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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
28 June 2012 (28.06.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/088029 A2

(51) International Patent Classification:
G06F 21/22 (2006.01) *G06F 9/22* (2006.01)

(21) International Application Number:
PCT/US2011/066014

(22) International Filing Date:
20 December 2011 (20.12.2011)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/978,457 24 December 2010 (24.12.2010) US

(71) Applicant *(for all designated States except US)*: **INTEL CORPORATION** [US/US]; 2200 Mission College Boulevard, MS: RNB-4-150, Santa Clara, California 95052 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **WOOD, Matthew D.** [US/US]; 4829 Nw 162nd Ter, Portland, Oregon 97229 (US). **SAINT-HILAIRE, Ylian** [US/US]; 1316 Ne Carlaby Way #173, Hillsboro, Oregon 97124 (US).

(74) Agents: **AGHEVLI, Ramin** et al.; Caven & Aghevli, c/o CPA GLOBAL, P.O. Box 52050, Minneapolis, Minnesota 55402 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, 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, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, 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, ML, MR, NE, SN, TD, TG).

Published:

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

(54) **Title**: SECURE APPLICATION ATTESTATION USING DYNAMIC MEASUREMENT KERNELS

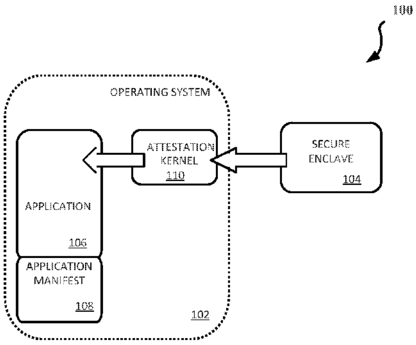

FIG. 1

(57) **Abstract**: Methods and apparatus to provide secure application attestation using dynamic measurement kernels are described. In some embodiments, secure application attestation is provided by using dynamic measurement kernels. In various embodiments, P-MAPS (Processor-Measured Application Protection Service), Secure Enclaves (SE), and/or combinations thereof may be used to provide dynamic measurement kernels to support secure application attestation. Other embodiments are also described.

SECURE APPLICATION ATTESTATION USING DYNAMIC MEASUREMENT KERNELS

FIELD

The present disclosure generally relates to the field of computing. More particularly, an embodiment of the invention generally relates to secure application attestation using dynamic measurement kernels.

BACKGROUND

As computer connectivity becomes more commonplace, securing computing devices from malicious entities, malware, etc. becomes a more challenging task. One way to increase security is to manage the privileged kernel of an operating system. As a result, ensuring the state of critical applications and being able to attest to their integrity to third parties may increase the security of the operating system as a whole.

Moreover, anti-virus software may be used for well-known types of attacks. However, such software is generally unable to address unknown threats or software that subverts the operating system and the services on which the anti-virus software depends.

BRIEF DESCRIPTION OF THE DRAWINGS

The detailed description is provided 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 figures indicates similar or identical items.

Figs. 1 and 3 illustrate embodiments of systems in accordance with some embodiments of the invention.

Figs. 2, 4A and 4B illustrate flow diagrams of methods, according to some embodiments of the invention.

Figs. 5 and 6 illustrate block diagrams of embodiments of computing systems, which may be utilized to implement some embodiments discussed herein.

DETAILED DESCRIPTION

In the following description, numerous specific details are set forth in order to provide a thorough understanding of various embodiments. However, various embodiments of the invention may be practiced without the specific details. In other instances, well-known methods, procedures, components, and circuits have not been described in detail so as not to obscure the particular embodiments of the invention. Further, various aspects of embodiments of the invention may be performed using various means, such as integrated semiconductor circuits ("hardware"), computer-readable instructions organized into one or more programs ("software"), or some combination of hardware and software. For the purposes of this disclosure reference to "logic" shall mean either hardware, software (including for example micro-code that controls the operations of a processor), or some combination thereof.

Generally, applications that handle sensitive data require the ability to protect the data from malicious entities, as well as a mechanism to prove to third parties that the applications remain unmodified and/or operating correctly. In some implementations, secure application attestation is provided by using dynamic measurement kernels. In various embodiments, P-MAPS (Processor-Measured Application Protection Service), Secure Enclaves (SE), and/or combinations thereof provide dynamic measurement kernels to support secure application attestation. In one or more embodiments, P-MAPS provides virtualization based container(s), while SE provides solutions based on support features present in hardware. In turn, applications with requirements for establishing trust with a third party (such as anti-virus software, context protection systems, etc.) may make use of one or more of the embodiments discussed herein.

More particularly, **Fig. 1** illustrates a block diagram of a system 100 to provide secure application attestation, in accordance with an embodiment. The system 100 illustrates a system utilizing SE (which is available from Intel® Corporation). However, the embodiments discussed herein are not limited to SE and other technologies having the same or similar components may be used. As shown in Fig. 1, the system 100 includes an Operating System (OS) 102 and a Secure Enclave (SE) 104. The OS 102 includes an application 106 (with an application manifest 108) and an attestation kernel 110.

Referring to **Fig. 2**, a flow diagram of a method 200 to provide secure application attestation is illustrated, in accordance with an embodiment. In an embodiment, one or more of the components discussed with reference to Fig. 1 may be used to perform one or more of the operations discussed with reference to method 200.

More particularly, SE allows one or more pieces of an application to be isolated from the rest of the process (and potentially the rest of the system). For example, when code is loaded into an enclave, the processor measures the content. This measurement is then used to attest to the state of the enclave. Also, the measurement may be used to recheck the content of the enclave at a later time to detect unexpected changes. In some embodiments, code within the enclave is allowed to access memory outside of the enclave, but code outside of the enclave is not allowed to access memory within the enclave.

Referring to Figs. 1-2, an application is allowed to generate an attestation of its state for verification by a third party using the method 200. At an operation 202, the application 106 receives an attestation request from a third party, e.g., including a random challenge nonce (CN) for freshness assurance and replay protection. At an operation 204, the application 106 loads (or causes loading of) an attestation kernel 110 into a storage unit such as an enclave (e.g., SE 104), also referred to as Attestation Enclave (AE).

At an operation 206, the application 106 executes (or through execution of the attestation kernel 110 causes execution of) the attestation-related operation(s) in the enclave (e.g., SE 104), e.g., passing a manifest signed by the application developer (or other trusted entity, such as an Information Technology (IT) department) and/or CN as parameter(s) in one or more embodiments.

At an operation 208, the AE (e.g., SE 104) generates an attestation of its own state – referred to as an Enclave Measurement (EM), e.g., which is cryptographically signed by the platform. At an operation 210, the AE verifies the authenticity of the manifest passed/generated at operation 206. At an operation 212, the AE uses the manifest contents to verify the state of the calling application by scanning memory, associated with the application, using the inside-out capabilities (i.e., where code within the enclave is allowed to access memory outside of the enclave, but code outside of the enclave is not allowed to access memory within the enclave).

At an operation 214, the AE generates a cryptographically signed statement – referred to as the Application Measurement (AM) – e.g., including a hash of the manifest and/or the nonce in one or more embodiments. At an operation 216, the AE returns the EM and AM to the application 106. The application sends the EM, AM, and manifest to the third party for verification at an operation 218.

Fig. 3 illustrates a block diagram of a system 300 to provide secure application attestation, in accordance with an embodiment. The system 300 illustrates a system utilizing a variant of the P-MAPS virtualization based container technology (which is available from Intel® Corporation). However, the embodiments discussed herein are not limited to P-MAPS and other technologies having the same or similar components may be used. As shown in Fig. 3, the system 300 includes the Operating System (OS) 102, application 106, application manifest 108, attestation kernel 110, secure VMM (Virtual Machine Manager) logic 302 (including attestation software 306), and Trusted eXecution Technology (TXT) logic (which is available from Intel® Corporation) 304. However, the embodiments discussed herein are not limited to TXT and other technologies having the same or similar components may be used.

Referring to **Fig. 4A**, a flow diagram of a method 400 to provide secure application attestation is illustrated, in accordance with an embodiment. In an embodiment, one or more of the components discussed with reference to Fig. 3 may be used to perform one or more of the operations discussed with reference to method 400.

More particularly, a P-MAPS container may be implemented using a relatively small VMM based on Intel® virtualization technologies (e.g., VT-x, VT-d, TXT, etc.) in some embodiments. The container may envelop an entire application, preventing software access to memory, even from the OS kernel. The P-MAPS VMM in turn verifies that an application matches a signed manifest at the time it constructs a container for the application. The application may request an attestation of itself from the VMM at runtime. In an embodiment, the P-MAPS VMM is modified to behave more closely to an SE technology that is based on hardware features. The SE hardware technology is capable of constructing a container around portions of an application; potentially constructing multiple independent containers within the same application.

Referring to Figs. 3-4A, an application is allowed to generate an attestation of its state for verification by a third party using the method 400. At an operation 402, the application 106 receives an attestation request from a third party, e.g., including a random CN for freshness assurance and replay protection. At an operation 404, the application 106 loads (or causes loading of) an attestation kernel (AK) 110 into a protected, attestable software container (AC), e.g., created by the secure VMM 302. In an embodiment, the VMM (e.g., secure VMM 302) checks the contents of the AK against a signed manifest at load time, e.g., using the attestation software 306.

At an operation 406, the application 106 executes (or through execution of the attestation kernel 110 causes execution of) the attestation-related operation(s) in the VMM and AK, e.g., passing a manifest signed by the application developer (or other trusted entity, such as an Information Technology (IT) department) and/or CN as parameter(s) in one or more embodiments.

At an operation 408, the AK requests an attestation of VMM's state from the VMM (e.g., secure VMM 302). In an embodiment, VMM uses a trusted hardware entity, such as a TPM (Trusted Platform Module) (which may also be used by the TXT 304), to provide quotes based on a secure measured launch of the VMM 302. The "quoted" attestation contains a measurement of the VMM's launch Measurement (VMMM) (which is cryptographically signed by the trusted hardware entity in an embodiment). In an embodiment, the trusted hardware entity (e.g., TPM) provides that measurement due to the measured launch of the VMM via TXT. At an operation 410, the VMM (e.g., secure VMM 302) rechecks/checks and/or issues/returns the measurement(s) of the AK-- referred to as Attestation Kernel Measurement (AKM), e.g., which the VMM cryptographically signs to provide an attestation of the VMM. In one embodiment, the VMM 302 uses the attestation software 306 to generate a quote of the AK 110 previously loaded in an AC.

At an operation 412, the AK 110 verifies the application manifest authenticity and uses the manifest contents to verify the state of the calling application by scanning its memory, e.g., using the inside-out capabilities (i.e., where code within the AC is allowed to access memory outside of the AC, but code outside of the AC is not allowed to access memory within the AC).

At an operation 414, the AK 110 generates a cryptographically signed statement – referred to as the Application Measurement (AM) – e.g., including a hash of the manifest and/or the nonce in one or more embodiments. At an operation 416, the AC returns the VMMM, AKM, and AM to the application 106. The application sends the VMMM, AKM, AM, and manifest to the third party for verification at an operation 418.

In one or more embodiments, in addition to making the operation of the P-MAPS container mechanism closer to that of the SE based mechanism, the embodiments discussed herein may provide additional benefits including the ability to load multiple isolated containers from different authors, as well as a performance boost for code not located within a container. Also, hardware protected code is used to measure and attest to unprotected code within the same process in some embodiments.

Fig. 4B illustrates a flow diagram of a method to provide secure application attestation, in accordance with an embodiment. In one embodiment, one or more of the components discussed with reference to Fig. 3 may be used to perform one or more of the operations discussed with reference to Fig. 4B.

As shown in Fig. 4B, a third party sends a request for attestation (including a CN), such as discussed with operation 402 of Fig. 4A. An application (such as application 106 of Fig. 3) then sends the request for attestation (including CN and a manifest such as discussed with reference to Fig. 4A) to an attestation kernel (such as the attestation kernel 110 of Fig. 3). The request is then forwarded to a VMM (such as the VMM 302 of Fig. 3). The VMM in turn utilizes a trusted hardware entity (such as TPM) to create a quote. The trusted hardware entity formats and signs the quote as VMMM. The generated quote (including VMMM) is forwarded to VMM. The VMM formats and signs an attestation kernel measurement as AKM and sends a response (including VMMM and AKM) to the attestation kernel. The attestation kernel verifies the manifest authenticity based on the response from VMM. The attestation kernel also verifies application according to the manifest. The attestation kernel formats and signs the attestation and CN as AM and sends a response to the application (including VMMM, AKM, and AM). In turn, the application responds to the third party with VMMM, AKM, and AM.

Fig. 5 illustrates a block diagram of an embodiment of a computing system 500. In various embodiments, one or more of the components of the system 500 may be provided in various electronic devices capable of performing one or more of the operations discussed herein with reference to some embodiments of the invention. For example, one or more of the components of the system 500 may be used to perform the operations discussed with reference to Figs. 1-4, e.g., by processing instructions, executing subroutines, etc. in accordance with the operations discussed herein. Also, various storage devices discussed herein (e.g., with reference to Figs. 5 and/or 6) may be used to store data, operation results, etc., including for example, the operating system 102 discussed with reference to Figs. 1-4. In one embodiment, one or more processors (or other hardware components) discussed with reference to Figs. 5-6 include one or more of the SE 104 of Fig. 1, secure VMM 302 of Fig. 3, and/or TXT 304 of Fig. 3.

More particularly, the computing system 500 may include one or more central processing unit(s) (CPUs) 502 or processors that communicate via an interconnection network (or bus) 504. Hence, various operations discussed herein may be performed by a CPU in some embodiments. Moreover, the processors 502 may include a general purpose processor, a network processor (that processes data communicated over a computer network 503), or other types of a processor (including a reduced instruction set computer (RISC) processor or a complex instruction set computer (CISC)). Moreover, the processors 502 may have a single or multiple core design. The processors 502 with a multiple core design may integrate different types of processor cores on the same integrated circuit (IC) die. Also, the processors 502 with a multiple core design may be implemented as symmetrical or asymmetrical multiprocessors. Moreover, the operations discussed with reference to Figs. 1-4 may be performed by one or more components of the system 500.

A chipset 506 may also communicate with the interconnection network 504. The chipset 506 may include a graphics and memory control hub (GMCH) 508. The GMCH 508 may include a memory controller 510 that communicates with a memory 512. The memory 512 may store data, including sequences of instructions that are executed by the CPU 502, or any other device included in the computing system 500. In an embodiment, the memory 512 may store an operating system 513, which may be the same or similar to the OS 102 of Figs. 1-4. Same or at least a portion of this data (including instructions) may be stored in disk drive 528 and/or one or more caches within processors 502. In one embodiment of the invention, the memory 512 may include one or more volatile storage (or memory) devices such as random access memory (RAM), dynamic RAM (DRAM), synchronous DRAM (SDRAM), static RAM (SRAM), or other types of storage devices. Nonvolatile memory may also be utilized such as a hard disk. Additional devices may communicate via the interconnection network 504, such as multiple CPUs and/or multiple system memories.

The GMCH 508 may also include a graphics interface 514 that communicates with a display 516. In one embodiment of the invention, the graphics interface 514 may communicate with the display 516 via an accelerated graphics port (AGP). In an embodiment of the invention, the display 516 may be a flat panel display that communicates with the graphics interface 514 through, for example, a signal converter that translates a digital representation of an image stored in a storage device such as video memory or system memory into display signals that are interpreted and displayed by the display 516. The display signals produced by the interface 514 may pass through various control devices before being interpreted by and subsequently displayed on the display 516. In some embodiments, the processors 502 and one or more other components (such as the memory controller 510, the graphics interface 514, the GMCH 508, the ICH 520, the peripheral bridge 524,. the chipset 506, etc.) may be provided on the same IC die.

A hub interface 518 may allow the GMCH 508 and an input/output control hub (ICH) 520 to communicate. The ICH 520 may provide an interface to I/O devices that communicate with the computing system 500. The ICH 520 may communicate with a bus 522 through a peripheral bridge (or controller) 524, such as a peripheral component interconnect (PCI) bridge, a universal serial bus (USB) controller, or other types of peripheral bridges or controllers. The bridge 524 may provide a data path between the CPU 502 and peripheral devices. Other types of topologies may be utilized. Also, multiple buses may communicate with the ICH 520, e.g., through multiple bridges or controllers. Moreover, other peripherals in communication with the ICH 520 may include, in various embodiments of the invention, integrated drive electronics (IDE) or small computer system interface (SCSI) hard drive(s), USB port(s), a keyboard, a mouse, parallel port(s), serial port(s), floppy disk drive(s), digital output support (e.g., digital video interface (DVI)), or other devices.

The bus 522 may communicate with an audio device 526, one or more disk drive(s) 528, and a network interface device 530, which may be in communication with the computer network 503. In an embodiment, the device 530 may be a NIC capable of wireless communication. Other devices may communicate via the bus 522. Also, various components (such as the network interface device 530) may communicate with the GMCH 508 in some embodiments of the invention. In addition, the processor 502, the GMCH 508, and/or the graphics interface 514 may be combined to form a single chip.

Furthermore, the computing system 500 may include volatile and/or nonvolatile memory (or storage). For example, nonvolatile memory may include one or more of the following: read-only memory (ROM), programmable ROM (PROM), erasable PROM (EPROM), electrically EPROM (EEPROM), a disk drive (e.g., 528), a floppy disk, a compact disk ROM (CD-ROM), a digital versatile disk (DVD), flash memory, a magneto-optical disk, or other types of nonvolatile machine-readable media that are capable of storing electronic data (e.g., including instructions). In an embodiment, components of the system 500 may be arranged in a point-to-point (PtP) configuration such as discussed with reference to Fig. 6. For example, processors, memory, and/or input/output devices may be interconnected by a number of point-to-point interfaces.

More specifically, **Fig. 6** illustrates a computing system 600 that is arranged in a point-to-point (PtP) configuration, according to an embodiment of the invention. In particular, Fig. 6 shows a system where processors, memory, and input/output devices are interconnected by a number of point-to-point interfaces. The operations discussed with reference to Figs. 1-5 may be performed by one or more components of the system 600.

As illustrated in Fig. 6, the system 600 may include several processors, of which only two, processors 602 and 604 are shown for clarity. The processors 602 and 604 may each include a local memory controller hub (MCH) 606 and 608 (which may be the same or similar to the GMCH 508 of Fig. 5 in some embodiments) to couple with memories 610 and 612. The memories 610 and/or 612 may store various data such as those discussed with reference to the memory 512 of Fig. 5.

The processors 602 and 604 may be any suitable processor such as those discussed with reference to the processors 602 of Fig. 6. The processors 602 and 604 may exchange data via a point-to-point (PtP) interface 614 using PtP interface circuits 616 and 618, respectively. The processors 602 and 604 may each exchange data with a chipset 620 via individual PtP interfaces 622 and 624 using point to point interface circuits 626, 628, 630, and 632. The chipset 620 may also exchange data with a high-performance graphics circuit 634 via a high-performance graphics interface 636, using a PtP interface circuit 637.

At least one embodiment of the invention may be provided by utilizing the processors 602 and 604. For example, the processors 602 and/or 604 may perform one or more of the operations of Figs. 1-5. Other embodiments of the invention, however, may exist in other circuits, logic units, or devices within the system 600 of Fig. 6. Furthermore, other embodiments of the invention may be distributed throughout several circuits, logic units, or devices illustrated in Fig. 6.

The chipset 620 may be coupled to a bus 640 using a PtP interface circuit 641. The bus 640 may have one or more devices coupled to it, such as a bus bridge 642 and I/O devices 643. Via a bus 644, the bus bridge 643 may be coupled to other devices such as a keyboard/mouse 645, the network interface device 630 discussed with reference to Fig. 6 (such as modems, network interface cards (NICs), or the like that may be coupled to the computer network 503), audio I/O device, and/or a data storage device 648. The data storage device 648 may store code 649 that may be executed by the processors 602 and/or 604.

In various embodiments of the invention, the operations discussed herein, e.g., with reference to Figs. 1-6, may be implemented as hardware (e.g., logic circuitry), software (including, for example, micro-code that controls the operations of a processor such as the processors discussed herein), firmware, or combinations thereof, which may be provided as a computer program product, e.g., including a tangible (e.g., non-transitory) machine-readable or computer-readable medium having stored thereon instructions (or software procedures) used to program a computer (e.g., a processor or other logic of a computing device) to perform an operation discussed herein. The machine-readable medium may include a storage device such as those discussed herein.

Reference in the specification to “one embodiment” or “an embodiment” means that a particular feature, structure, or characteristic described in connection with the embodiment may be included in at least an implementation. The appearances of the phrase “in one embodiment” in various places in the specification may or may not be all referring to the same embodiment.

Also, in the description and claims, the terms “coupled” and “connected,” along with their derivatives, may be used. In some embodiments of the invention, “connected” may be used to indicate that two or more elements are in direct physical or electrical contact with each other. “Coupled” may mean that two or more elements are in direct physical or electrical contact. However, “coupled” may also mean that two or more elements may not be in direct contact with each other, but may still cooperate or interact with each other.

Additionally, such computer-readable media may be downloaded as a computer program product, wherein the program may be transferred from a remote computer (e.g., a server) to a requesting computer (e.g., a client) by way of data signals, e.g., through a carrier wave or other propagation medium, via a communication link (e.g., a bus, a modem, or a network connection).

Thus, although embodiments of the invention have been described in language specific to structural features and/or methodological acts, it is to be understood that claimed subject matter may not be limited to the specific features or acts described. Rather, the specific features and acts are disclosed as sample forms of implementing the claimed subject matter.

CLAIMS

1. A method comprising:

   receiving an attestation request at an application;

   loading an attestation kernel into a storage unit in response to the attestation request;

   executing one or more operations, corresponding to the attestation request and in accordance with data stored in the storage unit, to generate a manifest;

   generating an attestation of data stored in the storage unit;

   verifying a state of the application based on the generated attestation of the data stored in the storage unit and the manifest;

   generating a statement of application measurement based on a hash of the manifest; and

   transmitting the application measurement and the attestation data to the application.

2. The method of claim 1, wherein the storage unit is one of an attestation enclave or an attestation container.

3. The method of claim 1, wherein verifying the state of the application is to comprise scanning memory associated with the application.

4. The method of claim 1, further comprising a virtual machine monitor checking the attestation of the data stored in the storage unit and issuing a measurement of the data stored in the storage unit, wherein the transmitting is to transmit the measurement of the data stored in the storage unit.

5. The method of claim 1, further comprising a virtual machine monitor checking the attestation of the data stored in the storage unit and issuing a measurement of the data stored in the storage unit, wherein the transmitting is to transmit the measurement of the data stored in the storage unit and a quote generated by a trusted hardware entity.

6. The method of claim 1, wherein verifying the state of the application is to comprise scanning memory associated with the application and wherein code within the storage unit is allowed to access memory outside of the storage unit, while code outside of the storage unit is prevented from accessing the data stored in the storage unit.

7. The method of claim 1, wherein executing the one or more operations is to be performed based on the attestation kernel.

8. The method of claim 1, wherein the manifest is comprise a random challenge nonce.

9. The method of claim 1, wherein the manifest is to be signed by a trusted entity.

10. The method of claim 1, further comprising cryptographically signing the attestation of the data stored in the storage unit.

11. The method of claim 1, further comprising transmitting the manifest, the application measurement, and the attestation data to a third party.

12. A computer-readable medium comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to:

receive an attestation request at an application;

load an attestation kernel into a storage unit in response to the attestation request;

execute one or more operations, corresponding to the attestation request and in accordance with data stored in the storage unit, to generate a manifest;

generate an attestation of data stored in the storage unit;

verify a state of the application based on the generated attestation of the data stored in the storage unit and the manifest;

generate a statement of application measurement based on a hash of the manifest; and

transmit the application measurement and the attestation data to the application.

13. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to scan memory associated with the application.

14. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to check, by a virtual machine monitor, the attestation of the data stored in the storage unit and to issue a measurement of the data stored in the storage unit.

15. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to check, by a virtual machine monitor, the attestation of the data stored in the

storage unit and to issue a measurement of the data stored in the storage unit.

16. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to scan memory associated with the application and wherein code within the storage unit is allowed to access memory outside of the storage unit, while code outside of the storage unit is prevented from accessing the data stored in the storage unit.

17. The computer-readable medium of claim 12, wherein the storage unit is one of an attestation enclave or an attestation container.

18. The computer-readable medium of claim 12, wherein the manifest is comprise a random challenge nonce.

19. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to cryptographically sign the attestation of the data stored in the storage unit.

20. The computer-readable medium of claim 12, further comprising one or more instructions that when executed on a processor configure the processor to perform one or more operations to transmit the manifest, the application measurement, and the attestation data to a third party.

21. A system comprising:

a memory to store one or more instructions corresponding to a container; and

a processor to execute the one or more instructions to:

receive an attestation request at an application;

load an attestation kernel into a storage unit in response to the attestation request;

execute one or more operations, corresponding to the attestation request and in accordance with data stored in the storage unit, to generate a manifest;

generate an attestation of data stored in the storage unit;

verify a state of the application based on the generated attestation of the data stored in the storage unit and the manifest;

generate a statement of application measurement based on a hash of the manifest; and

transmit the application measurement and the attestation data to the application.

22. The system of claim 21, wherein the storage unit is one of an attestation enclave or an attestation container.

23. The system of claim 21, further comprising a virtual machine monitor to check the attestation of the data stored in the storage unit and issue a measurement of the data stored in the storage unit.

24. The system of claim 21, further comprising a trusted entity to sign the manifest.

25. The system of claim 24, wherein the trusted entity is a trusted platform module.

26. The system of claim 21, further comprising logic to cryptographically sign the attestation of the data stored in the storage unit.

27. The system of claim 21, further comprising logic to transmit the manifest, the application measurement, and the attestation data to a third party.

28. The system of claim 21, wherein the manifest is comprise a random challenge nonce.

29. The system of claim 21, further comprising a virtual machine monitor to check the attestation of the data stored in the storage unit and issue a measurement of the data stored in the storage unit

30. The system of claim 29, further comprising logic to transmit the measurement of the data stored in the storage unit and a quote generated by a trusted hardware entity.

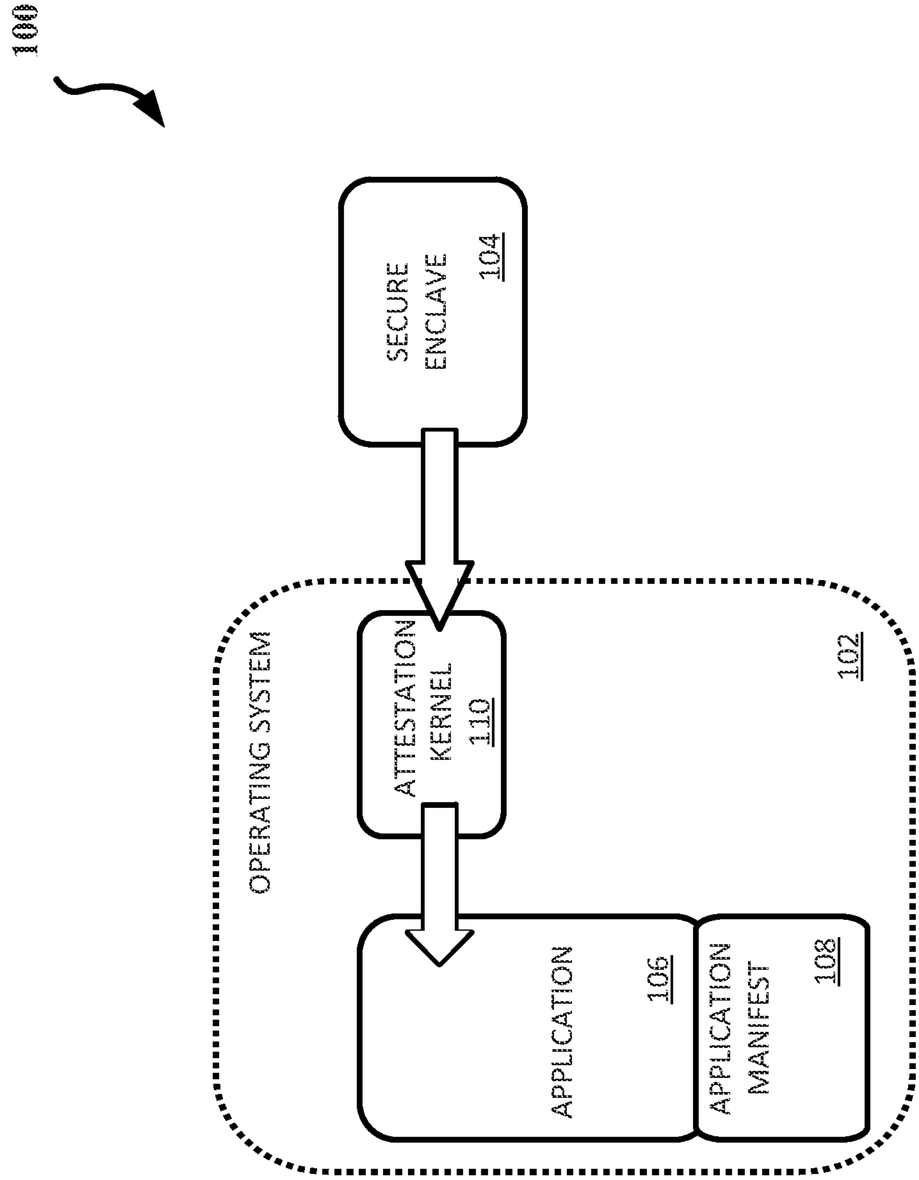
100
SECURE ENCLAVE 104
OPERATING SYSTEM
ATTESTATION KERNEL 110
102
APPLICATION 106
APPLICATION MANIFEST 108

FIG. 1

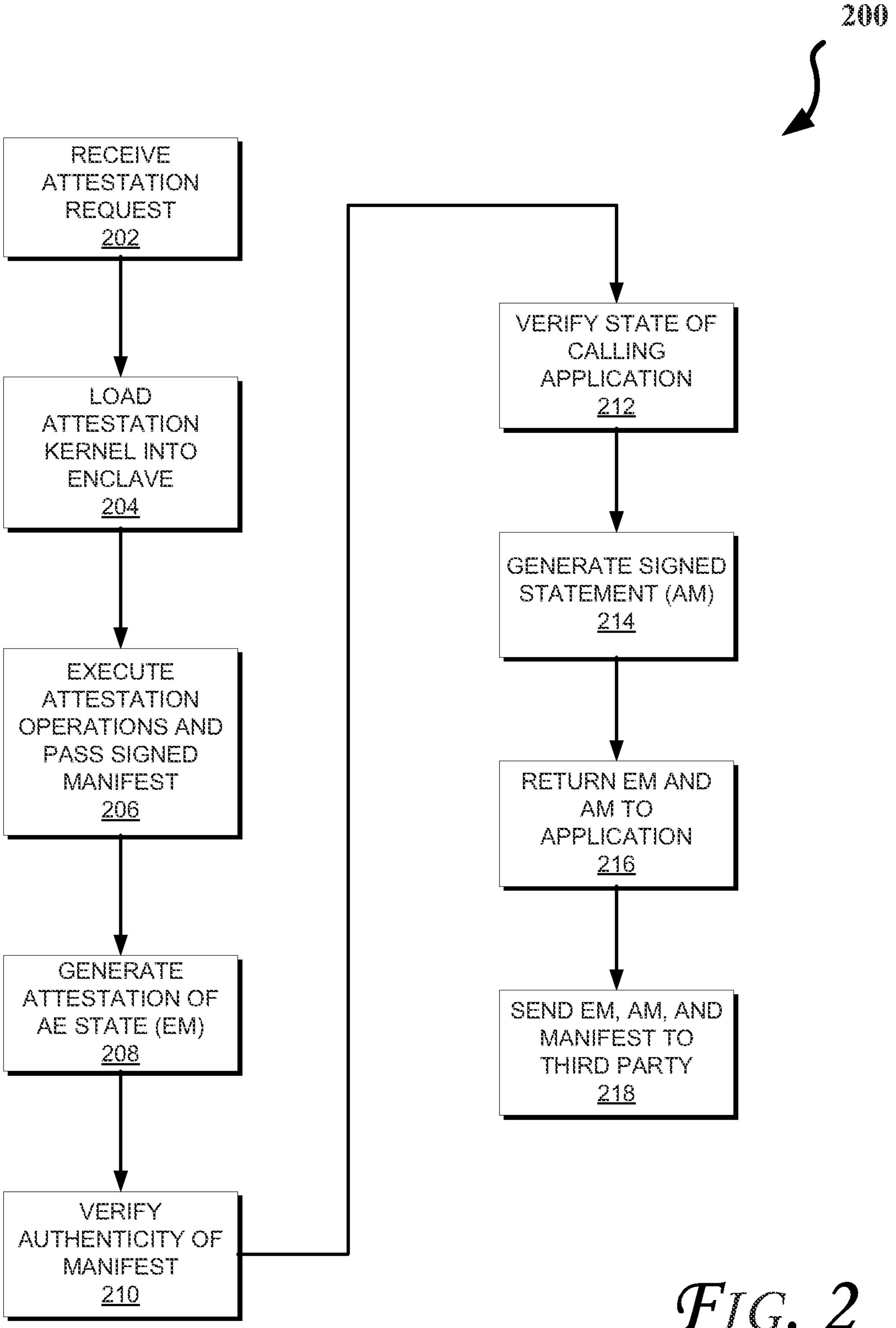
200
RECEIVE ATTESTATION REQUEST 202
LOAD ATTESTATION KERNEL INTO ENCLAVE 204
EXECUTE ATTESTATION OPERATIONS AND PASS SIGNED MANIFEST 206
GENERATE ATTESTATION OF AE STATE (EM) 208
VERIFY AUTHENTICITY OF MANIFEST 210
VERIFY STATE OF CALLING APPLICATION 212
GENERATE SIGNED STATEMENT (AM) 214
RETURN EM AND AM TO APPLICATION 216
SEND EM, AM, AND MANIFEST TO THIRD PARTY 218

FIG. 2

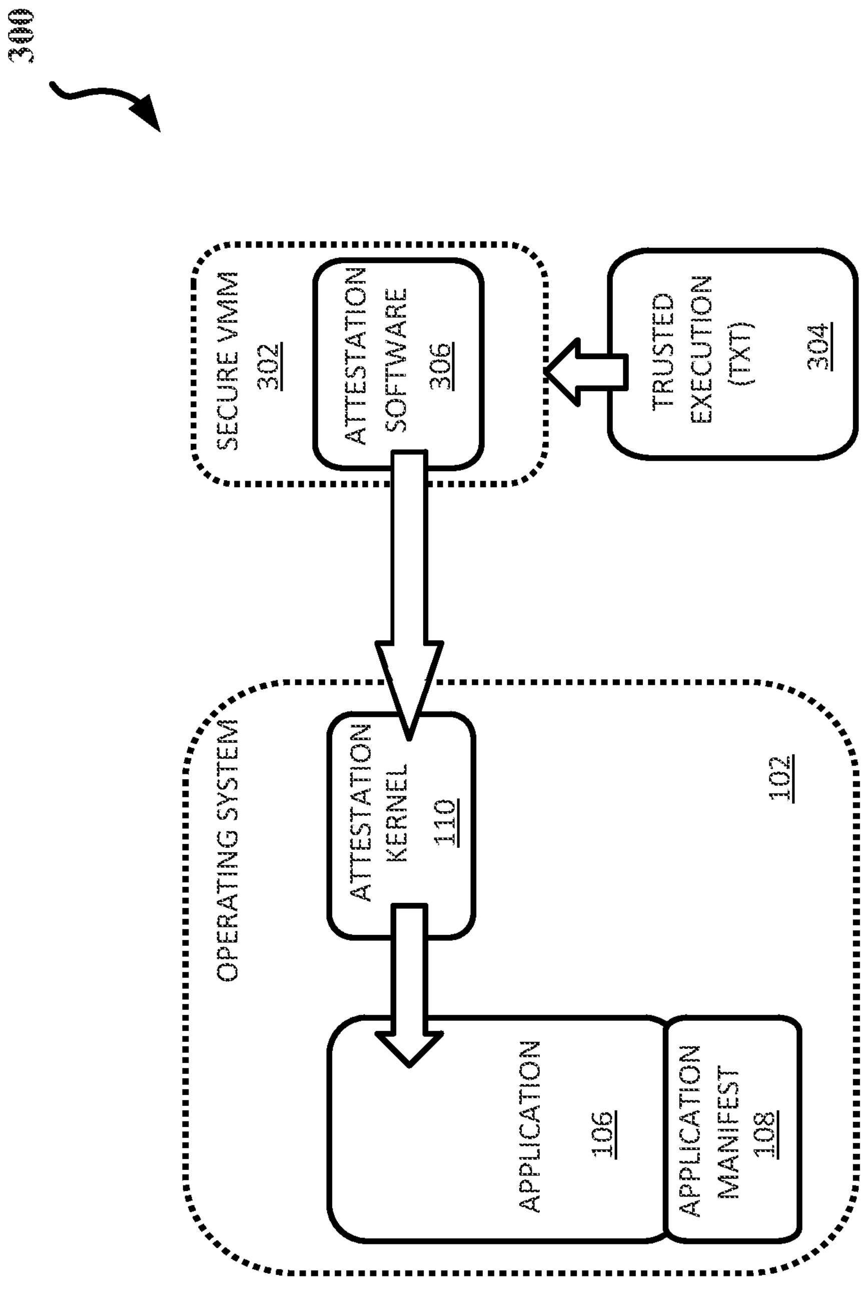
300
SECURE VMM
302
ATTESTATION SOFTWARE
306
TRUSTED EXECUTION (TXT)
304
OPERATING SYSTEM
102
ATTESTATION KERNEL
110
APPLICATION
106
APPLICATION MANIFEST
108

FIG. 3

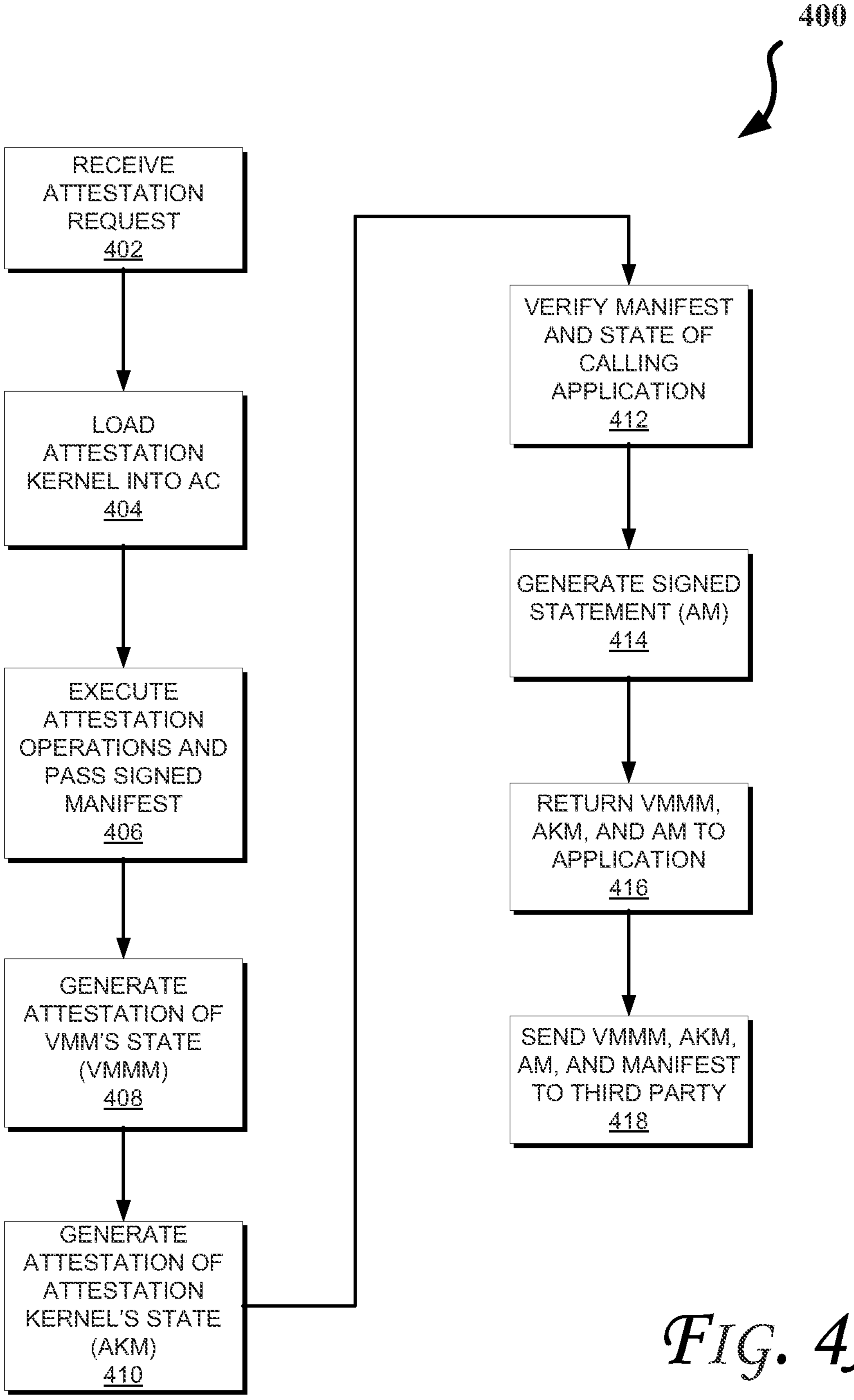
400
RECEIVE ATTESTATION REQUEST
402
LOAD ATTESTATION KERNEL INTO AC
404
EXECUTE ATTESTATION OPERATIONS AND PASS SIGNED MANIFEST
406
GENERATE ATTESTATION OF VMM'S STATE (VMMM)
408
GENERATE ATTESTATION OF ATTESTATION KERNEL'S STATE (AKM)
410
VERIFY MANIFEST AND STATE OF CALLING APPLICATION
412
GENERATE SIGNED STATEMENT (AM)
414
RETURN VMMM, AKM, AND AM TO APPLICATION
416
SEND VMMM, AKM, AM, AND MANIFEST TO THIRD PARTY
418

FIG. 4A

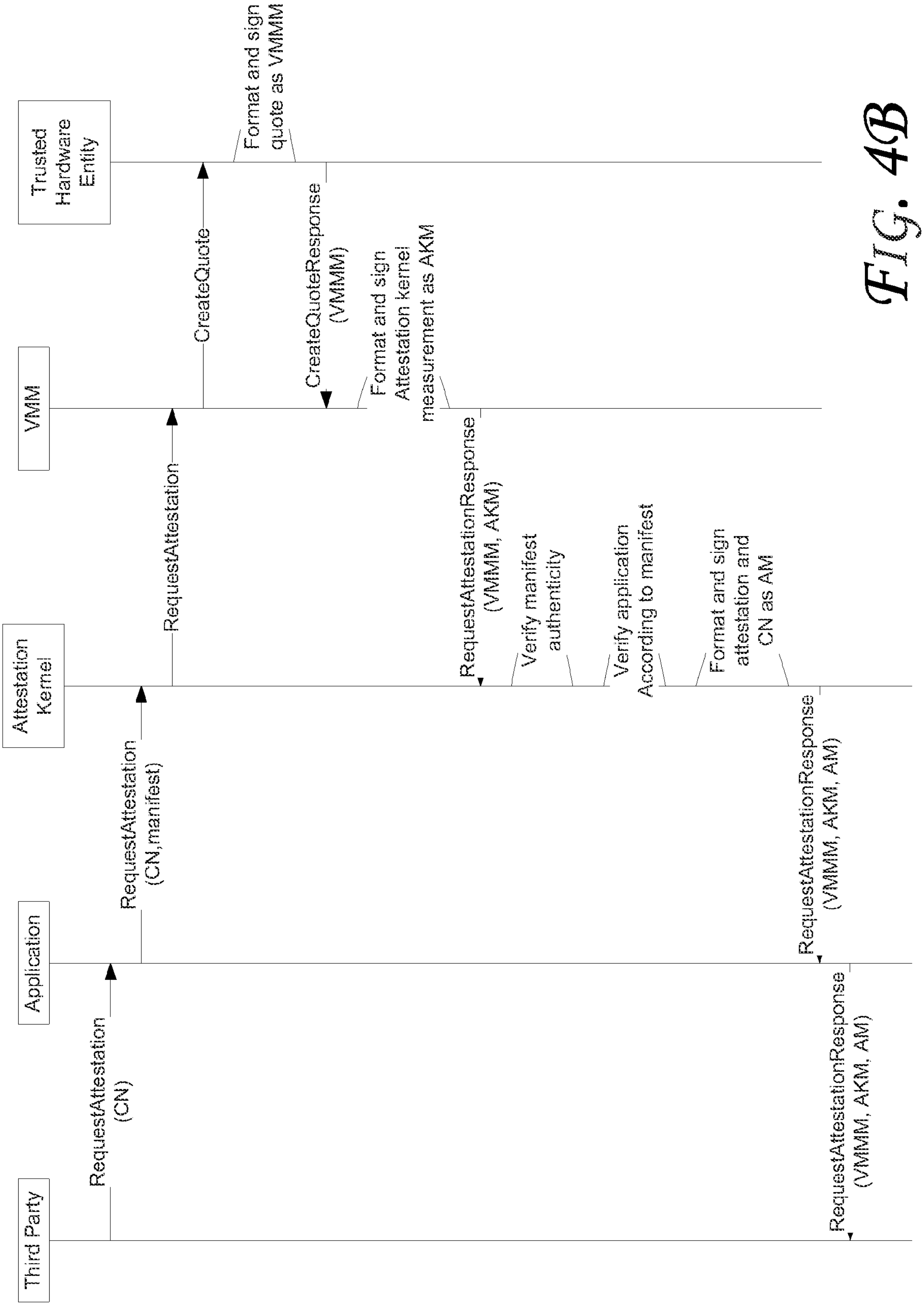
Third Party
Application
Attestation Kernel
VMM
Trusted Hardware Entity
RequestAttestation (CN)
RequestAttestation (CN,manifest)
RequestAttestation
CreateQuote
Format and sign quote as VMMM
CreateQuoteResponse (VMMM)
Format and sign Attestation kernel measurement as AKM
RequestAttestationResponse (VMMM, AKM)
Verify manifest authenticity
Verify application According to manifest
Format and sign attestation and CN as AM
RequestAttestationResponse (VMMM, AKM, AM)
RequestAttestationResponse (VMMM, AKM, AM)

FIG. 4B

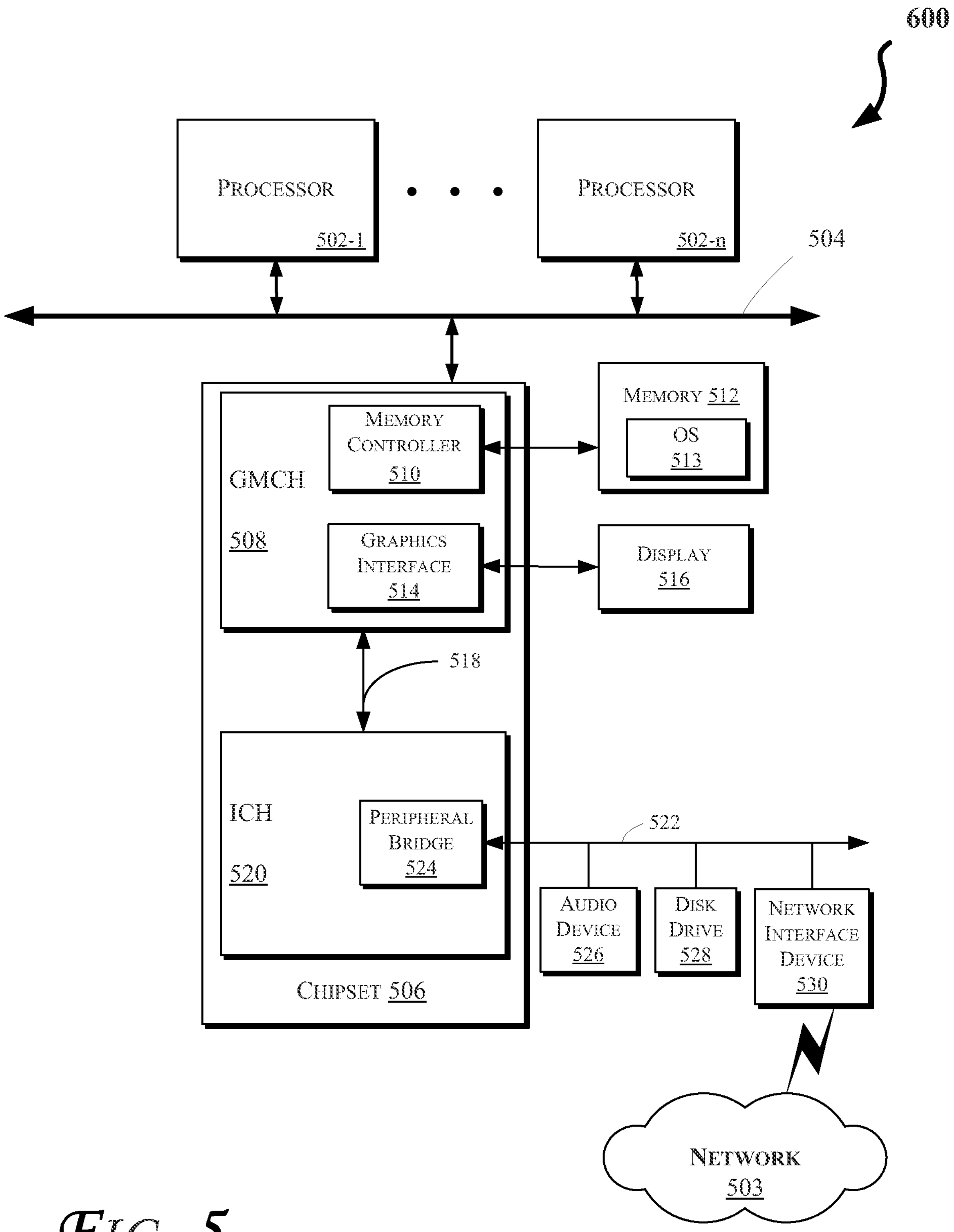
600
Processor
502-1
Processor
502-n
504
Memory Controller
510
GMCH
508
Graphics Interface
514
Memory 512
OS
513
Display
516
518
ICH
520
Peripheral Bridge
524
522
Audio Device
526
Disk Drive
528
Network Interface Device
530
Chipset 506
Network
503

FIG. 5

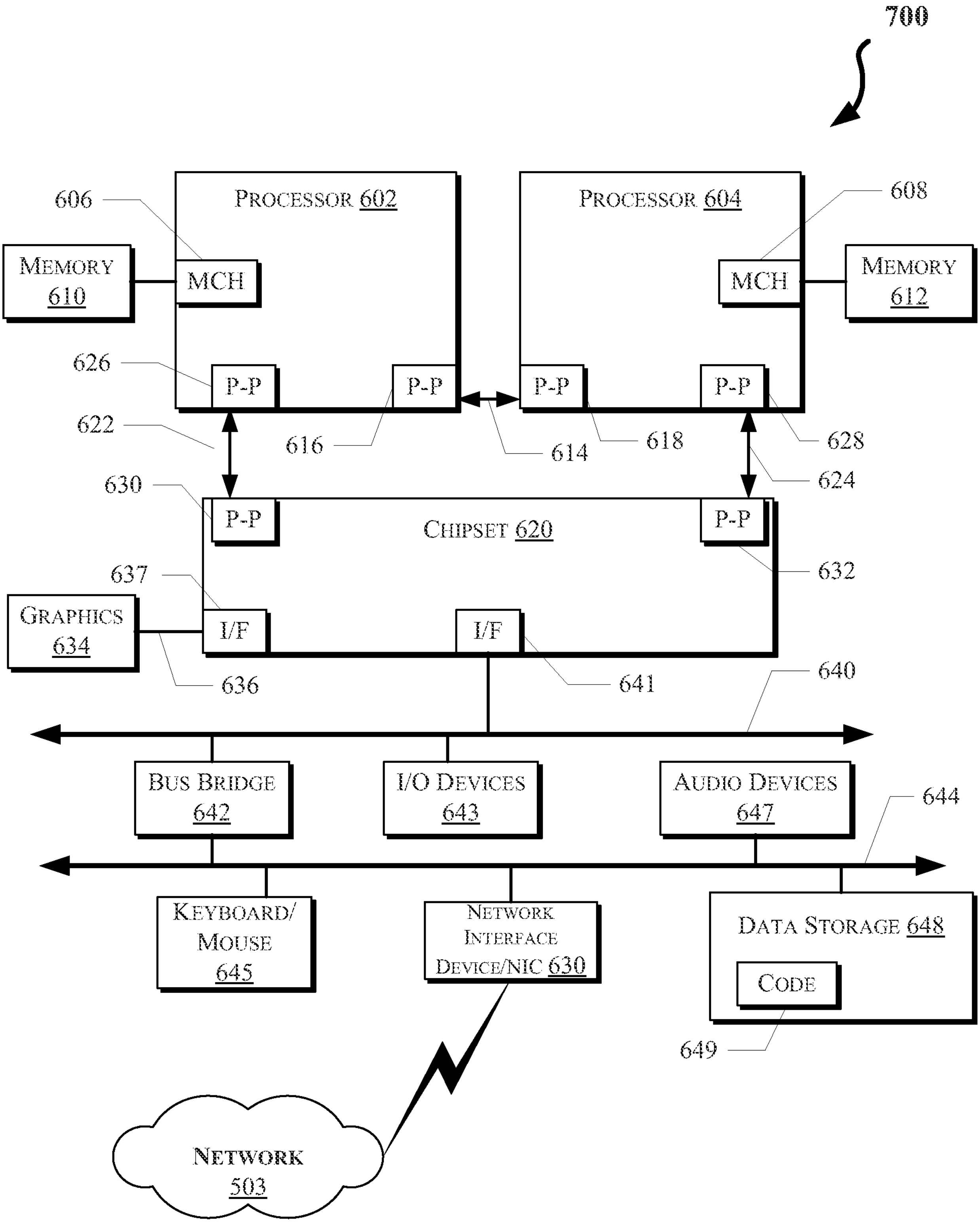
700
Processor 602
Processor 604
606
608
Memory 610
MCH
MCH
Memory 612
626
P-P
P-P
P-P
P-P
622
616
614
618
628
624
630
P-P
Chipset 620
P-P
632
637
Graphics 634
I/F
I/F
636
641
640
Bus Bridge 642
I/O Devices 643
Audio Devices 647
644
Keyboard/ Mouse 645
Network Interface Device/NIC 630
Data Storage 648
Code
649
Network 503

FIG. 6