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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2020 (22.05.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/101771 A2

(51) International Patent Classification:
Not classified

(21) International Application Number:
PCT/US2019/045610

(22) International Filing Date:
08 August 2019 (08.08.2019)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
62/728,235 07 September 2018 (07.09.2018) US
16/371,191 01 April 2019 (01.04.2019) US

(71) Applicant: **RAYTHEON COMPANY** [US/US]; 870 Winter Street, Waltham, Massachusetts 02451-1449 (US).

(72) Inventors: **ARENO, Matthew C.**; 3750 E. Palm Valley Blvd., Unit 3, Round Rock, Texas 78665 (US). **COLEMAN, Jody R.**; 1287 Muddy Ford Road, Chapin, South Carolina 29036 (US). **ADAMS, Daniel**; 1440 Burdette Road, Gray Court, South Carolina 29645 (US).

(74) Agent: **DURKEE, Paul D.** et al.; Daly, Crowley, Mofford & Durkee, LLP, One University Avenue, Suite 201B, Westwood, Massachusetts 02090 (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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ,

(54) Title: SYSTEM AND METHOD FOR BOOTING PROCESSORS WITH ENCRYPTED BOOT IMAGE

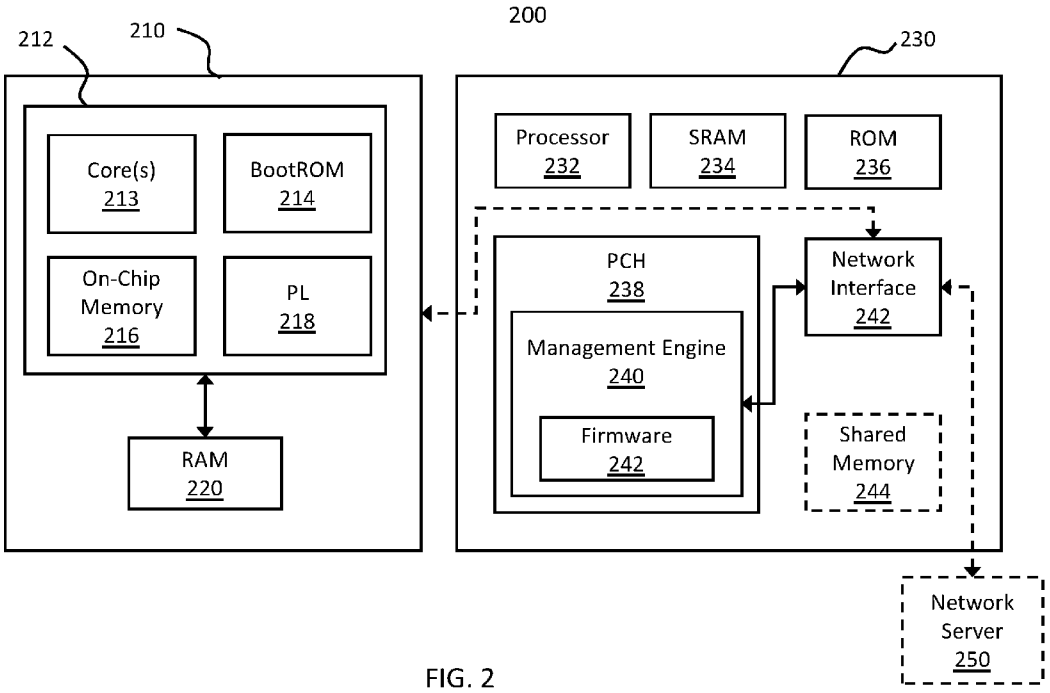

FIG. 2

(57) Abstract: The concepts, systems and methods described herein are directed towards a method for secure booting. The method is provided to including: loading and executing a firmware in a Management Engine (ME) of a system; establishing, by the ME, a communication channel to a security device; receiving, by the ME, an encrypted boot image from the security device; decrypting, by the ME, the encrypted boot image; storing, by the ME, the decrypted boot image in a secured storage medium; and resetting the system using the decrypted boot image in the secured storage medium.

WO 2020/101771 A2

[Continued on next page]

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:

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

SYSTEM AND METHOD FOR BOOTING PROCESSORS WITH ENCRYPTED BOOT IMAGE

BACKGROUND

[0001] Protection of firmware codes, such as Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI), is necessary to establish a secure booting and executing environment. However, such firmware codes may not be protectable during transit from a storage to a processor when the processor is only able to execute unencrypted images. Although these firmware codes may be cryptographically signed by a provider, they are unencrypted and may contain unsigned 3rd party software elements. Thus, it may be possible to extract the firmware for reverse engineering or perform other security attacks using the unencrypted firmware codes. The inability to use encrypted firmware may pose security risks for programs with high-security requirements.

SUMMARY

[0002] In accordance with the concepts, techniques and systems described herein provide methods and apparatus for transmitting an encrypted boot image to a system, decrypting locally, and executing on a processor in the system.

[0003] According to one illustrative embodiment, a method for secure booting may include: loading and executing a firmware in the ME; establishing, by the ME, a communication channel to a security device; receiving, by the ME, an encrypted boot image from the security device; decrypting, by the ME, the encrypted boot image; storing, by the ME, the decrypted boot image in the secured storage medium; and resetting the system using the decrypted boot image in the secured storage medium.

[0004] In one aspect, the system may comprise a System-on-a-Chip (SoC).

[0005] In one aspect, the encrypted boot image may be encrypted using a unique key that is generated by the ME.

[0006] In one aspect, the unique key may comprise a unique-per-processor key that is generated based upon unique key material of the processor, wherein the unique key material of the processor is accessible by the ME.

[0007] In one aspect, the method may further include: loading, by the ME, an unencrypted boot image; encrypting, by the ME, the unencrypted boot image using the unique key; and storing, by the ME, the encrypted boot image in the security device. In another aspect, the method may further include: generating, by the ME, an Advanced Encryption Standard (AES) key; encrypting, by the ME, the unencrypted boot image using the AES key; encrypting, by the ME, the AES key; and storing, by the ME, the encrypted boot image and the encrypted AES key in the security device.

[0008] In one aspect, the security device may comprise a Hardware Root of Trust (HRoT) device.

[0009] In one aspect, the security device may comprise a network server.

[0010] In one aspect, the communication channel to the security device may comprise an encrypted communication channel.

[0011] In one aspect, the method may further include: transferring data to and from the security device via the communication channel.

[0012] According to another illustrative embodiment, a system for secure booting may include: a processor and memory; a management engine (ME); and a secured storage medium. Herein, the ME may be configured to: load and execute a firmware; establish a

communication channel to a security device; receive an encrypted boot image from the security device; decrypt the encrypted boot image; and store the decrypted boot image in the secured storage medium. Furthermore, the system may reset using the decrypted boot image in the secured storage medium.

[0013] In one aspect, the ME may be further configured to: load an unencrypted boot image from the security device; generate a unique-per-processor key; encrypt the unencrypted boot image using the unique-per-processor key; and store the encrypted boot image in the security device.

[0014] The details of one or more embodiments of the disclosure are outlined in the accompanying drawings and the description below. Other features, objects, and advantages of the disclosure will be apparent from the description and drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] The foregoing features may be more fully understood from the following description of the drawings in which:

[0016] FIG. 1 is a block diagram of a conventional Management Engine (ME) system;

[0017] FIG. 2 is a block diagram showing an architecture of a Hardware Root of Trust (HRoT) device and a host platform comprising an ME according to the concepts described herein;

[0018] FIG. 3 is a flowchart of a secure booting process according to the concepts described herein;

[0019] FIG. 4 is a flowchart of creating an encrypted boot image according to the concepts described herein; and

[0020] FIG. 5 is an illustrative implementation of a security system described in FIG. 2 according to the concepts described herein.

DETAILED DESCRIPTION

[0021] Relative descriptions used herein, such as left, right, up, and down, are with reference to the figures, are merely relative and not meant in a limiting sense. Additionally, for clarity, common items and circuitry, such as integrated circuits, resistors, capacitors, transistors, and the like, have not been included in the figures, as can be appreciated by those of ordinary skill in the pertinent art. Unless otherwise specified, the illustrated embodiments may be understood as providing illustrative features of varying detail of certain embodiments, and therefore, unless otherwise specified, features, components, modules, elements, and/or aspects of the illustrations can be otherwise combined, interconnected, sequenced, separated, interchanged, positioned, and/or rearranged without materially departing from the disclosed concepts, systems, or methods. Additionally, the shapes and sizes of components are intended to be only illustrative and unless otherwise specified, can be altered without materially affecting or limiting the scope of the concepts sought to be protected herein.

[0022] For convenience, certain introductory concepts and terms used in the specification are collected here.

[0023] As used herein, the term Management Engine (ME), INTEL Management Engine (IME), or Converged Security and Management Engine (CSME) (hereinafter "ME") is used to describe an autonomous subsystem incorporated in some processor chipsets

produced by INTEL. The ME is a part of Intel Active Management Technology for performing tasks on a system remotely. The ME is located in a Platform Controller Hub (PCH) of an Intel-based motherboard or a System-on-a-Chip (SoC).

[0024] As used herein, the term "Root of Trust (RoT)" refers to a trusted computing module that provides a set of functions that are trusted by other modules in a computing environment, such as an Operating System (OS). The RoT may serve as an independent computing module providing authenticity of a computer or mobile device in which it is embedded. The term "Hardware Root of Trust (HRoT)" device is used to describe a hardware device that provides RoT functions. The HRoT device generally provides, but not limited to, performing device authentication to ensure that hardware has not been tampered with; verifying the authenticity of software, particularly boot images, to ensure they haven't been tampered with; providing One-Time Programmable (OTP) memory for secure key storage to facilitate encryption; and/or ensuring that the system is able to be brought into a known and trusted state.

[0025] Referring now to FIG. 1, a conventional Management Engine (ME) 100 is presented. The ME is a separate processing subsystem located inside a modern motherboard or SoC package. The ME may have a processor 110, Read-only Memory (ROM) 130, Static Random-Access Memory (SRAM) 140, cache 150, and a cryptography engine 160. The ME 100 may include other elements or modules including, but not limited to, an interrupt controller, high precision and watch dog timer, Direct Memory Access (DMA) engine, and others. The processor 110 and other elements are connected via an internal bus 120. As can be appreciated by a person in the pertinent art, there can be variations in the architecture of the ME. The ME is located inside of a Platform Controller Hub (PCH), as shown in FIG. 2.

[0026] The ME 100 is running as long as its host system (motherboard or SoC, e.g., 230 in FIG. 2) has power, regardless of whether the host processor is executing. That is, even if a computer system is turned off, the ME is capable of continuing to execute as long as there is a power connection to the host system 230. The ME has its own Operating System (OS), applications, and memory address space to support its security and management operations. One of the responsibilities of the ME is to generate and maintain device specific cryptographic keys. These keys may be accessible from outside through a collection of technologies, including Identity Protection Technology (IPT), Trusted eXecution Engine (TXE), and Dynamic Application Loader (DAL). The IPT, TXE, and DAL are technologies provided by INTEL. Through the technologies, an authorized developer can create custom applications that can run on the processor 110 of the ME 100.

[0027] Referring now to FIG. 2, an illustrative security system 200 may include a Hardware Root of Trust (HRoT) device 210 and a host platform. The HRoT device 210 and the host platform 230 may be respective independent systems with similar setups. Each of the systems may have its own processor, memory and other interfaces. In embodiments, the host platform 230 may include a central processing unit (CPU) 232, Static RAM 234, ROM 236, Platform Controller Hub (PCH) 238, and network interface 242. An ME 240 that is located in the PCH 238 may have a substantially similar architecture as the ME 100 presented in conjunction with FIG. 1. That is, the ME 240 may have its own processor (e.g., 110 in FIG. 1) and one or more memory modules.

[0028] In embodiments, the HRoT device 210 may include a multi-core processor 212 and memory 220. The processor 212 of the HRoT device 210 may include one or more cores 213 configured similarly with the processor 232 of the host platform 230. The processor 212 may also include a boot read-only memory (BootROM) 214, on-chip

memory 216, and programmable logic 218. The processor 212 of an HRoT device 210 generally requires higher security features than a conventional processor being used in a system that does not require higher security. In some embodiments, the HRoT device 210 may comprise a substantially similar architecture with the host platform 230. The BootROM 214 can be a place where the HRoT device 210 loads and executes secure boot codes that initialize a secure booting procedure of the HRoT device 210. The on-chip memory 216 may perform as storage for boot loader codes that are loaded and executed by the secure boot codes. The on-chip memory 216 may be volatile and maintain OS and other application for operation of the processor 212. The PL 218 may perform as a storage for a security monitoring application that is loaded and executed by the boot loader codes. As can be appreciated by those of ordinary skill in the pertinent art, the processor 212 may have more or lesser number of on-chip and off-chip storages than what is described in FIG. 2. It should also be appreciated that the structure of processor 232 in the host platform 230 may have a similar structure with the processor 212 in the HRoT device 210. In other embodiments, the secure boot code and boot loader codes may be loaded and executed in the same on-chip or off-chip memory, for example BIOS flash in an INTEL processor. Further, a security monitoring app may be loaded and executed in the same memory or storage where the boot codes are loaded.

[0029] The ME 210 is connected to a host platform 230 through a communication channel. The communication channel may be Host Embedded Controller Interface (HECI) bus. Through the HECI, the ME can communicate to the OS of the host platform. The ME 210 may also access unique key material stored inside a processor 232 of its host platform. The unique key material of the processor is uniquely signed to each processor by a provider of the processor. Further, the ME 210 may have access to an on-board

network interface device 242. The ME 210 may have access to the network interface even when the system is in a low-power state or a sleep mode, without the host OS knowing that the network interface 242 is being accessed by the ME 210.

[0030] When the host platform 230 is first booted, the ME 240 initially loads an initial firmware from a storage medium. In some embodiments, the storage that stores the initial firmware may be the ROM or SRAM (e.g., 130, 140 in FIG. 1) of the ME 240. In other embodiments, the initial firmware may be stored in and loaded from a shared memory located on the host platform 230 or BIOS flash in an INTEL processor (not shown). Herein, the initial firmware may be a custom firmware (sometimes referred as a "custom ME application"), which can be loaded into the internal SRAM (e.g., 140 in FIG. 1) of the ME 240 during the boot sequence. The custom firmware 242 then may access the network interface 242 to establish a secure communication channel with the HRoT device 210. As stated above, the ME 240 may access the network interface 242 even when the host platform 230 is not running. In some embodiments, the ME 240 and the HRoT device 210 may perform an authorization procedure before establishing the secure communication channel.

[0031] Once a secure communication channel is established between the ME 240 and the HRoT device 210, the custom firmware 242 in the ME 240 may receive an encrypted boot image (BIOS or UEFI) from the HRoT device 210. Further, the ME 240 and the HRoT device 210 may transfer data to each other through the secure communication channel. In some embodiments, the ME may receive the encrypted boot image from another type of security device, such as a network server 250 that is accessible by the ME 240. The network server 250 may be a Network-Attached Storage (NAS) or any other suitable device that functions as a storage device. In embodiments, the ME 240 can access

a cryptographic key to be used to decrypt the received encrypted boot image. In some embodiments, the cryptographic key may be generated by the ME 240 and stored in the ME 240. Herein, the encrypted boot image may be previously encrypted by the ME 240 and stored in the HRoT device 210 or the network server 250, as will be described in detail in conjunction with FIG. 4.

[0032] When the firmware 242 of the ME 240 decrypts the encrypted boot image, the ME 240 stores the decrypted boot image into a secured storage medium. In embodiments, the secured storage medium may be the shared memory 244 of the host platform 230. In other embodiments, the secured storage medium may be the SRAM 234 or any other suitable storage that the processor 232 may access during its booting sequence. Then, the firmware 242 of the ME 240 may restart the host platform 230. When the host platform restarts, the ME 240 may intervene during the booting process such that the processor 232 fetches the decrypted boot image instead of an unencrypted and non-trusted boot image. Herein, the processor 232 may boot using an encrypted image from a secured storage medium, such as the HRoT device 210. Accordingly, fetching unencrypted boot image for reverse engineering or other security attacks may be prevented. In some embodiments, the encrypted image may be stored in an external flash element along with the unencrypted image, stored on a peripheral device of the system (such as the HRoT device 210), or stored on a networked server 250.

[0033] In embodiments, instead of immediately restarting the process 232, the ME 240 may hold the processor 232 in reset until the ME 240 is ready with the decrypted BIOS code to serve the processor 232. In some embodiments, the ME 240 may fetch one or more blocks that are being executed from the encrypted boot image, decrypt, send to the

processor 232. Accordingly, the ME 240 may use less storage space while executing than holding the entire boot image.

[0034] In embodiments, the encryption of the boot image could be performed by creating a random Advanced Encryption Standard (AES) key and using the random AES key to encrypt the boot image. That random AES key could be encrypted with a unique-per-device key generated by the ME 240. In some embodiments, the ME 240 may generate the unique-per-device key based upon the unique key material in the processor 232. The unique key material in the processor 232 may be accessible only by the ME 240. The ME 240 may store the encrypted boot image and associated encrypted AES key in a storage device, such as the HRoT device 210. In the embodiments in which a Root of Trust (RoT) device (e.g., the HRoT device 210) is used or an RoT relationship is established with an external system (e.g., the network server 250), a shared secret material, such as a public key or certificate from a trusted server that are associated with the provisioning of the encrypted image, may be established between the RoT device and the ME 240 based upon the unique-per-device cryptographic key generated by the ME 240. In some embodiments, based upon this shared secret material, the ME 240 and the RoT device may establish a session key during boot-up that would allow the boot image to be uniquely encrypted every time the host platform 230 boots. Accordingly, it may be possible to provide even more secure executing environments because the boot image is encrypted with a different key whenever the host platform 230 reboots. In other embodiments, the ME 240 may support "asymmetric" keys for the encryption. Herein, the ME 240 may generate a private/public key pair and send the public key to a network server 250, which then uses the public key to encrypt the boot image. When the ME 240

gets the encrypted boot image, the ME 240 may use the pairing private key to encrypt the boot image.

[0035] FIGs. 3-4 are flow diagrams illustrating the processing performed by an ME (e.g., 240 in FIG. 2) and an HRoT device (e.g., 210 in FIG. 2). Rectangular elements (typified by element 310 in FIG. 3), herein denoted "processing blocks," represent computer software instructions or groups of instructions and diamond shaped elements (not shown), herein denoted decision blocks represent computer software instructions or groups of instructions which affect the flow of the processing blocks. The processing blocks may represent steps performed by functionally equivalent circuits such as a digital signal processor (DSP) circuit or an application specific integrated circuit (ASIC). The flow diagrams do not depict the syntax of any particular programming language but rather illustrate the functional information one of ordinary skill in the art requires to fabricate circuits or to generate computer software to perform the processing required of the particular apparatus. It should be noted that many routine program elements, such as initialization of loops and variables and the use of temporary variables may be omitted for clarity. The particular sequence of blocks described is illustrative only and can be varied without departing from the spirit of the concepts, structures, and techniques sought to be protected herein. Thus, unless otherwise stated, the blocks described below are unordered meaning that, when possible, the functions represented by the blocks can be performed in any convenient or desirable order.

[0036] Referring now to FIG. 3, a secure booting process 300 may be carried out by an ME (e.g., 240 in FIG. 2) and an HRoT device (e.g., 210 in FIG. 2). In processing step 310, when a host platform (e.g., 230 in FIG. 2) is first booted, the ME 240 may load an initial firmware from a storage device. Herein, the initial firmware may be a custom

firmware that is loaded into the internal SRAM (e.g., 140 in FIG. 1) of the ME 240. In processing step 320, the custom firmware (e.g., 242 in FIG. 2) may access a network interface 242 and establish a secure communication channel with the HRoT device 210. Once a secure communication channel is established between the ME 240 and the HRoT device 210, in processing step 330, the custom firmware 242 may receive an encrypted boot image (BIOS or UEFI) from the HRoT device 210. In some embodiments, the ME may receive the encrypted boot image from another type of security device, such as a network server 250. In processing step 340, the ME 240 decrypts the received encrypted boot image using an encryption key. In processing step 350, the ME 240 may store the decrypted boot image into a secured storage medium. Then, in processing step 360, the firmware 242 of the ME 240 resets the host platform 230. When the host platform 230 restarts, the processor 232 may fetch the decrypted boot image from the secured storage medium and boot the system using the decrypted boot image.

[0037] Referring now to FIG. 4, a process 400 for creating an encrypted boot image may be carried out by an ME (e.g., 240 in FIG. 2). In processing step 410, the ME 240 may fetch an unencrypted boot image and load the image in the ME 240. In embodiments, the unencrypted boot image may be fetched from an HRoT device (e.g., 210 in FIG. 2). In other embodiments, the unencrypted boot image may come from any suitable storage that the ME 240 can access. In processing step 420, the ME 240 may generate a unique-per-device key corresponding to a processor (e.g., 232 in FIG. 2) of its host platform (e.g., 230 in FIG. 2). In some embodiments, the ME 240 may generate the unique-per-device key based upon unique key material in the processor 232. In processing step 430, the ME 240 may encrypt the boot image using the unique-per-device key. In some embodiments, a random Advanced Encryption Standard (AES) key may be generated, and the random

AES key is used to encrypt the boot image. Herein, the random AES key is encrypted with the unique-per-device key generated by the ME 240 and stored along with the encrypted boot image. In processing step 440, the ME 240 may store the encrypted boot image in a security device, such as the HRoT device 210 or a network server 250.

[0038] Referring now to FIG. 5, an illustrative implementation of a processing device 500 which may be suitable to implement the processing techniques described herein includes a processor 502, a volatile memory 504, a non-volatile memory 508 (e.g., hard disk) and the interface module 508 (e.g., a user interface, USB interface and so forth). The non-volatile memory 508 stores computer instructions 510, an operating system 512 and data 514. In one example, the computer instructions 510 are executed by the processor 502 out of volatile memory 504 to perform all or part of the processes described herein (e.g., processes 300, 400).

[0039] The processes described herein (e.g., processes 300, 400) are not limited to use with hardware and software of FIGs. 1-2 or 4; they may find applicability in any computing or processing environment and with any type of machine or set of machines that is capable of running a computer program. The processes described herein may be implemented in hardware, software, or a combination of the two. The processes described herein may be implemented in computer programs executed on programmable computers/machines that each includes a processor, a non-transitory machine-readable medium or another article of manufacture that is readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and one or more output devices. Program code may be applied to data entered using an input device to perform any of the processes described herein and to generate output information.

[0040] The system may be implemented, at least in part, via a computer program product, (e.g., in a non-transitory machine-readable storage medium such as, for example, a non-transitory computer-readable medium), for execution by, or to control the operation of, data processing apparatus (e.g., a programmable processor, a computer, or multiple computers). Each such program may be implemented in a high level procedural or object-oriented programming language to work with the rest of the computer-based system. However, the programs may be implemented in assembly, machine language, or Hardware Description Language. The language may be a compiled or an interpreted language, and it may be deployed in any form, including as a stand-alone program or as a module, component, subroutine, or another unit suitable for use in a computing environment. A computer program may be deployed to be executed on one computer or multiple computers at one site or distributed across multiple sites and interconnected by a communication network. A computer program may be stored on a non-transitory machine-readable medium that is readable by a general or special purpose programmable computer for configuring and operating the computer when the non-transitory machine-readable medium is read by the computer to perform the processes described herein. For example, the processes described herein may also be implemented as a non-transitory machine-readable storage medium, configured with a computer program, where upon execution, instructions in the computer program cause the computer to operate in accordance with the processes. A non-transitory machine-readable medium may include but is not limited to a hard drive, compact disc, flash memory, non-volatile memory, volatile memory, magnetic diskette and so forth but does not include a transitory signal *per se*.

[0041] Having described preferred embodiments, which serve to illustrate various concepts, structures and techniques, which are the subject of this patent, it will now become apparent that other embodiments incorporating these concepts, structures and techniques may be used. Accordingly, it is submitted that the scope of the patent should not be limited to the described embodiments but rather should be limited only by the spirit and scope of the following claims.

[0042] Accordingly, other embodiments are within the scope of the following claims.

CLAIMS

1. A method for secure booting of a system comprising a processor, a management engine (ME) and a secured storage medium, the method comprising:

loading and executing a firmware in the ME;

establishing, by the ME, a communication channel to a security device;

receiving, by the ME, an encrypted boot image from the security device;

decrypting, by the ME, the encrypted boot image;

storing, by the ME, the decrypted boot image in the secured storage medium; and

resetting the system using the decrypted boot image in the secured storage medium.

2. The method of claim 1, wherein the system comprises a System-on-a-Chip (SoC).

3. The method of claim 1, wherein the encrypted boot image is encrypted using a unique key that is generated by the ME.

4. The method of claim 3, wherein the unique key comprises a unique-per-processor key that is generated based upon unique key material of the processor, wherein the unique key material of the processor is accessible by the ME.

5. The method of claim 3, further comprising:

loading, by the ME, an unencrypted boot image;

encrypting, by the ME, the unencrypted boot image using the unique key; and

storing, by the ME, the encrypted boot image in the security device.

6. The method of claim 5, further comprising:

generating, by the ME, an Advanced Encryption Standard (AES) key;

encrypting, by the ME, the unencrypted boot image using the AES key;

encrypting, by the ME, the AES key; and

storing, by the ME, the encrypted boot image and the encrypted AES key in the security device.

7. The method of claim 1, wherein the security device comprises a Hardware Root of Trust (HRoT) device.

8. The method of claim 1, wherein the security device comprises a network server.

9. The method of claim 1, wherein the communication channel to the security device comprises an encrypted communication channel.

10. The method of claim 1, further comprising:
transferring data to and from the security device via the communication channel.

11. A system comprising:
a processor and memory;
a management engine (ME); and
a secured storage medium;
wherein the ME is configured to:
load and execute a firmware;
establish a communication channel to a security device;
receive an encrypted boot image from the security device;
decrypt the encrypted boot image; and
store the decrypted boot image in the secured storage medium,
wherein the system resets using the decrypted boot image in the secured storage medium.

12. The system of claim 11, wherein the ME is further configured to:
load an unencrypted boot image from the security device;
generate a unique-per-processor key;
encrypt the unencrypted boot image using the unique-per-processor key; and
store the encrypted boot image in the security device.

13. The system of claim 11, wherein the security device comprises a Hardware Root of Trust (HRoT) device.

14. The system of claim 11, wherein the security device comprises a network server.

15. The system of claim 11, wherein the system comprises a System-on-a-Chip (SoC).

16. The system of claim 11, wherein the communication channel to the security device comprises an encrypted communication channel.

17. The system of claim 11, wherein the ME is located in a Platform Controller Hub (PCH) of the system.

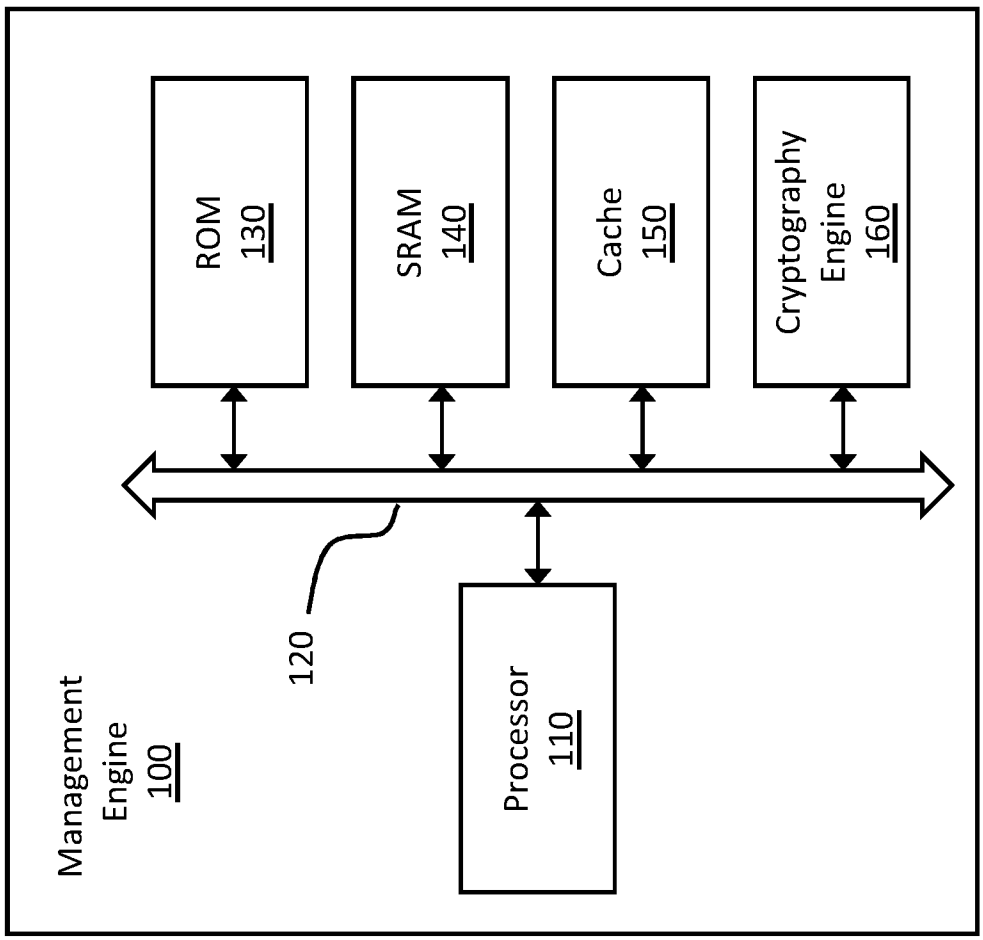

FIG. 1 (Prior Art)

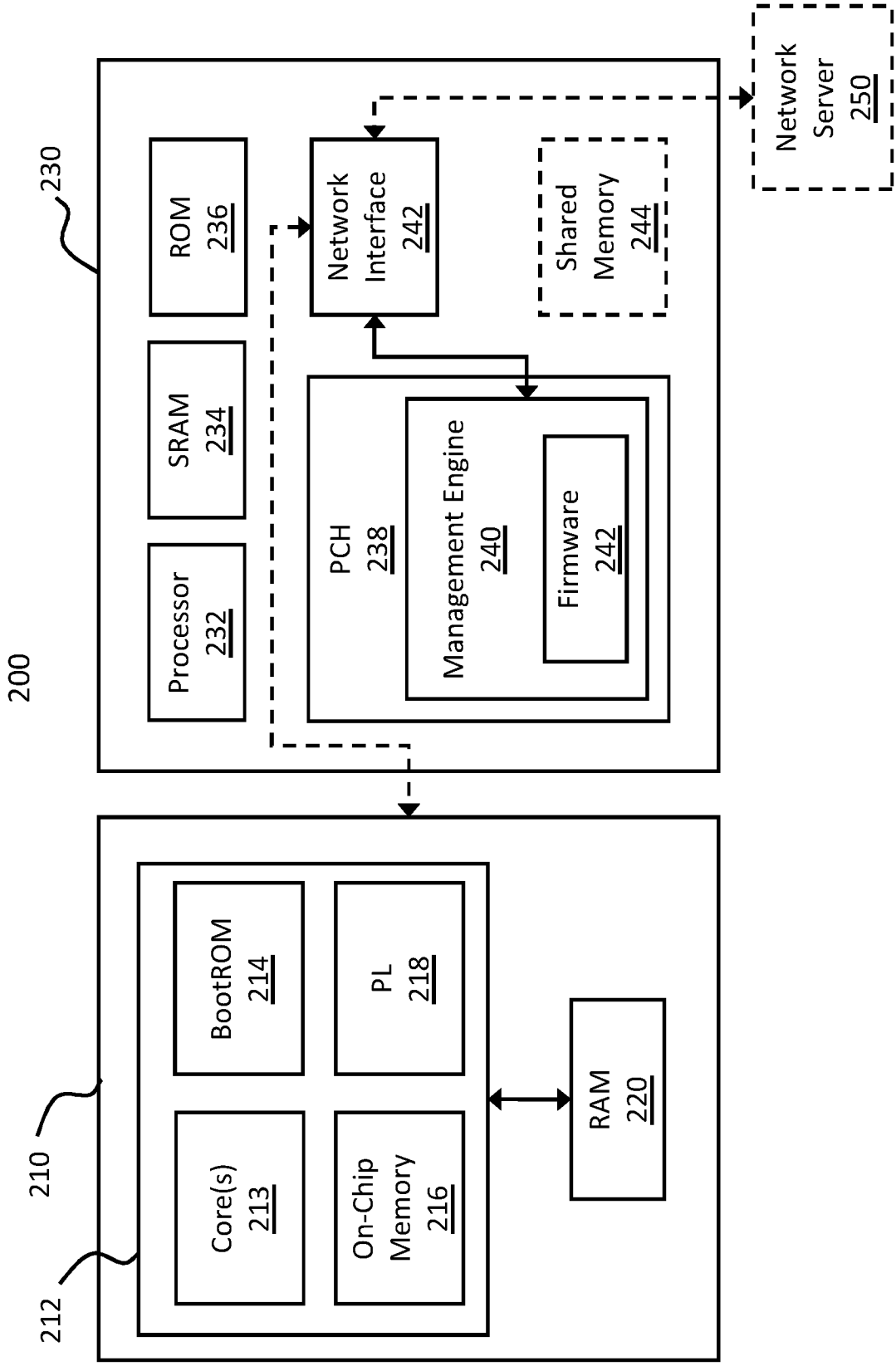
200
230
ROM 236
SRAM 234
Processor 232
Network Interface 242
Shared Memory 244
PCH 238
Management Engine 240
Firmware 242
Network Server 250
210
212
BootROM 214
PL 218
Core(s) 213
On-Chip Memory 216
RAM 220

FIG. 2

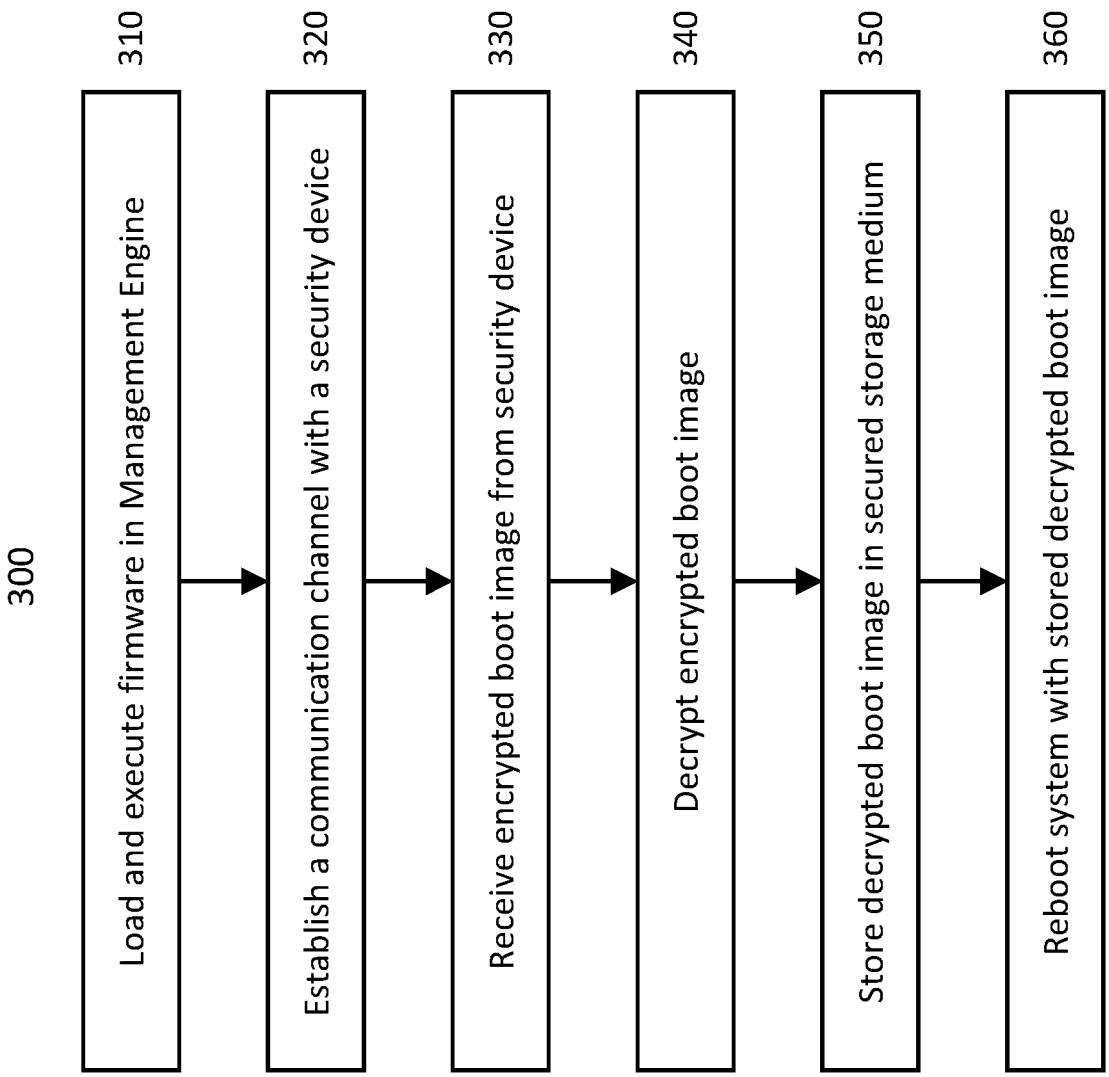
300
310
Load and execute firmware in Management Engine
320
Establish a communication channel with a security device
330
Receive encrypted boot image from security device
340
Decrypt encrypted boot image
350
Store decrypted boot image in secured storage medium
360
Reboot system with stored decrypted boot image

FIG. 3

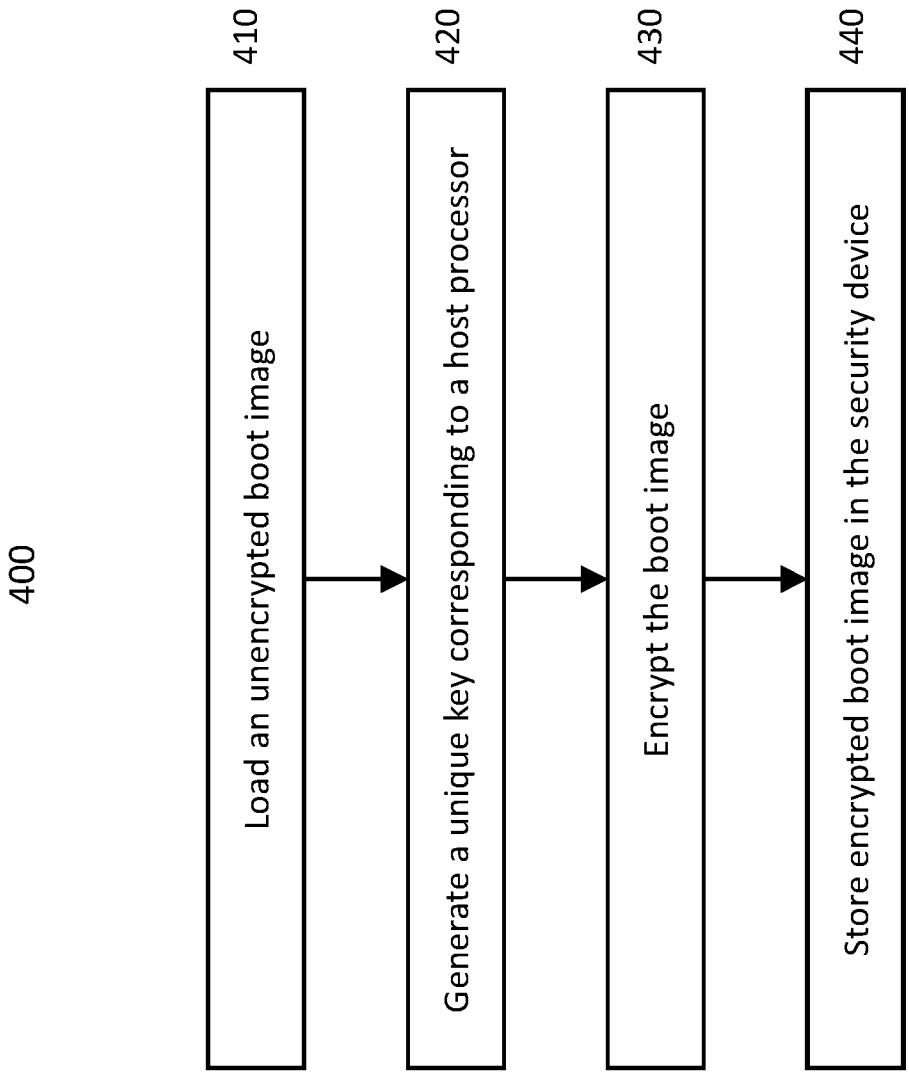
400
410
Load an unencrypted boot image
420
Generate a unique key corresponding to a host processor
430
Encrypt the boot image
440
Store encrypted boot image in the security device

FIG. 4

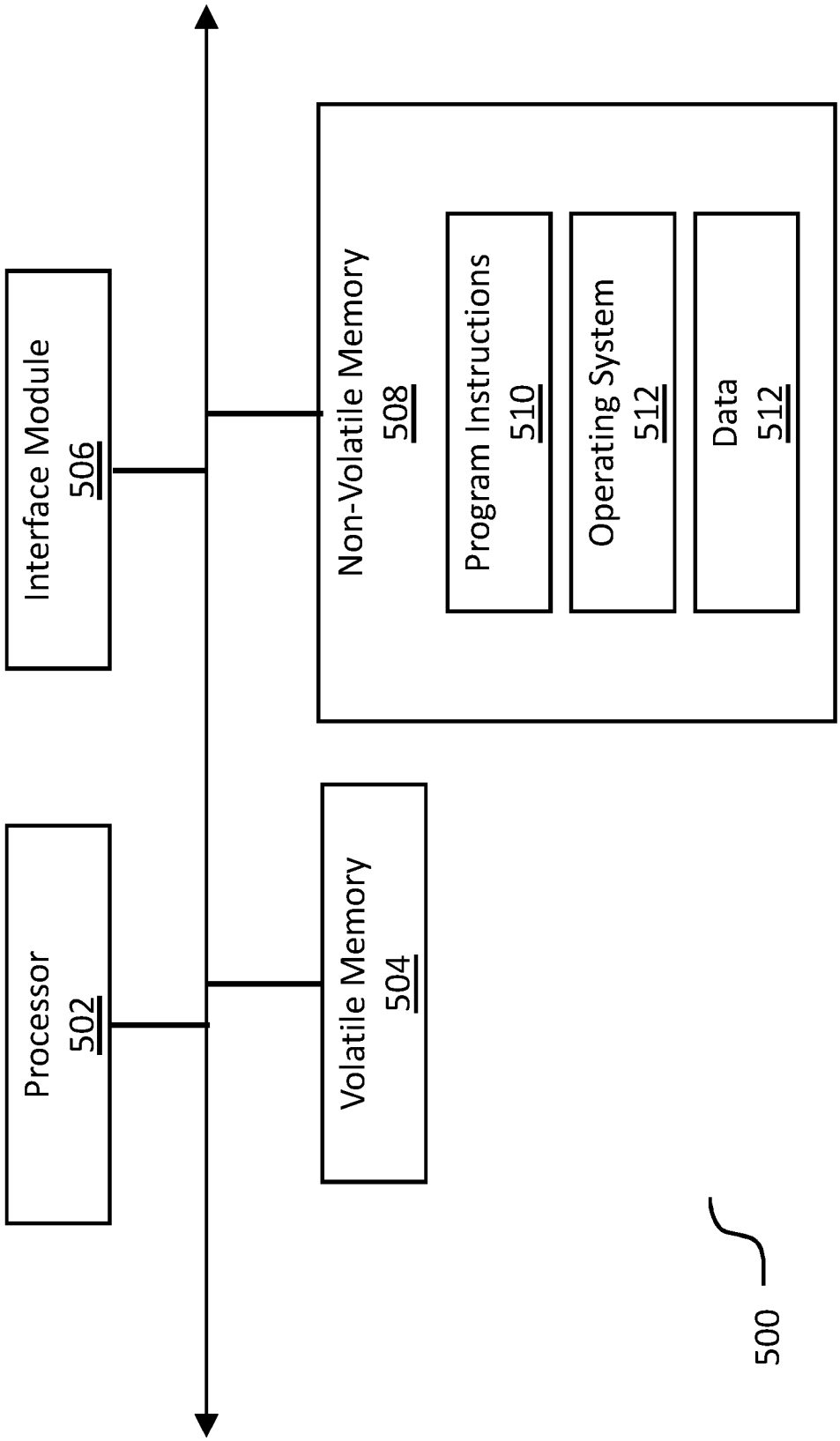
Interface Module
506
Processor
502
Volatile Memory
504
Non-Volatile Memory
508
Program Instructions
510
Operating System
512
Data
512
500

FIG. 5