2, although the request can alternatively be received from other modules such as the operating system 108 of Fig. 1.

[00100] In response to the request, the encryption key or protected encryption key is revealed as requested (act 804). The trusted runtime can optionally take various authentication or approval actions to verify that it should indeed reveal the requested key. The unprotected copy of the key can be revealed, or alternatively a protected copy of the key can be revealed, depending on the request. Alternatively, a copy of the key protected using (e.g., encrypted based on) a secret key of the trusted runtime can be revealed. Other devices or components can store such a protected key, but only the trusted runtime has the secret key that can be used to obtain the encryption key from the protected encryption key. Alternatively, the trusted runtime can choose to reveal the secret key to another user or device (e.g., an administrator) if desired.

[00101] In one or more embodiments, the request 802 is received from a backup system, such as an administrator or other device on a network. The backup system provides an indication of a key (e.g., provides the key) that is to be used to encrypt the encryption key, and the encrypted encryption key is provided to the backup system. The user and operating system of the computing device need not have access to the key used to encrypt the encryption key.

[00102] Additionally, an indication that the encryption key has been revealed is persisted in the trusted runtime (act 806). The indication is persisted, for example, in the persistent key store along with the revealed key. The indication can optionally indicate

how the key was revealed (e.g., whether the unprotected encryption key was revealed or the protected encryption key was revealed).

[00103] The operating system or storage device protection system 106 can retrieve this indication in act 806, and act as desired based on this indication. For example, a storage device protection system 106 may display a warning to a user of the computing device that the encryption key for the portion has been revealed, and receive the user's approval to proceed with using the portion before providing an authentication key to the trusted runtime.

[00104] Returning to Fig. 2, various operations regarding portions are discussed herein. It should be noted that these operations are examples, and that the system 200 may not support all of these operations. For example, the system 200 may not support portion recovery and migration. Additionally, it should be noted that the system 200 can support additional operations regarding portions. For example, the system 200 may support a key replacement operation that causes the trusted runtime to generate a new encryption key for a portion and replace the previous encryption key in the persistent key store with the newly generated encryption key, or replace the previous protected encryption key with a protected newly generated key. Although such key replacement would prevent any data previously written to the portion from being subsequently decrypted (because the encryption key used to decrypt such data has been deleted), such key replacement may allow continued use of the portion and peace of mind for the user if the previous encryption key had been revealed by the trusted runtime (e.g., due to a portion recovery and migration operation). Alternatively, such key replacement could allow for re-encryption of the portion with the newly generated key, preserving the data previously written to the portion (and data previously written to the portion to be decrypted if a backup system made the encryption key used to decrypt such previously written data available, or the encryption key was otherwise available).

[00105] It should also be noted that, although portions of the storage device 104 are discussed as having all the data written to the portion encrypted and read from the portion decrypted, in some situations such encryption and decryption need not be performed. In one or more embodiments, at least part of a portion can optionally include data that is not encrypted. Such parts may include metadata regarding the portion, such as an identifier of key protectors stored on the portion, how to access the portion, and so forth. Such data need not be encrypted or decrypted. Such data can be identified in different manners, such as the operating system providing an indication that no encryption key is to be used (e.g.,

providing an identifier associated with no encryption, such as a null identifier, or providing no identifier at all).

[00106] Additionally, situations can arise in which the operating system desires to read and write the data stored on a portion in its encrypted form. For example, an operating system may transfer encrypted data from one storage device to another, backup encrypted data, and so forth. The operating system can submit an I/O request indicating that no encryption key is to be used, so the inline cryptographic processor 114 provides the requested encrypted content to the operating system.

[00107] Thus, it should also be noted that some I/O requests to a portion can be supported even though the portion may be locked. For example, unencrypted data may still be read from the portion, or the encrypted data itself returned to the operating system in response to an I/O request, as no encryption key is used to decrypt such data.

[00108] The techniques discussed herein support various different usage scenarios. Data is encrypted on a portion of a storage device based on an encryption key generated by the trusted runtime and provided to the inline cryptographic processor. The encryption key can thus be generated in response to a user request at the computing device, and the user need not rely on a third party (e.g., a manufacturer of the computing device) provisioning the computing device with the encryption key. Furthermore, neither the trusted runtime nor the inline cryptographic processor need disclose the encryption key to other programs external to the trusted runtime and the inline cryptographic processor – no third party or other device need be provided with the encryption key.

[00109] Additionally, in response to a user request to have the portion encrypted or protected, an indication can be rapidly (immediately) returned that the data on the portion is encrypted or protected. The indication can be returned without waiting for data on the portion to be encrypted because any data written to the portion since its creation is already encrypted. Some parts of the portion may not be encrypted, but those parts are the parts of the portion to which no data has been written since the portion was created. The user thus need not wait an extended period of time (e.g., several minutes to many hours) for the data on the portion to be encrypted. Rather, the indication can be returned rapidly (e.g., within a threshold amount of time, such as 1-5 seconds), and can optionally be returned synchronously as part of a call made to have the portion encrypted or protected.

[00110] Although particular functionality is discussed herein with reference to particular modules, it should be noted that the functionality of individual modules discussed herein can be separated into multiple modules, and/or at least some functionality

of multiple modules can be combined into a single module. Additionally, a particular module discussed herein as performing an action includes that particular module itself performing the action, or alternatively that particular module invoking or otherwise accessing another component or module that performs the action (or performs the action in conjunction with that particular module). Thus, a particular module performing an action includes that particular module itself performing the action and/or another module invoked or otherwise accessed by that particular module performing the action.

[00111] Fig. 9 illustrates an example system generally at 900 that includes an example computing device 902 that is representative of one or more systems and/or devices that may implement the various techniques described herein. The computing device 902 may be, for example, a server of a service provider, a device associated with a client (e.g., a client device), an on-chip system, and/or any other suitable computing device or computing system.

[00112] The example computing device 902 as illustrated includes a processing system 904, one or more computer-readable media 906, and one or more I/O Interfaces 908 that are communicatively coupled, one to another. Although not shown, the computing device 902 may further include a system bus or other data and command transfer system that couples the various components, one to another. A system bus can include any one or combination of different bus structures, such as a memory bus or memory controller, a peripheral bus, a universal serial bus, and/or a processor or local bus that utilizes any of a variety of bus architectures. A variety of other examples are also contemplated, such as control and data lines.

[00113] The processing system 904 is representative of functionality to perform one or more operations using hardware. Accordingly, the processing system 904 is illustrated as including hardware elements 910 that may be configured as processors, functional blocks, and so forth. This may include implementation in hardware as an application specific integrated circuit or other logic device formed using one or more semiconductors. The hardware elements 910 are not limited by the materials from which they are formed or the processing mechanisms employed therein. For example, processors may be comprised of semiconductor(s) and/or transistors (e.g., electronic integrated circuits (ICs)). In such a context, processor-executable instructions may be electronically-executable instructions.

[00114] The computer-readable media 906 is illustrated as including memory/storage 912. The memory/storage 912 represents memory/storage capacity associated with one or more computer-readable media. The memory/storage 912 may include volatile media

(such as random access memory (RAM)) and/or nonvolatile media (such as read only memory (ROM), Flash memory, optical disks, magnetic disks, and so forth). The memory/storage 912 may include fixed media (e.g., RAM, ROM, a fixed hard drive, and so on) as well as removable media (e.g., Flash memory, a removable hard drive, an optical disc, and so forth). The computer-readable media 906 may be configured in a variety of other ways as further described below.

[00115] Input/output interface(s) 908 are representative of functionality to allow a user to enter commands and information to computing device 902, and also allow information to be presented to the user and/or other components or devices using various input/output devices. Examples of input devices include a keyboard, a cursor control device (e.g., a mouse), a microphone (e.g., for voice inputs), a scanner, touch functionality (e.g., capacitive or other sensors that are configured to detect physical touch), a camera (e.g., which may employ visible or non-visible wavelengths such as infrared frequencies to detect movement that does not involve touch as gestures), and so forth. Examples of output devices include a display device (e.g., a monitor or projector), speakers, a printer, a network card, tactile-response device, and so forth. Thus, the computing device 902 may be configured in a variety of ways as further described below to support user interaction.

[00116] Computing device 902 also includes a data encryption and decryption system 914. Data encryption and decryption system 914 provides various data encryption and decryption support, including a trusted runtime and an inline cryptographic processor as discussed above. Data encryption and decryption system 914 can implement, for example, trusted runtime 112 and inline cryptographic processor 114 of Figs. 1 and 2.

[00117] Various techniques may be described herein in the general context of software, hardware elements, or program modules. Generally, such modules include routines, programs, objects, elements, components, data structures, and so forth that perform particular tasks or implement particular abstract data types. The terms "module," "functionality," and "component" as used herein generally represent software, firmware, hardware, or a combination thereof. The features of the techniques described herein are platform-independent, meaning that the techniques may be implemented on a variety of computing platforms having a variety of processors.

[00118] An implementation of the described modules and techniques may be stored on or transmitted across some form of computer-readable media. The computer-readable media may include a variety of media that may be accessed by the computing device 902.

By way of example, and not limitation, computer-readable media may include "computer-readable storage media" and "computer-readable signal media."

[00119] "Computer-readable storage media" refers to media and/or devices that enable persistent storage of information and/or storage that is tangible, in contrast to mere signal transmission, carrier waves, or signals per se. Thus, computer-readable storage media refers to non-signal bearing media. The computer-readable storage media includes hardware such as volatile and non-volatile, removable and non-removable media and/or storage devices implemented in a method or technology suitable for storage of information such as computer readable instructions, data structures, program modules, logic elements/circuits, or other data. Examples of computer-readable storage media may include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, hard disks, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or other storage device, tangible media, or article of manufacture suitable to store the desired information and which may be accessed by a computer.

[00120] "Computer-readable signal media" refers to a signal-bearing medium that is configured to transmit instructions to the hardware of the computing device 902, such as via a network. Signal media typically may embody computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as carrier waves, data signals, or other transport mechanism. Signal media also include any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared, and other wireless media.

[00121] As previously described, hardware elements 910 and computer-readable media 906 are representative of instructions, modules, programmable device logic and/or fixed device logic implemented in a hardware form that may be employed in some embodiments to implement at least some aspects of the techniques described herein. Hardware elements may include components of an integrated circuit or on-chip system, an application-specific integrated circuit (ASIC), a field-programmable gate array (FPGA), a complex programmable logic device (CPLD), and other implementations in silicon or other hardware devices. In this context, a hardware element may operate as a processing device that performs program tasks defined by instructions, modules, and/or logic embodied by

the hardware element as well as a hardware device utilized to store instructions for execution, e.g., the computer-readable storage media described previously.

[00122] Combinations of the foregoing may also be employed to implement various techniques and modules described herein. Accordingly, software, hardware, or program modules and other program modules may be implemented as one or more instructions and/or logic embodied on some form of computer-readable storage media and/or by one or more hardware elements 910. The computing device 902 may be configured to implement particular instructions and/or functions corresponding to the software and/or hardware modules. Accordingly, implementation of modules as a module that is executable by the computing device 902 as software may be achieved at least partially in hardware, e.g., through use of computer-readable storage media and/or hardware elements 910 of the processing system. The instructions and/or functions may be executable/operable by one or more articles of manufacture (for example, one or more computing devices 902 and/or processing systems 904) to implement techniques, modules, and examples described herein.

[00123] As further illustrated in Fig. 9, the example system 900 enables ubiquitous environments for a seamless user experience when running applications on a personal computer (PC), a television device, and/or a mobile device. Services and applications run substantially similar in all three environments for a common user experience when transitioning from one device to the next while utilizing an application, playing a video game, watching a video, and so on.

[00124] In the example system 900, multiple devices are interconnected through a central computing device. The central computing device may be local to the multiple devices or may be located remotely from the multiple devices. In one or more embodiments, the central computing device may be a cloud of one or more server computers that are connected to the multiple devices through a network, the Internet, or other data communication link.

[00125] In one or more embodiments, this interconnection architecture enables functionality to be delivered across multiple devices to provide a common and seamless experience to a user of the multiple devices. Each of the multiple devices may have different physical requirements and capabilities, and the central computing device uses a platform to enable the delivery of an experience to the device that is both tailored to the device and yet common to all devices. In one or more embodiments, a class of target devices is created and experiences are tailored to the generic class of devices. A class of

devices may be defined by physical features, types of usage, or other common characteristics of the devices.

[00126] In various implementations, the computing device 902 may assume a variety of different configurations, such as for computer 916, mobile 918, and television 920 uses. Each of these configurations includes devices that may have generally different constructs and capabilities, and thus the computing device 902 may be configured according to one or more of the different device classes. For instance, the computing device 902 may be implemented as the computer 916 class of a device that includes a personal computer, desktop computer, a multi-screen computer, laptop computer, netbook, and so on.

[00127] The computing device 902 may also be implemented as the mobile 918 class of device that includes mobile devices, such as a mobile phone, portable music player, portable gaming device, a tablet computer, a multi-screen computer, a wearable device, an Internet of Things (IoT) device, and so on. The computing device 902 may also be implemented as the television 920 class of device that includes devices having or connected to generally larger screens in casual viewing environments. These devices include televisions, set-top boxes, gaming consoles, and so on.

[00128] The techniques described herein may be supported by these various configurations of the computing device 902 and are not limited to the specific examples of the techniques described herein. This functionality may also be implemented all or in part through use of a distributed system, such as over a "cloud" 922 via a platform 924 as described below.

[00129] The cloud 922 includes and/or is representative of a platform 924 for resources 926. The platform 924 abstracts underlying functionality of hardware (e.g., servers) and software resources of the cloud 922. The resources 926 may include applications and/or data that can be utilized while computer processing is executed on servers that are remote from the computing device 902. Resources 926 can also include services provided over the Internet and/or through a subscriber network, such as a cellular or Wi-Fi network.

[00130] The platform 924 may abstract resources and functions to connect the computing device 902 with other computing devices. The platform 924 may also serve to abstract scaling of resources to provide a corresponding level of scale to encountered demand for the resources 926 that are implemented via the platform 924. Accordingly, in an interconnected device embodiment, implementation of functionality described herein may be distributed throughout the system 900. For example, the functionality may be

implemented in part on the computing device 902 as well as via the platform 924 that abstracts the functionality of the cloud 922.

[00131] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

CLAIMS

1. A method, implemented in a trusted runtime of a computing device, to rapidly protect data on a storage device, the method comprising:

receiving, at the trusted runtime, a request to generate a key for a portion of the storage device, the trusted runtime being isolated from other programs in the computing device;

in response to the request to generate the key,

generating the key,

persisting, using the trusted runtime, the key across power cycles of the computing device, and

provisioning an inline cryptographic processor of the computing device with the key so that subsequent writes to the portion of the storage device are encrypted by the inline cryptographic processor based on the key and data written to the portion of the storage device is protected; and

returning, from the trusted runtime in response to a request that protection of the portion be enabled, an indication that protection of the portion is enabled despite at least part of the portion being unencrypted.

2. A method as recited in claim 1, further comprising returning, in response to the request to generate the key, an identifier of the key to an operating system of the computing device.

3. A method as recited in claim 1, further comprising, in response to the request that protection of the portion be enabled:

protecting the key based on an authentication key associated with authentication credentials provided to the computing device by encrypting the key based on the authentication key;

ceasing having the key persisted across power cycles of the computing device; and

having the protected key persisted across power cycles of the computing device.

4. A method as recited in claim 3, further comprising:

receiving, after receiving the request that protection of the portion be enabled, a request to lock the portion; and

in response to the request to lock the portion, removing the key from both the trusted runtime and the inline cryptographic processor.

5. A computing device to rapidly protect data on a storage device, the computing device comprising:

a trusted runtime that is isolated from an operating system of the computing device and that is configured to:

generate a key for a portion of the storage device,

persist the key across power cycles of the computing device,

return, in response to a request that protection of the portion of the storage device be enabled, an indication that protection of the portion of the storage device is enabled despite one or more parts of the portion of the storage device being unencrypted; and

an inline cryptographic processor to protect data on the storage device by being configured to:

receive the key from the trusted runtime,

encrypt subsequent writes to the portion of the storage device based on the key, and

decrypt subsequent reads from the portion of the storage device based on the key.

6. A computing device as recited in claim 5, the trusted runtime being further configured to, in response to the request that protection of the portion be enabled:

protect the key based on an authentication key associated with authentication credentials provided to the computing device;

cease having the key persisted across power cycles of the computing device; and

have the protected key persisted across power cycles of the computing device.

7. A computing device as recited in claim 6, the trusted runtime being further configured to:

receive, after receiving the request that protection of the portion be enabled, a request to lock the portion; and

in response to the request to lock the portion, remove the key from both the trusted runtime and the inline cryptographic processor.

8. A computing device as recited in claim 7, the trusted runtime being further configured to:

receive, after removing the key from both the trusted runtime and the inline cryptographic processor, a request to unlock the portion; and

in response to the request to unlock the portion:

obtain, based on the protected key, the key from the protected key, and

provision the inline cryptographic processor of the computing device with the obtained key.

9. A computing device as recited in claim 6, the trusted runtime being further configured to:

receive, after receiving the request that protection of the portion be enabled, a request to disable protection of the portion; and

in response to the request to disable protection of the portion:

obtain, based on the protected key, the key from the protected key, and

persist the obtained key using the trusted runtime.

10. A computing device as recited in claim 6, the trusted runtime being further configured to:

receive, after receiving the request that protection of the portion be enabled, a request for migration or recovery of the key; and

in response to the request for migration or recovery of the key, reveal the protected key to the requester.

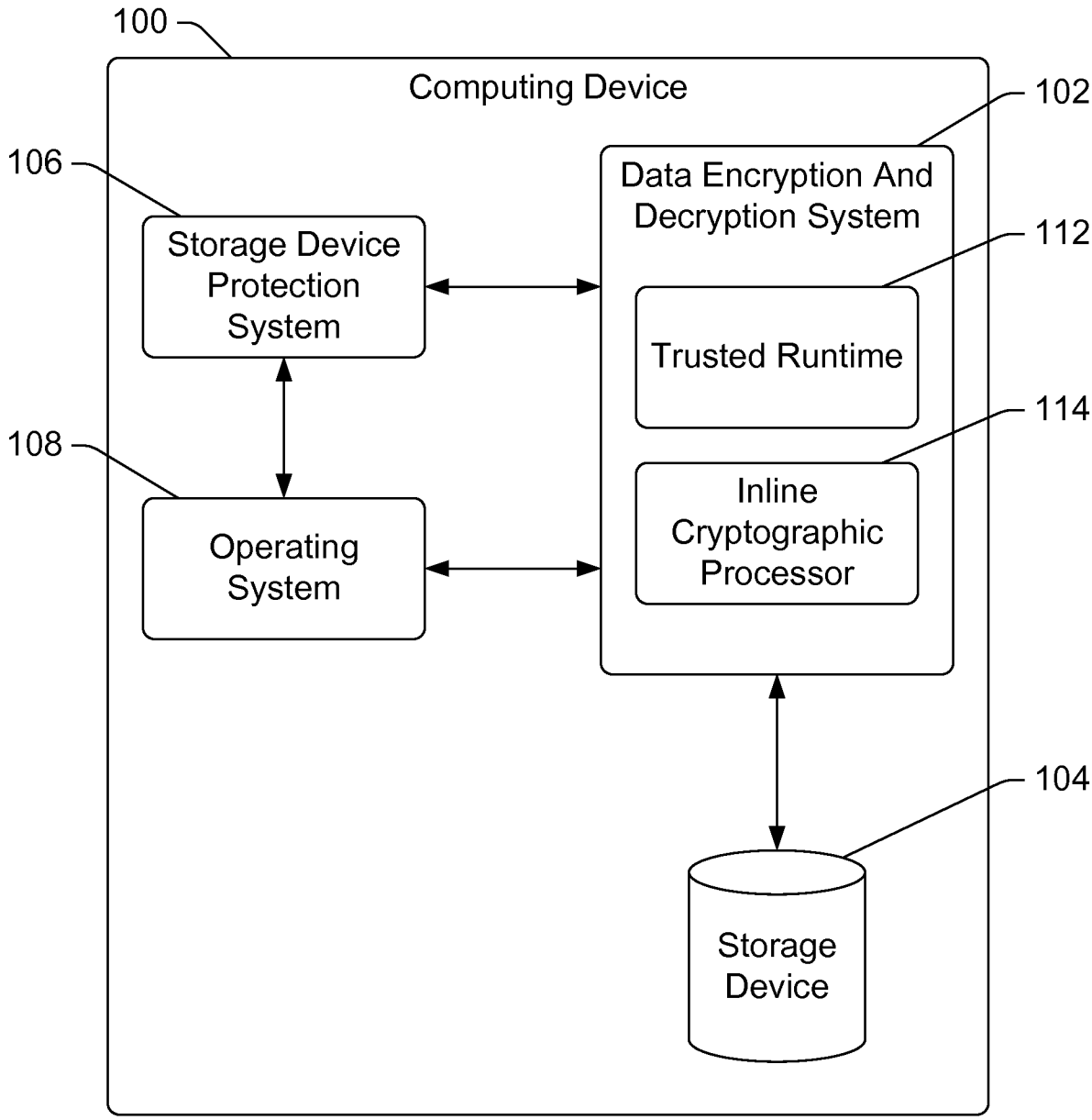
100
Computing Device
102
106
Data Encryption And Decryption System
Storage Device Protection System
112
Trusted Runtime
114
108
Inline Cryptographic Processor
Operating System
104
Storage Device

Fig. 1

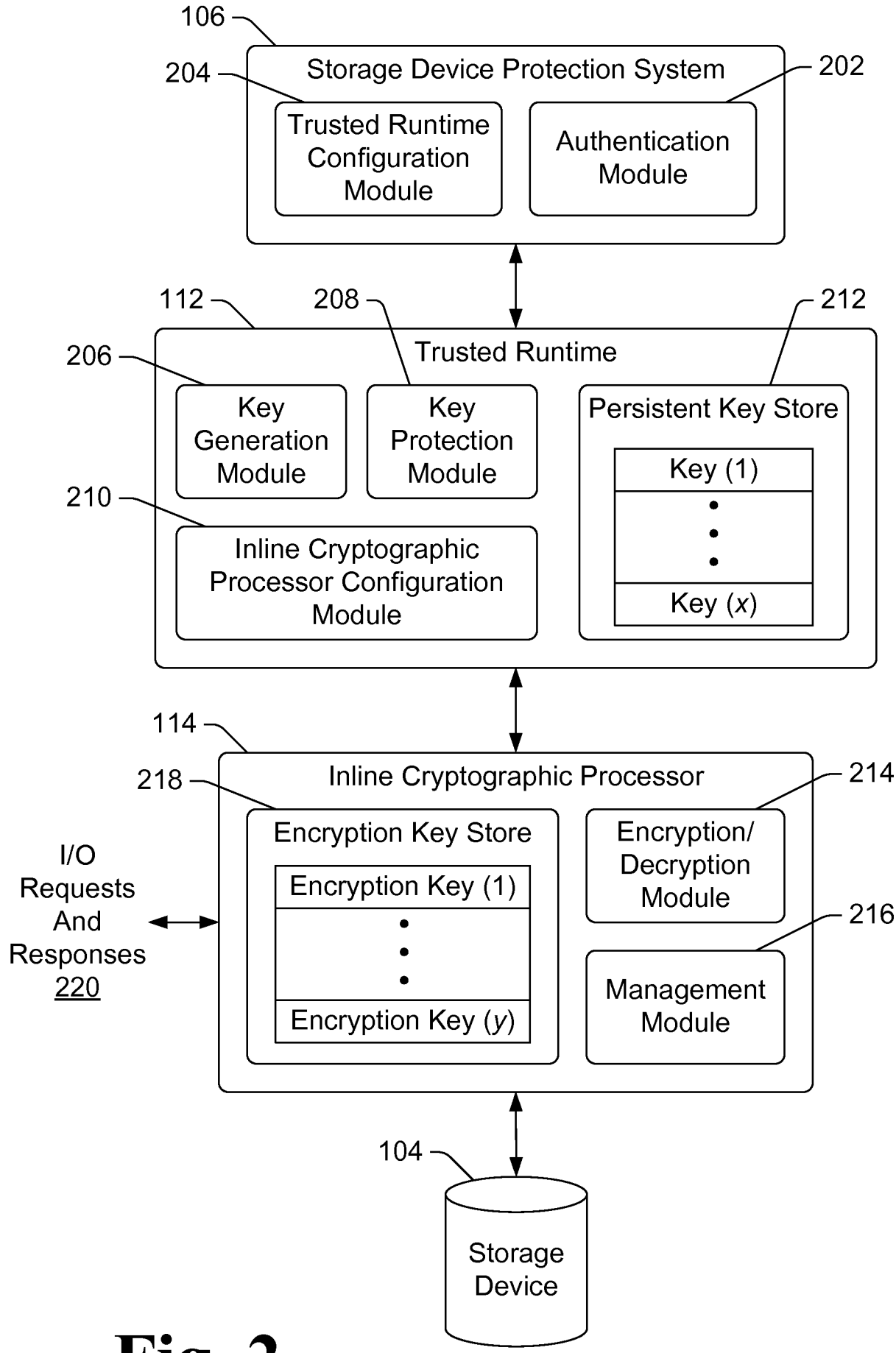
200
106
Storage Device Protection System
204
202
Trusted Runtime Configuration Module
Authentication Module
112
208
212
Trusted Runtime
206
Key Generation Module
Key Protection Module
Persistent Key Store
Key (1)
Key (x)
210
Inline Cryptographic Processor Configuration Module
114
Inline Cryptographic Processor
218
214
Encryption Key Store
Encryption Key (1)
Encryption Key (y)
Encryption/ Decryption Module
216
Management Module
I/O Requests And Responses
220
104
Storage Device

Fig. 2

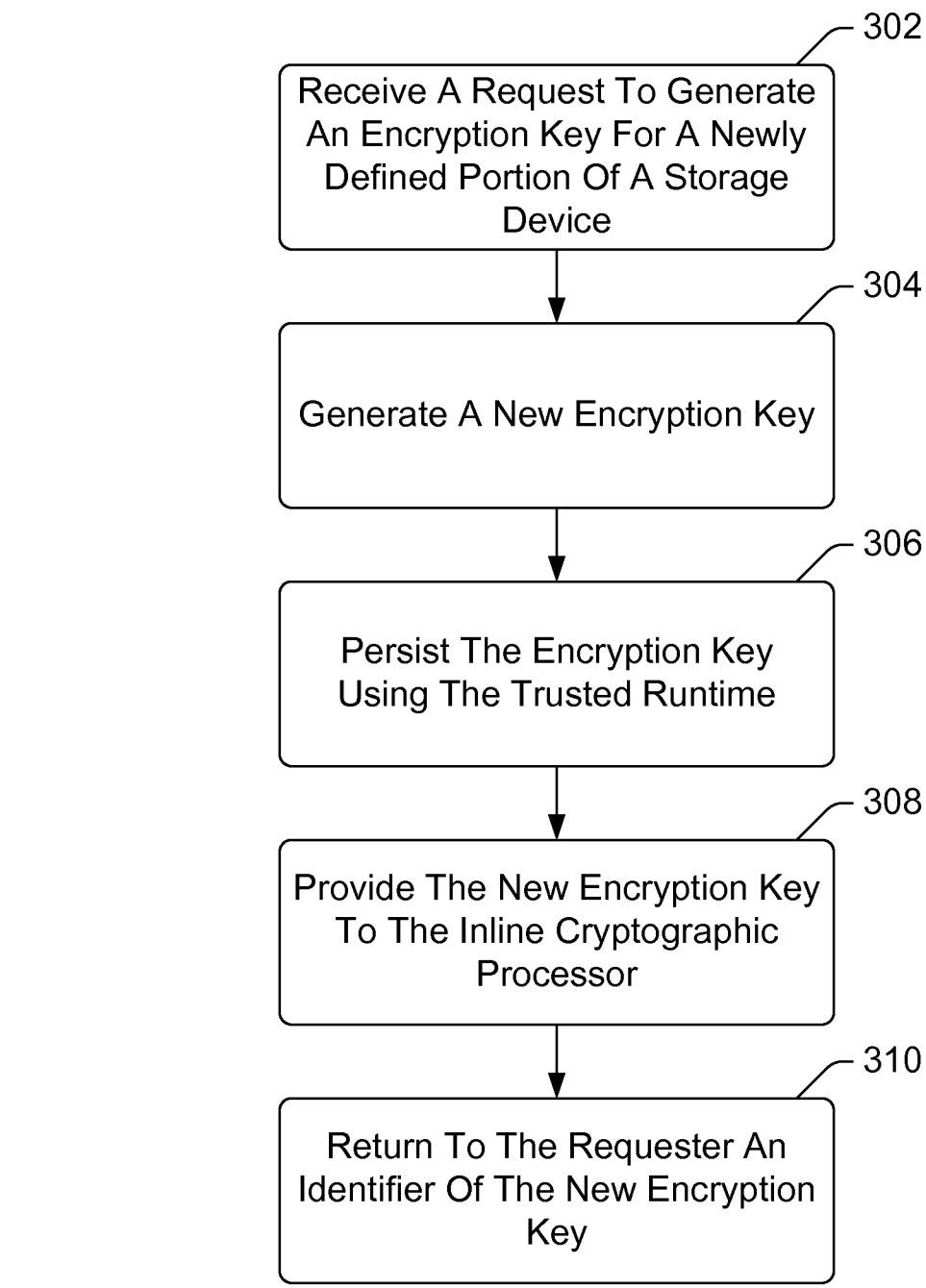
300
302
Receive A Request To Generate An Encryption Key For A Newly Defined Portion Of A Storage Device
304
Generate A New Encryption Key
306
Persist The Encryption Key Using The Trusted Runtime
308
Provide The New Encryption Key To The Inline Cryptographic Processor
310
Return To The Requester An Identifier Of The New Encryption Key

Fig. 3

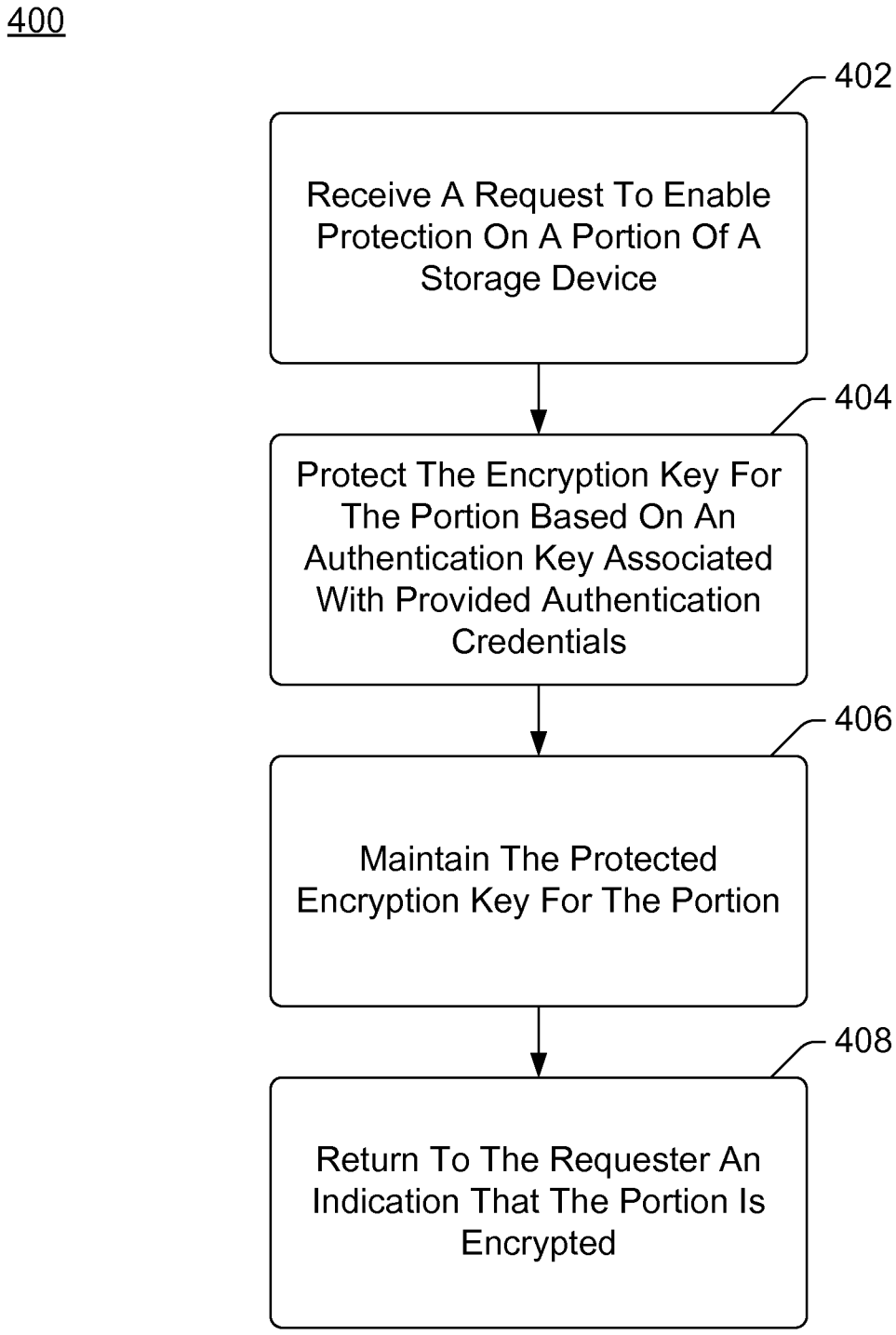
400
402
Receive A Request To Enable Protection On A Portion Of A Storage Device
404
Protect The Encryption Key For The Portion Based On An Authentication Key Associated With Provided Authentication Credentials
406
Maintain The Protected Encryption Key For The Portion
408
Return To The Requester An Indication That The Portion Is Encrypted

Fig. 4

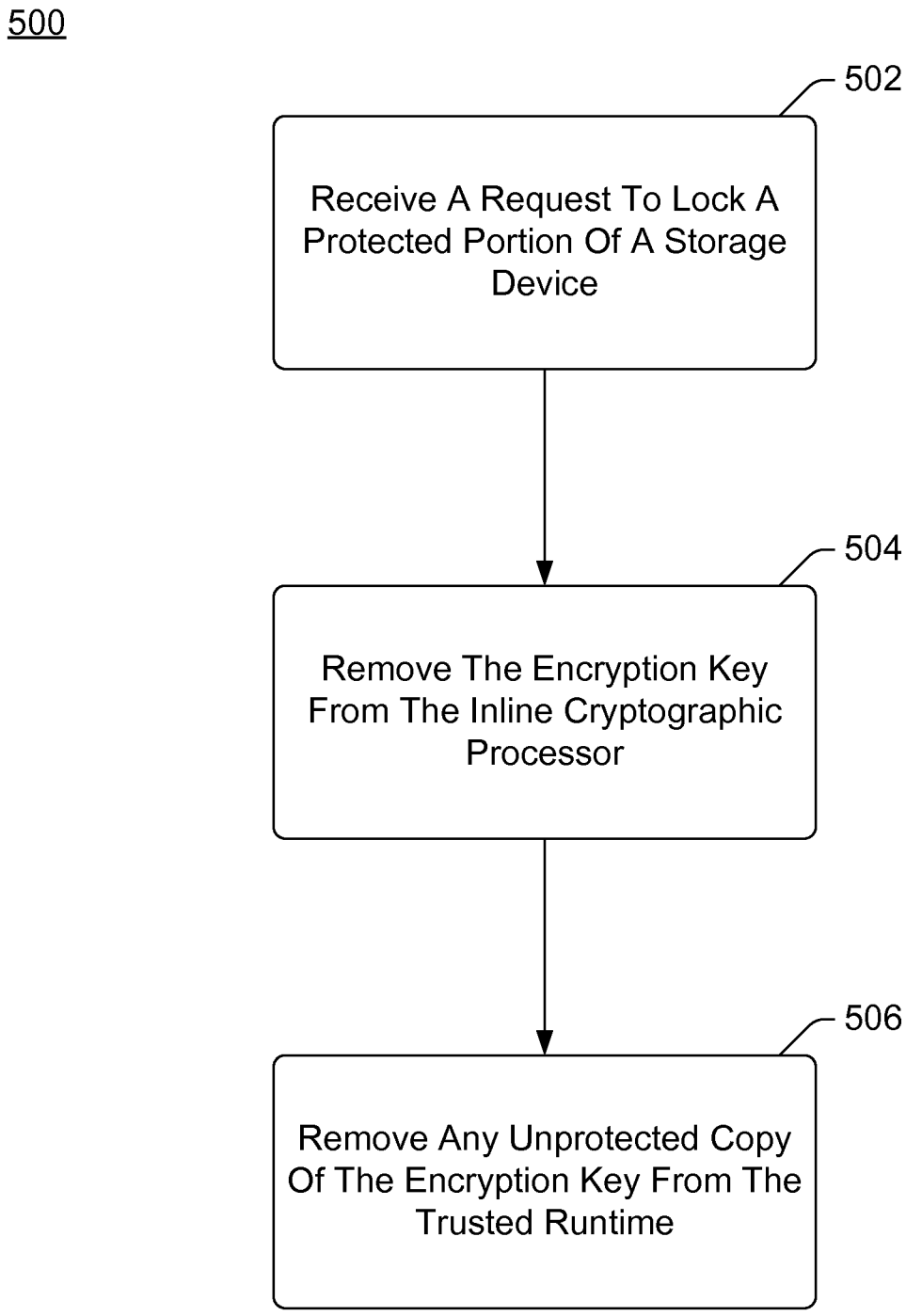
500
502
Receive A Request To Lock A Protected Portion Of A Storage Device
504
Remove The Encryption Key From The Inline Cryptographic Processor
506
Remove Any Unprotected Copy Of The Encryption Key From The Trusted Runtime

Fig. 5

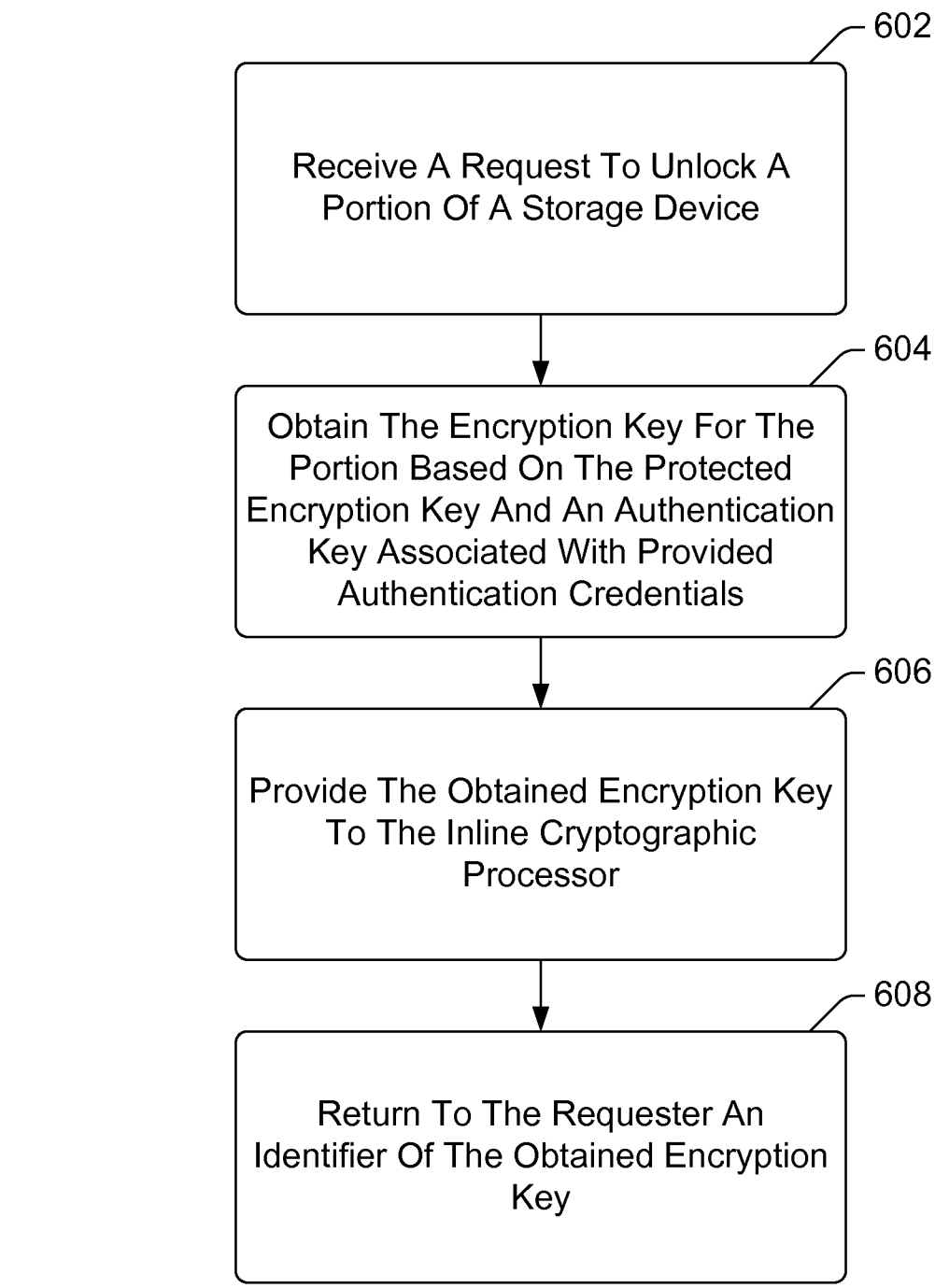
600
602
Receive A Request To Unlock A Portion Of A Storage Device
604
Obtain The Encryption Key For The Portion Based On The Protected Encryption Key And An Authentication Key Associated With Provided Authentication Credentials
606
Provide The Obtained Encryption Key To The Inline Cryptographic Processor
608
Return To The Requester An Identifier Of The Obtained Encryption Key

Fig. 6

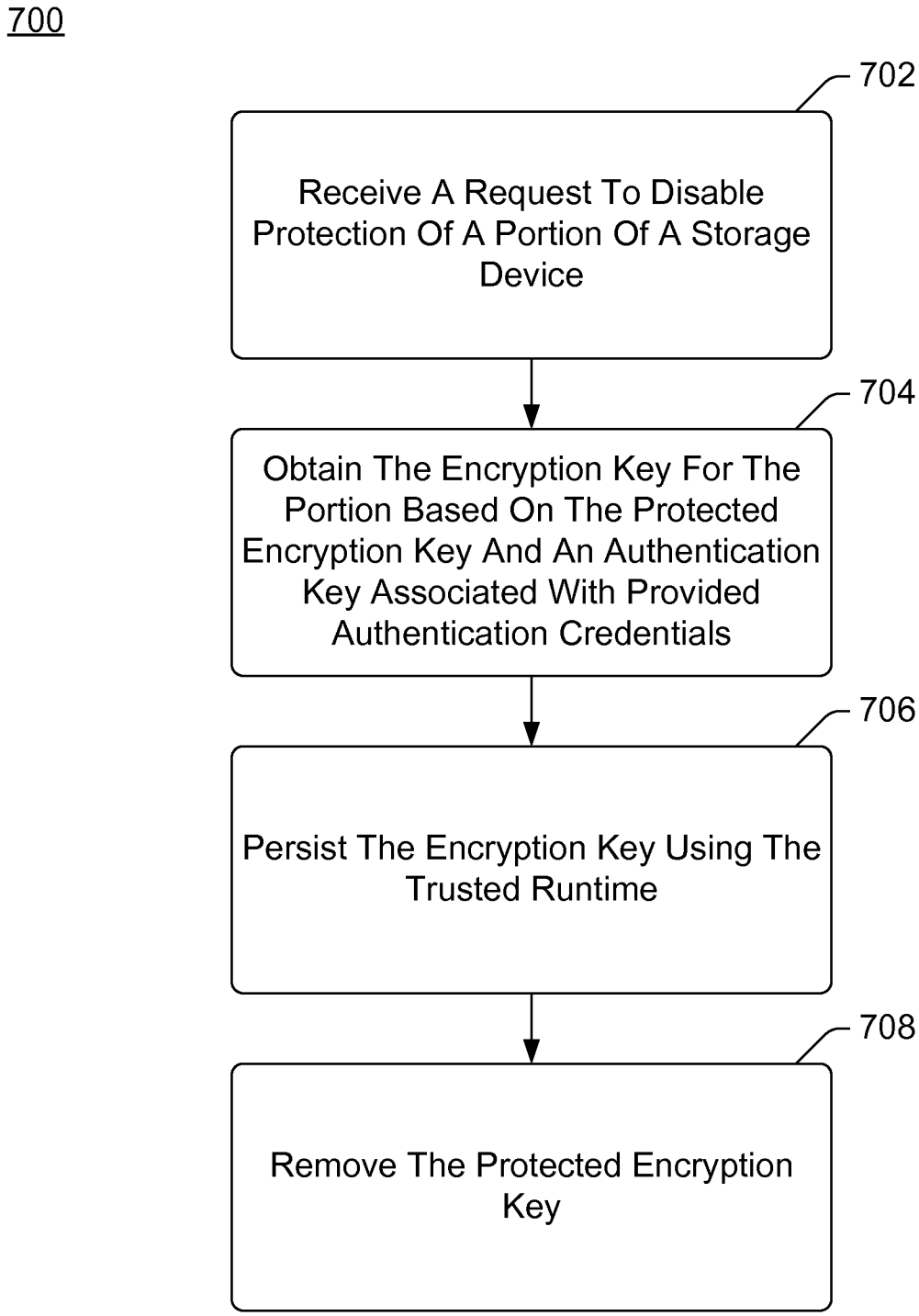
700
702
Receive A Request To Disable Protection Of A Portion Of A Storage Device
704
Obtain The Encryption Key For The Portion Based On The Protected Encryption Key And An Authentication Key Associated With Provided Authentication Credentials
706
Persist The Encryption Key Using The Trusted Runtime
708
Remove The Protected Encryption Key

Fig. 7

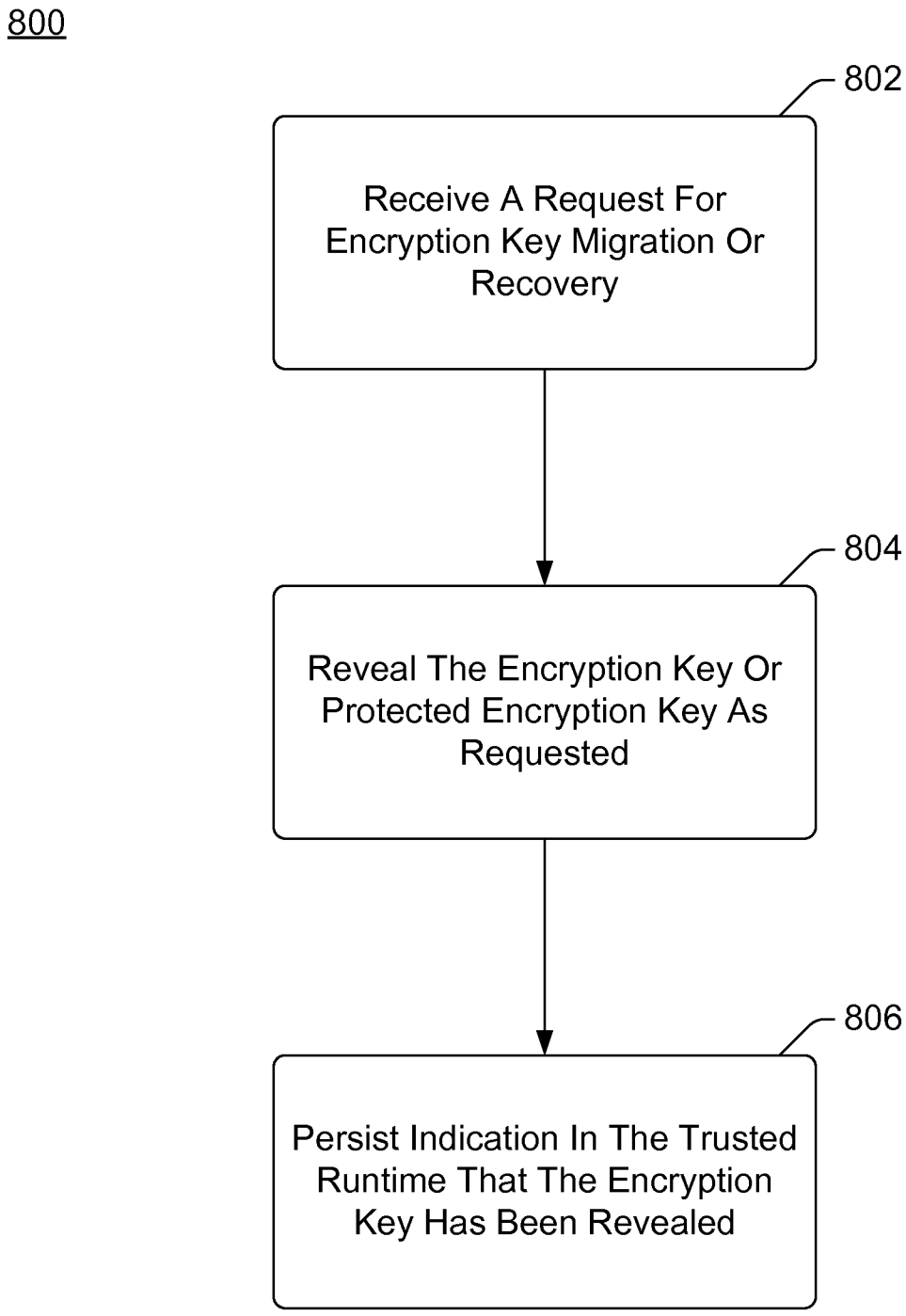
800
802
Receive A Request For Encryption Key Migration Or Recovery
804
Reveal The Encryption Key Or Protected Encryption Key As Requested
806
Persist Indication In The Trusted Runtime That The Encryption Key Has Been Revealed

Fig. 8

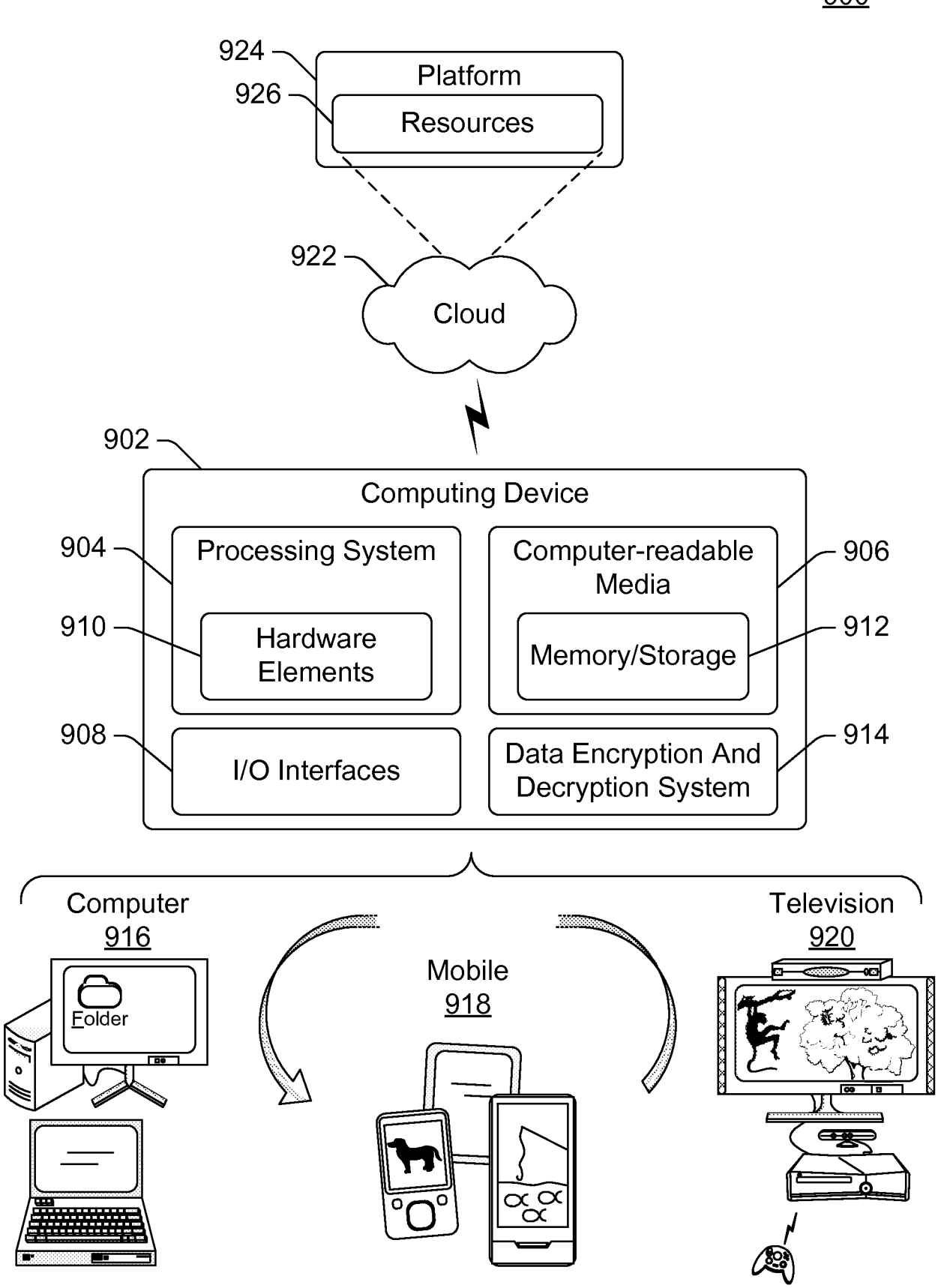
900
924
Platform
926
Resources
922
Cloud
902
Computing Device
904
Processing System
910
Hardware Elements
908
I/O Interfaces
906
Computer-readable Media
912
Memory/Storage
914
Data Encryption And Decryption System
Computer
916
Folder
Mobile
918
Television
920

Fig. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2015/021125

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/74 G06F21/78
ADD.

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)
G06F

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

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 2 562 675 A1 (GEMALTO SA [FR]) 27 February 2013 (2013-02-27) paragraphs [0013] - [0017], [0021]; claims 1,5; figures 1-3	1-10
X	US 2011/087890 A1 (MUNSIL JEFFREY L [US] ET AL) 14 April 2011 (2011-04-14) paragraphs [0009], [0046] - [0048], [0058]; figures 1,4,7	1-10
Y	US 8 416 954 B1 (RAIZEN HELEN S [US] ET AL) 9 April 2013 (2013-04-09) figure 5	1-10
Y	US 2013/086691 A1 (FIELDER GUY [US]) 4 April 2013 (2013-04-04) paragraphs [0027] - [0028], [0123]; figure 1c,	1-10

☐ 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 another citation or other special reason (as specified)

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 June 2015	03/07/2015

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Widera, Sabine

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2015/021125

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 2562675 A1	27-02-2013	EP 2562675 A1	27-02-2013
		EP 2745233 A1	25-06-2014
		US 2014189373 A1	03-07-2014
		WO 2013026662 A1	28-02-2013
US 2011087890 A1	14-04-2011	US 2011087890 A1	14-04-2011
		US 2011087898 A1	14-04-2011
US 8416954 B1	09-04-2013	US 8416954 B1	09-04-2013
		US 8966281 B1	24-02-2015
US 2013086691 A1	04-04-2013	US 2013086691 A1	04-04-2013
		US 2013262882 A1	03-10-2013