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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 June 2019 (13.06.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/112972 A1

(51) International Patent Classification:
G06F 21/57 (2013.01)

(21) International Application Number:
PCT/US2018/063686

(22) International Filing Date:
03 December 2018 (03.12.2018)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

62/596,081	07 December 2017 (07.12.2017)	US
62/596,099	07 December 2017 (07.12.2017)	US
16/206,092	30 November 2018 (30.11.2018)	US

(71) Applicant: **APPLE INC.** [US/US]; One Apple Park Way, Cupertino, California 95014 (US).

(72) Inventors: **DE CESARE, Joshua P.**; One Apple Park Way, Cupertino, California 95014 (US). **PAASKE, Timothy R.**; One Apple Park Way, Cupertino, California 95014 (US). **KOVAH, Xeno S.**; One Apple Park Way, Cupertino, California 95014 (US). **SCHLEJ, Nikolaj**; One Apple Park Way, Cupertino, California 95014 (US). **WILCOX, Jeffrey R.**; One Apple Park Way, Cupertino, California 95014 (US). **DOSHI, Hardik K.**; One Apple Park Way, Cupertino, California 95014 (US). **ALDERFER, Kevin H.**; One Apple Park Way, Cupertino, California 95014 (US). **KALLENBERG, Corey T.**; One Apple Park Way, Cupertino, California 95014 (US).

(74) Agent: **MEYERTONS, HOOD, KIVLIN, KOWERT & GOETZEL, P.C.**; SEEGERS, Paul T., P.O. Box 398, Austin, Texas 78767-0398 (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,

(54) Title: METHOD AND APPARATUS FOR BOOT VARIABLE PROTECTION

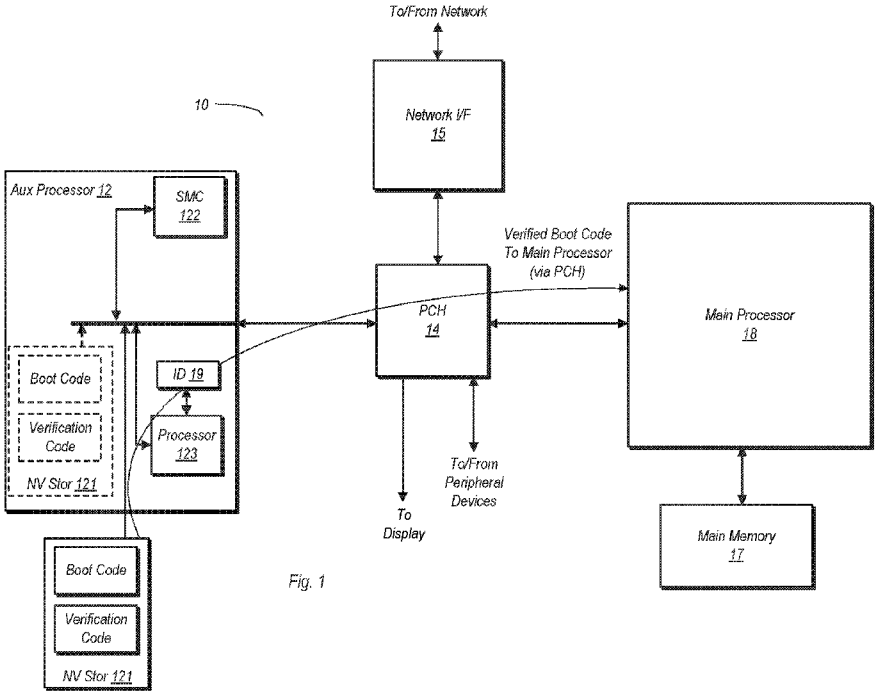

Fig. 1

(57) Abstract: A method and apparatus for protecting boot variables is disclosed. A computer system includes a main processor and an auxiliary processor. The auxiliary processor is associated with a non-volatile memory that stores variables associated with boot code that is also stored thereon. The main processor may send a request to the auxiliary processor to alter one of the variables stored in the non-volatile memory. Responsive to receiving the request, the auxiliary processor may execute a security policy to determine if the main processor meets the criteria for altering the variable. If the auxiliary processor determines that the main processor meets the criteria, it may grant permission to alter the variable.

WO 2019/112972 A1

[Continued on next page]

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, 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).

Published:

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

METHOD AND APPARATUS FOR BOOT VARIABLE PROTECTION

Technical Field

[0001] This disclosure is directed to computer systems, and more particularly, to security during the booting of computer systems.

Description of the Related Art

[0002] The beginning of operation in a computer system begins with a process known as boot, or "booting up." Firmware in a computer system may provide the first instructions that are executed by a processor in order to begin the boot process. From there, the processor may execute instructions to perform functions such as configuring I/O drivers, loading an operating system kernel, and so forth. Upon successful completion of the boot process, the computer system is ready for normal operation.

[0003] One of the times that a computer may be vulnerable to attacks by malicious software is during the boot process. Accordingly, many computer systems are designed with extra security measures to ensure that they are not compromised during the boot process. Despite these measures, an attacker may nevertheless gain access to the system, e.g., to an operating system kernel. Once the attacker has gained access, the firmware used to boot the system can be altered, variables used in booting the computer system can be altered, and information can be stolen. Thus, the computer itself may be compromised, as may one or more users of the system.

SUMMARY

[0004] A method and apparatus for protecting boot variables is disclosed. In one embodiment, computer system includes a main processor and an auxiliary processor. The auxiliary processor is associated with a non-volatile storage that stores variables associated with boot code that is also stored thereon. The main processor may send a request to the auxiliary processor to alter one of the variables stored in the non-volatile storage. Responsive to receiving the request, the auxiliary processor may execute a security policy to determine if the main processor meets the criteria for altering the variable. If the auxiliary processor determines that the main processor meets the criteria, it may grant permission to alter the variable.

[0005] In one embodiment, the criteria include determining if the main processor is operating in a recovery mode and if the main processor has provided proper credentials (e.g., a password or some other form of identification information). If the auxiliary processor determines that the main processor is not operating in the recovery mode, has not provided the proper credentials, or

both, then it may inhibit the main processor from altering the variable. Operating in the recovery mode comprises the main processor operating in a mode the computer system may operate with a reduced set of capabilities relative to a normal operating system.

[0006] During a system boot procedure, the main processor may request access (e.g., read access) to various ones of the variables stored in the non-volatile memory. The auxiliary processor may request that the main processor provide proper credentials, if not already provided. Responsive to determining that the main processor has provided the proper credentials, the auxiliary processor may allow the main processor access to the variable. The auxiliary processor may further limit access to variables stored on the non-volatile memory to one at a time.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The following detailed description refers to the accompanying drawings, which are now briefly described.

[0008] Fig. 1 is a block diagram of one embodiment of a computer system having main and auxiliary processors.

[0009] Fig. 2 is a simplified flow diagram illustrating a boot procedure in one embodiment of a computer system.

[0010] Fig. 3 is a flow diagram illustrating additional details of a boot procedure for one embodiment of a computer system.

[0011] Fig. 4 a simplified flow diagram illustrating another embodiment of a boot procedure for a computer system.

[0012] Fig. 5 is a flow diagram illustrating one embodiment of operations in a recovery operating system.

[0013] Fig. 6 is a block diagram illustrating information stored in one embodiment of a non-volatile memory implemented on an auxiliary processor.

[0014] Fig. 7A is a block diagram of additional details of one embodiment of an auxiliary processor.

[0015] Fig. 7B is a diagram illustrating information exchanges between various units in attempting to modify a variable in one embodiment.

[0016] Fig. 8 is a flow diagram of one embodiment of a method for determining whether a variable stored in non-volatile memory may be overwritten.

[0017] Fig. 9 is a flow diagram of one embodiment of a method for determining whether to grant access to a variable stored in a non-volatile memory during a system boot procedure.

[0018] Although the embodiments disclosed herein are susceptible to various modifications and alternative forms, specific embodiments are shown by way of example in the drawings and are described herein in detail. It should be understood, however, that drawings and detailed description thereto are not intended to limit the scope of the claims to the particular forms disclosed. On the contrary, this application is intended to cover all modifications, equivalents and alternatives falling within the spirit and scope of the disclosure of the present application as defined by the appended claims.

[0019] This disclosure includes references to "one embodiment," "a particular embodiment," "some embodiments," "various embodiments," or "an embodiment." The appearances of the phrases "in one embodiment," "in a particular embodiment," "in some embodiments," "in various embodiments," or "in an embodiment" do not necessarily refer to the same embodiment. Particular features, structures, or characteristics may be combined in any suitable manner consistent with this disclosure.

[0020] Within this disclosure, different entities (which may variously be referred to as "units," "circuits," other components, etc.) may be described or claimed as "configured" to perform one or more tasks or operations. This formulation—[entity] configured to [perform one or more tasks]—is used herein to refer to structure (i.e., something physical, such as an electronic circuit). More specifically, this formulation is used to indicate that this structure is arranged to perform the one or more tasks during operation. A structure can be said to be "configured to" perform some task even if the structure is not currently being operated. A "credit distribution circuit configured to distribute credits to a plurality of processor cores" is intended to cover, for example, an integrated circuit that has circuitry that performs this function during operation, even if the integrated circuit in question is not currently being used (e.g., a power supply is not connected to it). Thus, an entity described or recited as "configured to" perform some task refers to something physical, such as a device, circuit, memory storing program instructions executable to implement the task, etc. This phrase is not used herein to refer to something intangible.

[0021] The term "configured to" is not intended to mean "configurable to." An unprogrammed FPGA, for example, would not be considered to be "configured to" perform some specific function, although it may be "configurable to" perform that function after programming.

[0022] Reciting in the appended claims that a structure is "configured to" perform one or more tasks is expressly intended not to invoke 35 U.S.C. § 112(f) for that claim element. Accordingly, none of the claims in this application as filed are intended to be interpreted as having means-plus-function elements. Should Applicant wish to invoke Section 112(f) during prosecution, it will recite claim elements using the "means for" [performing a function] construct.

[0023] As used herein, the term "based on" is used to describe one or more factors that affect a determination. This term does not foreclose the possibility that additional factors may affect the determination. That is, a determination may be solely based on specified factors or based on the specified factors as well as other, unspecified factors. Consider the phrase "determine A based on B." This phrase specifies that B is a factor that is used to determine A or that affects the determination of A. This phrase does not foreclose that the determination of A may also be based on some other factor, such as C. This phrase is also intended to cover an embodiment in which A is determined based solely on B. As used herein, the phrase "based on" is synonymous with the phrase "based at least in part on."

[0024] As used herein, the phrase "in response to" describes one or more factors that trigger an effect. This phrase does not foreclose the possibility that additional factors may affect or otherwise trigger the effect. That is, an effect may be solely in response to those factors, or may be in response to the specified factors as well as other, unspecified factors. Consider the phrase "perform A in response to B." This phrase specifies that B is a factor that triggers the performance of A. This phrase does not foreclose that performing A may also be in response to some other factor, such as C. This phrase is also intended to cover an embodiment in which A is performed solely in response to B.

[0025] As used herein, the terms "first," "second," etc. are used as labels for nouns that they precede, and do not imply any type of ordering (e.g., spatial, temporal, logical, etc.), unless stated otherwise. For example, in a register file having eight registers, the terms "first register" and "second register" can be used to refer to any two of the eight registers, and not, for example, just logical registers 0 and 1.

[0026] When used in the claims, the term "or" is used as an inclusive or and not as an exclusive or. For example, the phrase "at least one of x, y, or z" means any one of x, y, and z, as well as any combination thereof.

[0027] In the following description, numerous specific details are set forth to provide a thorough understanding of the disclosed embodiments. One having ordinary skill in the art, however, should recognize that aspects of disclosed embodiments might be practiced without these specific

details. In some instances, well-known circuits, structures, signals, computer program instruction, and techniques have not been shown in detail to avoid obscuring the disclosed embodiments.

DETAILED DESCRIPTION

[0028] Turning now to Fig. 1, a block diagram of one embodiment of a computer system is shown. The embodiment of computer system 10 is exemplary and is not intended to be limiting. The various units shown here may be implemented as integrated circuits, or as parts thereof.

[0029] Computer system 10 in the embodiment shown includes a main processor 18 and an auxiliary processor 12. Main processor 18 may be one of a number of different types of processors used in various types of computer systems. In one embodiment, main processor 18 may be a general purpose processor designed to execute software of a variety of types. In some embodiments, main processor 18 may be a multi-core processor. Multi-core embodiments may be homogeneous (e.g., each core has substantially the same architecture) or heterogeneous (e.g., one or more of the cores may have a substantially different architecture relative to other cores). In heterogeneous multi-core embodiments, the various processor cores may be optimized for different goals. For example, one processor core may be optimized for maximum performance (e.g., in instructions executed per unit time), while another processor core may be optimized to for power efficiency (e.g., to minimize power consumed for various processing workloads).

[0030] In addition to one or more processor cores, main processor 18 may also include a memory controller that is coupled to main memory 17. The memory controller implemented on main processor 18 may interface with the cores and other functional circuit blocks implemented on the same integrated circuit die. Furthermore, other functional circuit blocks implemented elsewhere in computer system 10 may also interface with the memory controller implemented on main processor 18 for access to main memory 17.

[0031] Computer system 10 in the embodiment shown includes platform controller hub (PCH) 14 coupled to main processor 18. PCH 14 may include a number of different functional circuit blocks used to provide interfacing and communications with other portions of computer system 10. For example, PCH 14 in the embodiment shown is coupled to a network interface 15 (e.g., such as network interface chip or card) to provide communications between computer system 10 and other systems coupled to a network. PCH 14 may also include a display controller, a real-time clock circuit, and an I/O controller, among other functional circuit blocks. PCH 14 may facilitate an interface with various types of peripherals on corresponding bus types. Such bus types may include PCI/PCIe, USB, and other commonly known communications buses. It is

noted that embodiments are possible and contemplated wherein function of PCH 14 are incorporated into main processor 18, and thus the disclosure is not limited to embodiments in which these units are separate entities.

[0032] PCH 14 may also include power management circuitry. The power management circuitry may perform actions such as power and/or clock gating of idle functional circuit blocks, restoring power/clock to circuit blocks exiting a sleep state, adjusting operating voltages for varying operating conditions, and so forth.

[0033] Auxiliary processor 12 in the embodiment shown may perform a number of functions that support operations of computer system 10. In the simplified embodiment shown, auxiliary processor 12 includes a system management controller (SMC) 122 and a processor 123. Auxiliary processor 12 may include a number of other functional circuit blocks that are not shown here for the sake of simplicity. A more detailed embodiment of auxiliary processor 12 is discussed below. In the illustrated embodiment, auxiliary processor 12 is coupled to a non-volatile storage 121 that is external thereto (and may be removable from computer system 10 and transferred to another system). However, as shown by the dashed lines, alternate embodiments are possible and contemplated in which non-volatile storage 121 is implemented on auxiliary processor 12. In various embodiment, NV storage may be directly accessible by auxiliary storage 121, but not directly accessible by main processor 18 or any other agent within computer system 10.

[0034] Auxiliary processor 12 also includes a mechanism for storing a unique identifier, ID 19. In one embodiment, a bank of fuses may be used to store ID 19 as a unique combination of blown and unblown fuses. Embodiments that store ID 19 in a non-volatile storage circuit (e.g., read-only memory, or ROM), or other suitable mechanism are also possible and contemplated.

[0035] NV storage 121 in the embodiment shown may store information that may be used by computer system 10 to conduct various operations. As shown herein, NV storage 121 stores boot code in a binary format that may be executed by main processor 18 during a system boot (e.g., start-up) procedure. In one embodiment, the NV storage 121 is the sole source of boot code. NV storage 121 also stores verification code that may be executed by processor 123 to verify the boot code. Execution of the verification code may determine if the boot code is uncorrupted and/or has not been subject to tampering. This may include comparing a hash of the boot code to a known hash value, verifying the presence of a manufacturer's signature, and so on. Furthermore, the verification may include determining that the boot code is associated with an identifier that is unique to that particular computer system. Auxiliary processor 12 in the embodiment shown is

configured to access and release the boot code for transfer to main processor 18 (via PCH 14) after it has been successfully verified. In one embodiment, the boot code includes code conforming to the unified extensible firmware interface (UEFI) specification. However, the disclosure herein is not intended to be limiting to this specification, and thus embodiments in which the boot code conform to other specifications, formats, etc., are possible and contemplated.

[0036] NV storage 121 may also store variables used to set certain parameters and/or used by main processor 18 during the execution of the boot code. In some embodiments, NV storage 121 may store information to enable different operating systems to be utilized by computer system 10 according to inputs from a user during the boot procedure. In addition to information for boot services, runtime information may also store on NV storage 121. In general, NV storage 121 may store any information used to provide an interface between an operating system executing on main processor 18 and system firmware. Furthermore NV storage 121 may also store information used to ensure the security of computer system 10 against various types of attacks (e.g., malware, etc.).

[0037] NV storage 121 may be implemented using any suitable type of memory technology that enables contents stored thereon to remain even after power has been removed. In one embodiment, NV storage 121 may be implemented using flash memory. However, other types of memory (e.g., EPROM) may be used to implement NV storage 121 in other embodiments.

[0038] Processor 123 in the embodiment shown may perform various functions related to security of computer system 10. Among these functions is controlling access to and verifying the information stored in NV storage 121. For example, processor 123 may perform verification of the boot code to insure the integrity of its contents before it is release to the main processor for execution. Processor 123 may also operate in conjunction with a mailbox mechanism discussed in further detail below.

[0039] During a boot procedure, main processor 18 may request access (e.g., read access) to variables stored in NV storage 121. Processor 123 may control access to a given variable by determining if main processor 18 is authorized access thereto, and may also limit access to one variable at any given time (e.g., one per request). Similarly, if access to a variable is requested at any time during system operation by main processor 123 or another agent, processor 123 may control access thereto in a similar manner. Processor 123 may also may perform a comparison of a hash of the boot code to an expected value in order to verify the information is uncorrupted.

[0040] SMC 122 in the embodiment shown may perform various management functions pertaining to computer system 10. Such functions include causing the main processor 18 to

remain in a reset state upon system startup until such time that it is ready to receive verified boot code from auxiliary processor 12. Upon processor 123 completing successful verification of boot code during a system boot procedure, SMC 122 may provide an indication to PCH that the boot code is ready to be conveyed to main processor 18. Power management circuitry in PCH 14 may then release main processor 18 and send a request for the boot code to auxiliary processor 12. Responsive to the request, auxiliary processor 12 may provide the boot code to PCH 14, which may then relay it to main processor 18. Main processor 18 may then begin execution of the boot code to continue the boot procedure.

[0041] Fig. 2 is a simplified flow diagram illustrating a boot procedure in one embodiment of a computer system. Method 200 may be performed by various embodiments of the hardware disclosed herein. It is further contemplated that hardware embodiments not explicitly disclosed herein may also carry out method 200, and may thus fall within the scope of this disclosure.

[0042] As shown herein, method 200 illustrates various method tasks performed by an auxiliary processor and a main processor. Generally speaking, in the boot procedure discussed herein, an auxiliary processor may begin booting prior to the main processor. As part or at completion of the boot procedure for the auxiliary processor, boot code used by the main processor may be verified by circuitry within the auxiliary processor. Such verification may include ensuring that the boot code is uncorrupted, and has any necessary signatures (e.g., signed by the manufacturer). Upon successful verification, the boot code may be conveyed to the main processor for execution. Such a boot procedure may be referred to as a secure boot, as the verification of the boot code by secure circuitry within the auxiliary processor.

[0043] Method 200 begins with the access of a secure read-only memory (ROM) to begin the boot procedure (block 205). The secure ROM may store code that is executed by processing circuitry within the auxiliary processor in the booting. The secure ROM in various embodiments is located within the auxiliary processor. Furthermore, the secure ROM may be invisible to any component external to the auxiliary processor.

[0044] In block 210, an operating system (OS) used by the auxiliary processor is booted, and a system management controller (SMC) within the auxiliary processor may be started. The OS booted by the auxiliary system may be separate from that used by the main processor during normal system operations. Starting the SMC may include applying power thereto or otherwise removing it from a reset state.

[0045] Method 200 continues with the ongoing boot of the auxiliary OS, which includes loading a kernel and various drivers (block 215). The drivers may include any drivers used to operate

other components on the auxiliary processor, such as those that may be used to facilitate communications with external devices (e.g., a platform controller hub, main processor, etc.).

[0046] Upon completion of the auxiliary OS boot procedure in this particular embodiment, a verification of the boot code is performed (block 220). Verification of the boot code may generally ensure that the boot code has not been subject to tampering or inadvertent corruption. The verification may also include ensuring that the boot code is associated with an identifier that is unique to the computer system.

[0047] At this point method 200, the booting of the main processor may begin. The boot code may be conveyed by the auxiliary processor to the main processor (block 225), and a backlight of the computer system display may come on. Upon its receipt, the main processor may begin executing the boot code. Executing the boot code may include executing a boot loader routine for a main OS (block 230). As defined herein, the main OS may be that which is used by the computer system during normal operations, and through which a user interacts with the computer system. As the boot loader is executed, a manufacturer logo may appear on the system display.

[0048] Responsive to the execution of the boot loader, the main OS is invoked, with the main processor causing the activation of the system kernel and various system drivers (block 235). Upon completing the activation of the system kernel and drivers, the boot procedure may be complete, and the computer system may be ready for operations by a user (block 240).

[0049] Fig. 3 is a flow diagram illustrating additional details of a boot procedure for one embodiment of a computer system. Method 300 provides further illustration of the operations carried about by the auxiliary and main processors, as well as the SMC implemented on the auxiliary processor.

[0050] Responsive to initial power being applied to the computer system (e.g., the computer system is plugged into a wall outlet), a boot of the auxiliary processor along with verification of boot code for a main processor is performed (block 305). In conjunction with the boot of the auxiliary processor, SMC firmware may be initialized. Once the SMC firmware is running (block 320), it may provide an indication of the same to other circuitry within the auxiliary processor. When the SMC firmware is running, it may send a request to a power management unit (PMU) in the computer system to allow power to be applied to certain portions, such as a bus to enable communications between the auxiliary processor and other units within the system.

[0051] After completing the boot process, if no other external interaction with the computer system has taken place, the auxiliary processor may enter a wait state (block 310). Similarly, the

SMC may either remain running or may be placed into a power gated state while idle (block 330).

[0052] Responsive to user input (e.g., a user pressing a power button on the computer system), the boot procedure may continue. The user input may cause the SMC to exit the power gated state if it is not otherwise active. If the remaining circuitry in the auxiliary processor is in a wait state (block 335, yes), a wakeup signal may be sent thereto by the SMC. Responsive to receiving the wake signal, any circuitry within auxiliary processor that is in a wait state may be brought into a fully active state. After the wake signal has been sent to the other circuitry in the auxiliary processor, or the auxiliary processor was already in a fully awakened state (block 335, no), the SMC may send a signal to the power management unit to wake the PCH (block 345).

[0053] When the PCH is in the awakened state, the SMC may transmit a signal thereto (block 350) in order to indicate that the boot code has been verified and is ready to be conveyed to the main processor. The PCH may then release the main processor from the reset state (block 360). The auxiliary processor may then release the boot code for transfer (block 355), with the PCH responding by transferring the boot code to the main processor (block 365). Thereafter, the main processor may execute the boot code to load an OS and complete the boot procedure.

[0054] In some embodiments of a computer system according to this disclosure, booting of the main and auxiliary processor may occur concurrently. The may enable an overall faster boot procedure to occur. As part of such a boot procedure, a manufacturer logo may appear on a display of the computer system earlier than it would relative to the embodiment discussed in conjunction with Fig. 2. As a part of this boot procedure, verification of the main boot code (i.e. the boot code for the main processor) may occur before the auxiliary processor is fully booted. Fig. 4 is a simplified block diagram of one such embodiment.

[0055] Method 400 begins with processing circuitry within the auxiliary processor accessing secure ROM and beginning the boot procedure (block 405). As part of the execution of the code from the secure ROM, the SMC is started and verification of the boot code is performed (block 410). Upon successful completion of the verification process, the boot code may be forwarded to the main processor (block 425), and execution of the same may begin. Thus, at this point of the boot procedure, both the auxiliary and main processors are concurrently booting.

[0056] The auxiliary processor may complete its booting by loading the auxiliary OS kernel and various drivers (block 415), executing code from the secure ROM until the booting of the auxiliary OS is complete (block 420). The manufacturer's logo may appear on a display of the

computer system while the main processor is executing the boot code received thereby, and while the auxiliary processor is still executing code from the secure ROM.

[0057] The main processor may continue its boot procedure by invoking a boot loader to load the OS (block 430). Thereafter, the main OS kernel and drivers may be loaded and activated (block 435). Thereafter, the boot procedure may complete itself, and the computer system may be ready for operation by a user (block 440).

[0058] As previously noted, part of the boot code verification process in some embodiments of a computer system may include determining whether the boot code is associated with a unique identifier for the computer system. Associating the boot code with an identifier unique to a particular system (or particular chip in the system) may be known as "personalization." Through personalization, a particular instance of boot code (or other information) may be uniquely associated with a particular system such that the same code would be prevented from executing on another system. Boot code that is executable on a particular system based on a unique identifier may be said to be personalized to that particular system. It is further noted that some computer systems may be capable of loading different operating systems (e.g., based on a user selection at boot time). In such systems, personalization may be enabled for each of the different operating systems that are available for execution thereon.

[0059] In many computer systems, integrated circuits such as a main processor may have their own, unique identifier. The identifier may be implemented by, e.g., blowing fuses in the integrated circuit, storing the identifier in a portion of a non-volatile memory, or by other mechanisms. This unique identifier may also be embedded into files that also include the boot code. When the boot code is verified, the identifier may be checked along with the known identifier of the computer system to ensure there is a match. If there is no match, the boot may not proceed with the current boot code, and other mechanisms may be invoked. In such a case, the computer system may enter a recovery mode, or operate using a recovery OS. In the recovery mode/OS, the computer system may have a reduced set of capabilities relative to a normal OS through which normal operations are conducted (e.g., by a user, with full system capabilities activated and ready).

[0060] Fig. 5 is a flow diagram illustrating one embodiment of operations in a recovery operating system. More particularly, method 500 illustrates one embodiment of a methodology where a system may eventually be booted after an initial verification failure in which the identifier of the boot code did not match that of the system identifier.

[0061] Method 500 begins with the initiation of a boot procedure which includes a verification of the boot code (block 505). Additionally, the verification may include checking an identifier of the boot code against a unique identifier associate with the computer system (e.g., an identifier embedded in the main processor). If the verification does not fail (block 510, no), then the method may continue at block 530 with the continuation and eventual completion of the boot process.

[0062] If the verification fails (block 510, yes) e.g., because the identifier associated with the boot code does not match the unique system identifier, then the computer system may enter a recovery mode/OS and the manufacture may be contacted with a request to send a signed file (block 515). In sending the request, the unique identifier of the computer system may be included, along with other relevant information, such as type of computer, system configuration, and so on. The manufacturer may then generate a file or files that include the boot code. These files may be associated with the unique system identifier and may each be signed by the manufacturer. Upon completion of file generation, the manufacturer may send the files back to the requesting computer system (block 520).

[0063] Once the files have been received, they may be stored in the system. The newly received signed files may be added to storage while retaining any previously signed files (block 523). Thereafter, the boot procedure may be re-started by performing a re-verification with the newly received/signed files (block 525). Upon completing the re-verification, the boot process may continue to completion (block 530).

[0064] Fig. 6 is a block diagram illustrating information stored in one embodiment of a non-volatile memory implemented on an auxiliary processor. In the embodiment shown, NV storage 121 includes a payload 131, a manifest 132, and a set of variables 135. Payload 131 includes boot code 138, while manifest 132 includes a hash 139 that may be used as part of a verification of the boot code 138. In some embodiments, a single file may include at least the payload 131 and manifest 132, although these may be separate as shown here. Furthermore, multiple instances of these files may be present in some embodiments. For example, in a computer system capable of booting into multiple, different operating system, multiple instances of payload 131 (with corresponding, multiple instances of boot code 138) may be present. A boot procedure in such a system may include a user selecting which operating system to which the computer is to be booted. For embodiments which are capable of booting into multiple, different operating systems, personalization may be performed for each individual operating system.

[0065] During the verification process, circuitry in the auxiliary processor may create a hash from the boot code 138. This hash may be compared with the hash 139 in manifest 132. If the boot code has not been altered in any way, the hash created from the boot code should match hash 139. As with the files discussed above, multiple instances of hash 139 may be present to support personalization for corresponding ones of multiple boot code files and corresponding operating systems.

[0066] The manifest may also include other information, such as a signature from the manufacturer, identification information, and so on. In the embodiment shown, manifest 132 includes an ID hash 191, which may be used in checking a unique identifier (e.g., ID 19 on auxiliary processor 12 of Fig. 1) during certain operations (e.g., boot). Circuitry in the auxiliary processor may create a hash from ID 19, and compare that with ID hash 191 in order to perform the ID verification as part of the personalization discussed above. In addition to the boot code 138, payload 131 may also include other information, such as type and version information.

[0067] Variables 135 may include a wide variety of information. Variables may include system configuration variables regarding drivers to invoke, operational modes, security related variables, variables used in the boot process, and so forth.

[0068] Access to the variables stored in NV storage 121 may be tightly controlled by the auxiliary processor. Responsive to receiving a request for access to a variable stored in NV storage 121, auxiliary processor 12 may execute a security policy to determine whether variables may be overwritten and/or deleted and whether they may be accessed by the requesting agent, such as the main processor. Executing the security policy may include executing code to determine whether the requesting agent meets various criteria for performing the desired operations with the requested variable. The criteria may include credentials such as a password, an identifier or other information. Furthermore, as will be discussed below, embodiments are possible and contemplated wherein the auxiliary processor may control access to variables such that only one variable at a time may be accessed. The various mechanisms for protecting variables stored in NV storage 121 will now be discussed in further detail with reference to Fig. 7A, along with other aspects of an embodiment of an auxiliary processor.

[0069] Turning now to Fig. 7A, a block diagram of illustrating further details of auxiliary processor 12 is depicted. In the illustrated embodiment, auxiliary processor 12 includes at least one processor 123, ID 19, secure mailbox 145, and SMC 122. Within SMC 122 is included a doorbell mechanism 155 (hereinafter 'doorbell 155')—a memory used to store an indication signaling the presence of information in mailbox 145 such as requests for information or other

actions. In some embodiments, auxiliary processor 12 may include more (or less) components than shown in Fig. 7. For example, embodiments including cryptography circuitry are possible and contemplated. Through mechanisms such as the secure mailbox 145, auxiliary processor 12 implements a secure circuit that protects internal resources, e.g., such as NV storage 121 and the contents stored therein.

[0070] Secure mailbox 145, in one embodiment, includes an inbox and an outbox. As shown herein, secure mailbox 145 is coupled to processor 123 and a PCH (e.g. PCH 14 in Fig. 1). Both the inbox and the outbox may be first-in, first-out buffers (FIFOs) for data. The buffers may have any size (e.g. any number of entries, where each entry is capable of storing data from a read/write operation). Particularly, the inbox may be configured to store write data from write operations sourced from components external to auxiliary processor 12. The outbox may store write data from write operations sourced by processor 123 (As used herein, a "mailbox mechanism" refers to a memory circuit that temporarily stores 1) an input for a secure circuit until it can be retrieved by the circuit and/or 2) an output of a secure circuit until it can be retrieved by an external circuit).

[0071] In some embodiments, software executing on main processor 18 (or various hardware such as peripherals not otherwise shown) may request services of a component or components within auxiliary processor 12 via an application programming interface (API) supported by an operating system of computer system 10—i.e., a requester may make API calls that request services of some component within. These calls may cause an operating system executing on processor 18 to write corresponding requests to secure mailbox 145, which are then retrieved and analyzed by processor 123 to determine whether it should service the requests. By isolating the components within auxiliary processor 12 in this manner, overall security of the system may be enhanced.

[0072] SMC 122 in the embodiment shown may perform various ones of the functions as discussed above (in reference to Fig. 1). Additionally, as shown here, the illustrated embodiment includes doorbell 155. Doorbell 155 may be used by, e.g., components external to SMC 122 (on or off of auxiliary processor 12) to indicate that certain operations are requested. For example (and as discussed in further detail below), if a main processor wishes to modify a variable stored in the NV storage 121 coupled to auxiliary processor 12, it may cause the variable (or associated information) to be deposited in secure mailbox 145 while also sending an indication to doorbell 155 ("ringing" the doorbell). SMC 122 may then, for example, notify processor 123, that a

request for modification of a variable is present, with corresponding information deposited in secure mailbox 145.

[0073] Processor 123, in one embodiment, is configured to process commands received from various sources in computing device 10 (e.g. from main processor 18) and may use various secure peripherals to accomplish the commands. In various embodiments, processor 123 may execute securely loaded software that facilitates implementing functionality descried with respect to auxiliary processor 12. This software may include encrypted program instructions loaded from, e.g., a trusted zone in NV storage 121. Furthermore, processor 123 may limit access to variables used by main processor 18 during a boot procedure. For example, in one embodiment, during the boot procedure, processor 123 may limit access to variables from NV storage 121 to one variable at a time. Thus, main processor 18 may be required to send separate requests for each variable it wishes to access.

[0074] Control of access to variables may also require that a requesting agent meet certain criteria. For example, main processor 18 may be required to provide credentials such as the unique identifier discussed above, a password, or other information that indicates that it is authorized to access the variables. Furthermore, any agent that wishes to alter (e.g., overwrite or change) or delete a variable may also be required to provide credentials indicating they are authorized to perform such actions.

[0075] Processor 123 may also perform functions to verify boot code used by main processor 18 during the boot procedure. For example, referring back to Fig. 6, processor 123 may perform a comparison of hash 139 stored as part of manifest 132 to a hash of boot code 138. If the hashes match, processor 123 may then allow the boot code to be released and provided to main processor 18. Otherwise, the boot code is not released, and alternate procedures may be taken to boot computer system 10 (e.g., obtaining files signed and verified by the manufacturer).

[0076] Fig. 7B is a diagram illustrating information exchanges between various units in attempting to modify a variable in one embodiment. The discussion of method 700 below is in the context of the hardware embodiments discussed above with reference to Figs. 1 and 7A. However, it is noted method 700 may be carried out on other hardware embodiments not explicitly discussed herein.

[0077] Method 700 begins with the execution of a variable API (application programming interface) call executed on the main processor of a computer system. Responsive to the execution of the variable API call, the doorbell in the SMC is rung, while the variable (or an indication of which variable is to be modified) is forwarded to the mailbox. Responsive to the ringing of the

doorbell mechanism in the SMC, the SMC notifies the processor core of the auxiliary processor (e.g., processor 123 of Fig. 7A) to process a storage command for the NV storage. When this command is processed, the auxiliary processor retrieves variable data from the NV storage and deposits it into the mailbox.

[0078] In addition to depositing information regarding the variable into the mailbox, the auxiliary processor also executes a security policy related to the variable. In particular, execution of the security policy is used to determine if the requesting agent (in this case, the main processor) has authorization to modify the variable. If executing the security policy indicates that the main processor is not authorized, a fail message is deposited in the mailbox. If the main processor is authorized according to the security policy, the auxiliary processor may then cause the variable to be modified. Upon completion, the modified variable is then saved to NV storage. Additionally, an indication that the variable was successfully modified is deposited in the mailbox.

[0079] The main processor may receive an indication regarding whether the modification of the variable was successful or not from the mailbox. In particular, the main processor may poll the mailbox, with the mailbox returning the status to the main processor.

[0080] Fig. 8 is a flow diagram of one embodiment of a method for determining whether a variable stored in non-volatile memory may be overwritten. Method 800 may be performed by the auxiliary processor discussed above using processor 123 and secure mailbox 145. However, other embodiments of an auxiliary processor may also be able to perform the method shown herein, and thus may fall within the scope of this disclosure.

[0081] It is noted that while method 800 is directed to a main processor sending requests, the broader methodology contemplated herein may allow for other agents sending the same types of requests. Thus, the use of the main processor in the illustrated embodiment should be considered exemplary.

[0082] Method 800 begins with the main processor sending a request to the auxiliary processor to alter a variable stored in NV memory (block 805). Altering the variable may comprise deleting the variable, changing the value of a variable, or overwriting the variable. In sending the request, the main processor may include an address along with information indicative of the nature of the request. As discussed above, if the address is not directed to the inbox portion of the secure mailbox, the request may be denied without any traffic entering the auxiliary processor.

[0083] An auxiliary processor (e.g., processor 123 of Fig. 7A) may determine if the main processor meets the criteria for the request, e.g., by executing a security policy. For example, the

processor 123 may, through execution of the security policy, determine if the main processor has provided an identifier, a password, or other information that would indicate authorization to alter the variable. Furthermore, in some embodiments, the altering of at least some variables may be limited to certain modes of operation, such as a recovery mode. Thus, the mode of operation may also be criteria considered for the altering of variables.

[0084] If the main processor does not meet the criteria for altering the variables (block 810, no), alterations of the variable are made (block 820). In some embodiments, the auxiliary processor may cause an indication to be sent to the main processor that its request has been denied. However, embodiments in which no indication of the request denial is sent are also possible and contemplated.

[0085] If it is determined that all criteria have been met for altering the variable (block 810, yes), then the altering may be allowed to proceed (block 815). If the value of the variable is to be changed or overwritten, the secure processor may alter the variable and complete the operation by writing the desired information to the NV storage. If the variable is to be deleted, the auxiliary processor may carry out the actual deletion by removing the variable from NV storage.

[0086] Fig. 9 is a flow diagram of one embodiment of a method for determining whether to grant access to a variable stored in a non-volatile memory during a system boot procedure. It is noted that while the exemplary embodiment is directed to a boot procedure, portions of method 900 may be performed outside of the boot procedure, and thus are contemplated as alternate embodiments. Furthermore, while method 900 contemplates access requests made by a main processor, embodiments are possible and contemplated in which other agents request access to variables stored in an NV memory of an auxiliary processor.

[0087] Method 900 begins with the performing of a boot procedure, including the verification of main processor boot code by the auxiliary processor (block 905). In one embodiment, the boot code may be accessed solely from the NV storage associated with (e.g., coupled to) the auxiliary processor. Upon completing the verification, the main boot code may be conveyed to the main processor, where it may be executed to begin the main processor portion of the boot procedure (block 910). During the boot procedure, the main processor may send a request to the auxiliary processor for access (e.g., read access) to a variable stored within the NV storage (block 915). Responsive to the request, a secure processor or other circuitry within the auxiliary processor may execute a security policy to determine if the main processor is authorized access to the variable.

[0088] If it is determined that the main processor is authorized access to the variable (block 920, yes), the auxiliary processor may grant the request and allow access by the main processor (block 925). Access to the variable may depend on the nature of the request. For example, if the request is a read request, a secure processor may read the variable from the NV storage and deposit it in an outbox portion of the mailbox. Thereafter, the variable may be forwarded to the main processor.

[0089] If more variables are to be accessed (block 930, yes), the method may return to block 915 and the cycle may repeat as necessary. If no more variables are to be access (block 930, no), then the boot procedure may continue running to completion (block 935).

[0090] In the event that it is determined that the main processor is not authorized to access the variable (block 920, no), the boot procedure may be discontinued and the computer system may initiate recovery procedures (block 940). For example, the computer system may be re-directed to enter a recovery mode responsive to the denial of a request to a variable during the boot procedure.

[0091] Numerous variations and modifications will become apparent to those skilled in the art once the above disclosure is fully appreciated. It is intended that the following claims be interpreted to embrace all such variations and modifications.

WHAT IS CLAIMED IS:

1. A method, comprising:

a first processor receiving a request from a second processor to alter a first variable associated with boot code of the second processor, wherein the receiving includes the second processor storing the request in a mailbox circuit and sending a separate indication of the storing to the first processor;

in response to the indication and the request, the first processor evaluating a security policy to determine whether the second processor meets criteria for altering the first variable; and

the first processor granting permission to the second processor to alter the first variable responsive to determining that the second processor meets the criteria for altering the first variable.

2. The method of claim 1, wherein the first processor is configured to inhibit the second processor from altering the first variable responsive to determining that the second processor does not meet the criteria for altering the first variable, wherein the first variable is stored in a non-volatile memory external to the first processor and accessible to the second processor via the first processor.

3. The method of claim 1, wherein the evaluating includes determining whether the second processor is operating in a recovery mode that provides a reduced set of capabilities relative to a set of capabilities available during execution of an operating system by the second processor.

4. The method of claim 1, wherein the evaluating includes determining whether the second processor has provided proper credentials for altering the first variable.

5. The method of claim 1, wherein the first variable is one of a plurality of boot variables accessed during a boot procedure in which the boot code is executed by the second processor.

6. The method of claim 5, further comprising:

the second processor requesting access to one of the plurality of boot variables during a boot procedure, wherein the first processor evaluates the security policy during the boot procedure to determine whether to grant access to the one of the plurality of variables.

7. The method of claim 5, wherein the first processor is configured to limit access to the plurality of variables by the second processor to a single one of the boot variables at a given time.

8. The method of claim 5, further comprising:

the second processor conveying a request to the first processor to delete one of the plurality of variables;

responsive to receiving the request, the first processor evaluating the security policy to determine whether the second processor is authorized to delete the one of the plurality of variables; and

responsive to determining that the second processor is authorized, the first processor granting permission to the second processor to delete the one of the plurality of variables.

9. The method of claim 5, further comprising:

the first processor adding a new variable to the plurality of variables; and

the first processor setting attributes associated with the new variable.

10. The method of claim 9, wherein the attributes associated with the new variable include one or more of the following:

an operating mode in which the new variable may be altered;

an indication as to whether the new variable can be deleted; and

credentials required for access to the new variable.

11. A computer system comprising:

a main processor;

an auxiliary processor; and

a non-volatile memory a plurality of variables stored therein, wherein the main processor is configured to use the plurality of variables during a boot procedure;

wherein the auxiliary processor is configured to:

receive, at a mailbox circuit, a request to alter a first variable of the plurality of variables by the main processor;

receive an indication of the request being received at the mailbox circuit;

in response to the request and the indication, enforce a security policy that includes determining whether the main processor meets criteria for altering the first variable; and

grant permission to alter the first variable to the main processor responsive to determining that the main processor meets criteria for altering the variable.

12. The computer system of claim 11, wherein the auxiliary processor is configured to:

deny the main processor access to the first variable responsive to determining that the main processor has not met the criteria for altering the variable.

13. The computer system of claim 11, wherein the criteria include a criterion that the main processor is operating in a recovery mode.

14. The computer system of claim 11, wherein the criteria include a criterion that the main processor has provided proper credentials for altering the first variable.

15. The computer system of claim 11, wherein the auxiliary processor is configured to:

verify boot code of the main processor during the boot procedure.

16. The computer system of claim 15, wherein the boot code is stored in the non-volatile memory with the first variable.

17. The computer system of claim 11, wherein the auxiliary processor is configured to:

receive a request from the main processor to delete one of the plurality of variables; and

determine whether to allow deletion of the one of the plurality of variables based on at least a current operating mode of the main processor and credentials provided by the main processor.

18. A method, comprising:

an auxiliary processor in a computer system receiving a request at a mailbox circuit of an auxiliary processor, wherein the request is from a main processor of the computer system to alter a first one of a plurality of boot variables stored in a non-volatile memory accessible to the auxiliary processor;

the auxiliary processor evaluating a security policy to determine whether to grant the main processor permission to alter the first boot variable, wherein determining whether to grant the main processor access includes the auxiliary processor:

determining whether the main processor is operating in recovery mode; and

determining whether the main processor has provided authorization credentials for altering the first boot variable; and

the auxiliary processor granting permission to the main processor to alter the first boot variable to determining that the main processor is operating in the recovery mode and has provided the authorization credentials.

19. The method of claim 18, wherein the non-volatile memory is external to the auxiliary processor, and wherein the plurality of boot variables include variables defined in a unified extensible firmware interface (UEFI) specification.

20. The method of claim 18, wherein the request is received during a boot procedure of the main processor in which the auxiliary processor verifies boot code of the main processor.

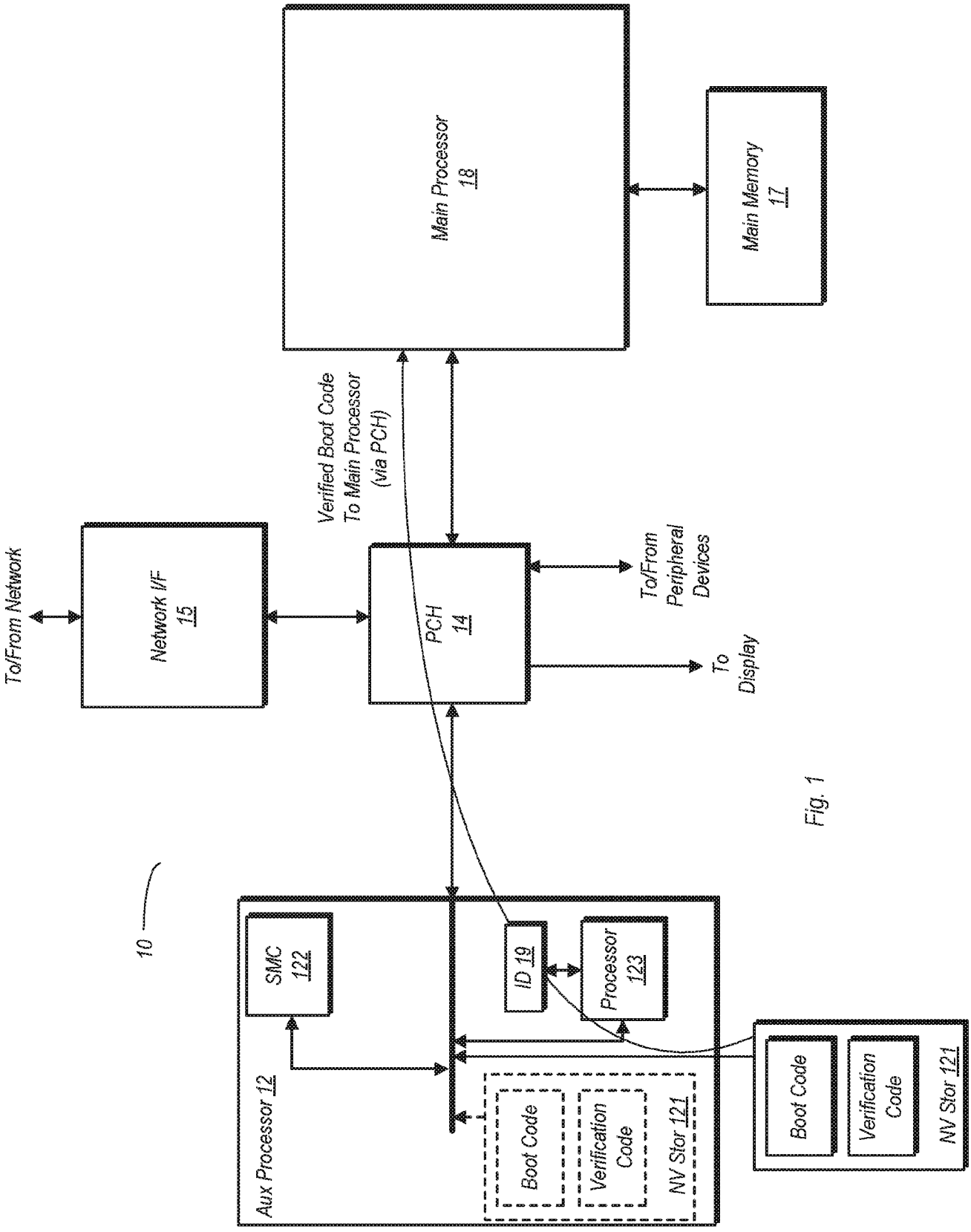
Main Processor 18
Main Memory 17
Verified Boot Code To Main Processor (via PCH)
To/From Network
Network I/F 15
PCH 14
To/From Peripheral Devices
To Display
10
SMC 122
ID 19
Processor 123
Aux Processor 12
Boot Code
Verification Code
NV Stor 121
Boot Code
Verification Code
NV Stor 121

Fig. 1

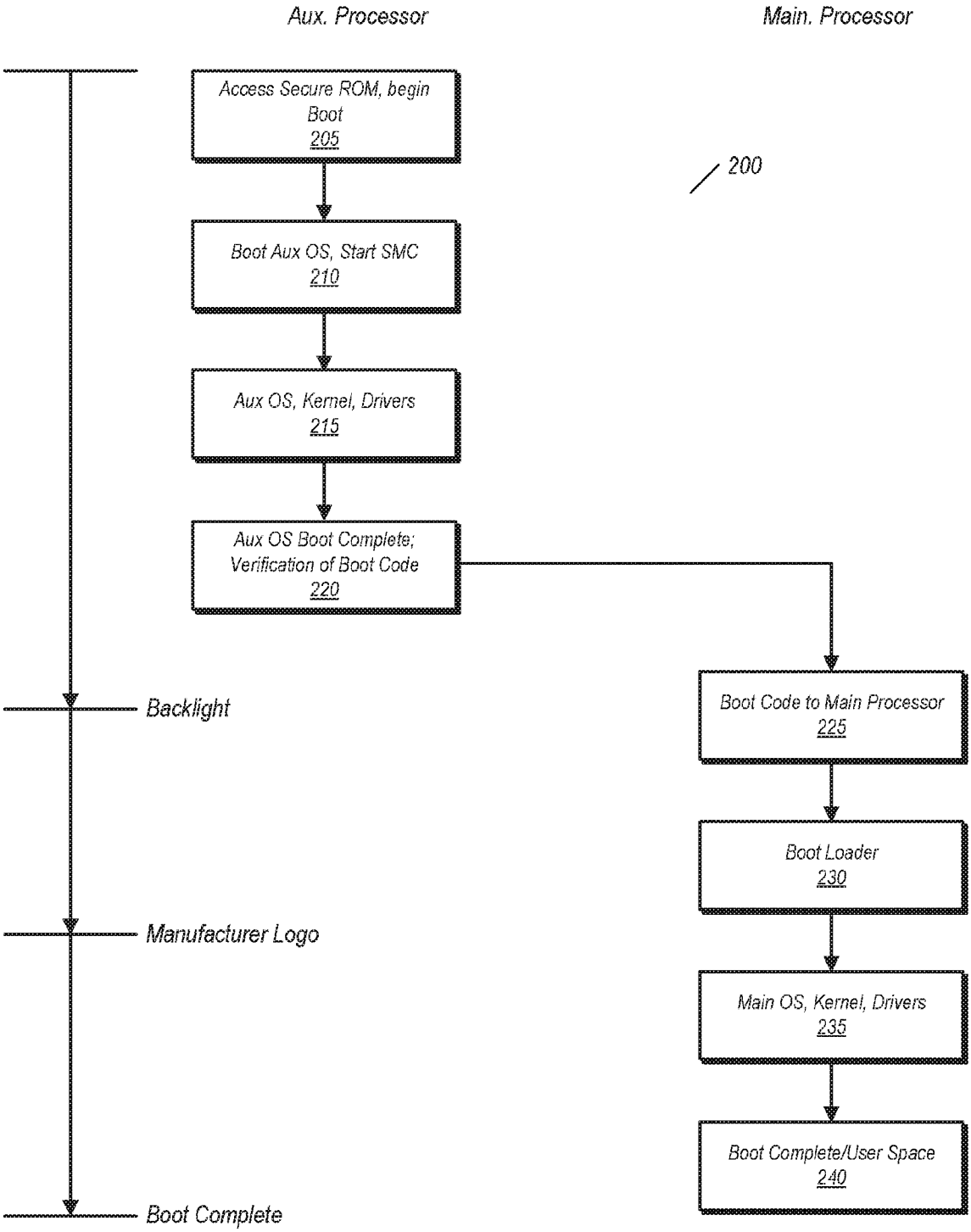
Aux. Processor
Main. Processor
Access Secure ROM, begin Boot
205
200
Boot Aux OS, Start SMC
210
Aux OS, Kernel, Drivers
215
Aux OS Boot Complete; Verification of Boot Code
220
Backlight
Boot Code to Main Processor
225
Boot Loader
230
Manufacturer Logo
Main OS, Kernel, Drivers
235
Boot Complete/User Space
240
Boot Complete

Fig. 2

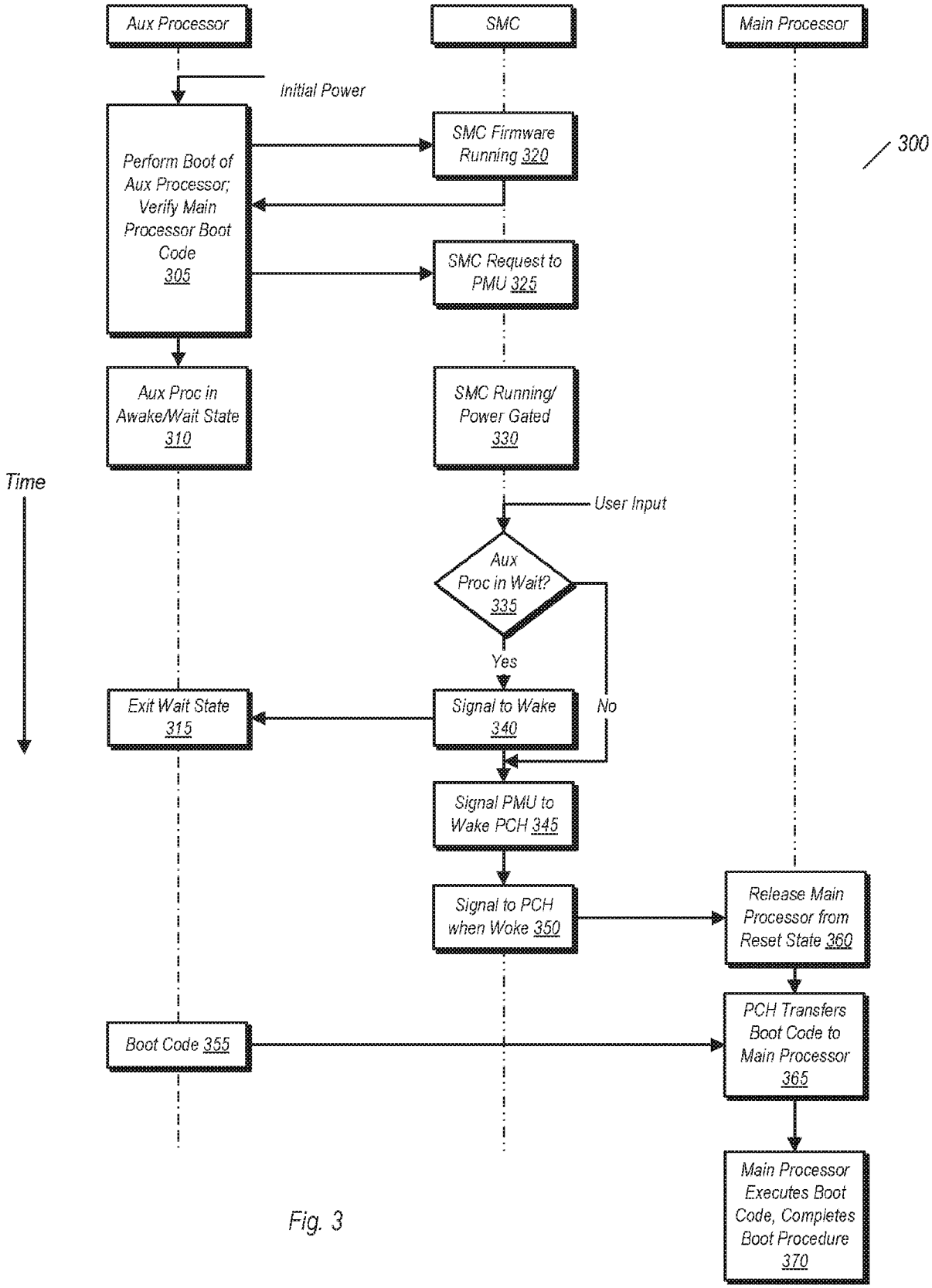
Aux Processor
SMC
Main Processor
Initial Power
300
Perform Boot of Aux Processor; Verify Main Processor Boot Code 305
SMC Firmware Running 320
SMC Request to PMU 325
Aux Proc in Awake/Wait State 310
SMC Running/ Power Gated 330
Time
User Input
Aux Proc in Wait? 335
Yes
No
Exit Wait State 315
Signal to Wake 340
Signal PMU to Wake PCH 345
Signal to PCH when Woke 350
Release Main Processor from Reset State 360
PCH Transfers Boot Code to Main Processor 365
Boot Code 355
Main Processor Executes Boot Code, Completes Boot Procedure 370

Fig. 3

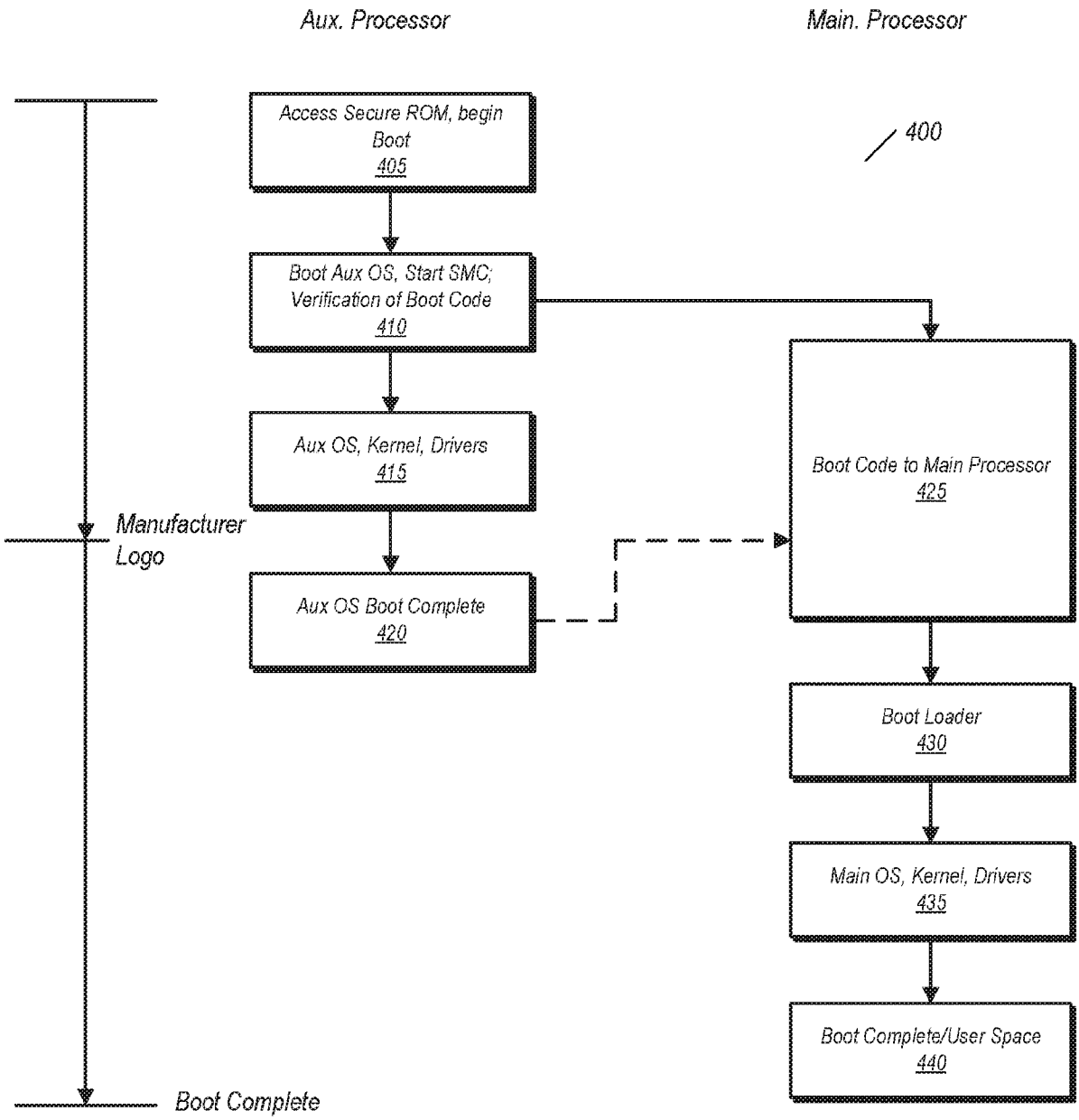
Aux. Processor
Main. Processor
400
Access Secure ROM, begin Boot
405
Boot Aux OS, Start SMC; Verification of Boot Code
410
Aux OS, Kernel, Drivers
415
Aux OS Boot Complete
420
Boot Code to Main Processor
425
Boot Loader
430
Main OS, Kernel, Drivers
435
Boot Complete/User Space
440
Manufacturer Logo
Boot Complete

Fig. 4

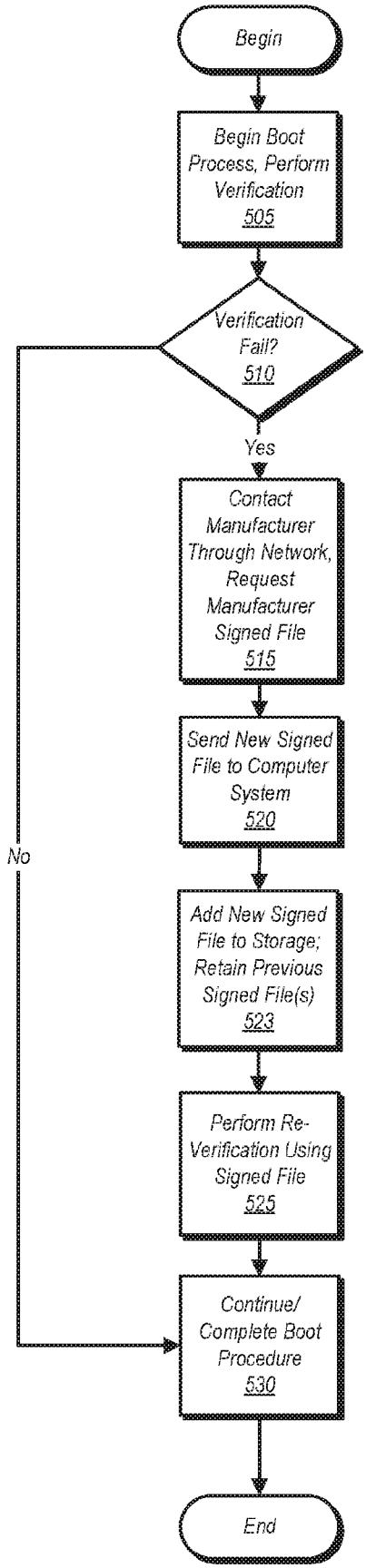
Begin
500
Begin Boot Process, Perform Verification
505
Verification Fail?
510
Yes
No
Contact Manufacturer Through Network, Request Manufacturer Signed File
515
Send New Signed File to Computer System
520
Add New Signed File to Storage; Retain Previous Signed File(s)
523
Perform Re-Verification Using Signed File
525
Continue/ Complete Boot Procedure
530
End

Fig. 5

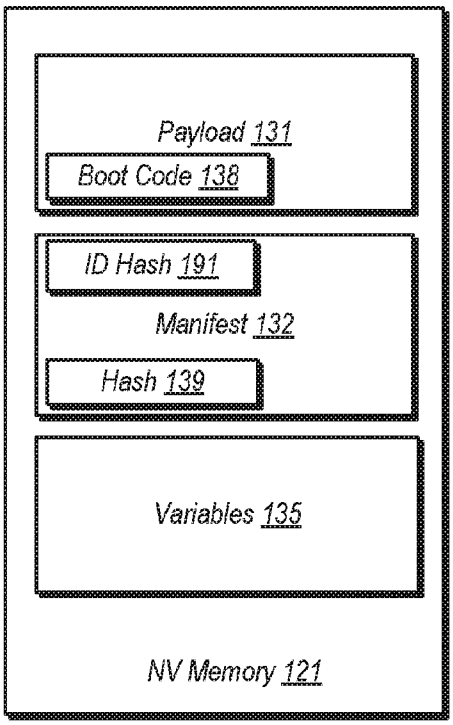
Payload 131
Boot Code 138
ID Hash 191
Manifest 132
Hash 139
Variables 135
NV Memory 121

Fig. 6

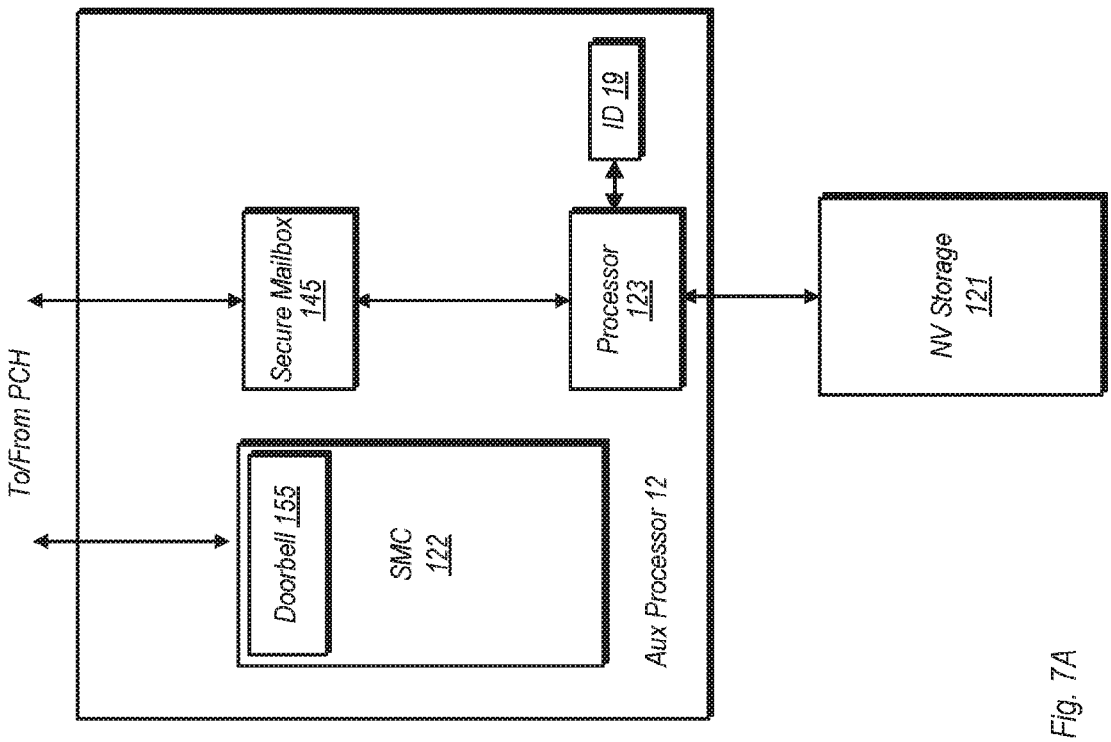
To/From PCH
Secure Mailbox 145
ID 19
Processor 123
NV Storage 121
Doorbell 155
SMC 122
Aux Processor 12

Fig. 7A

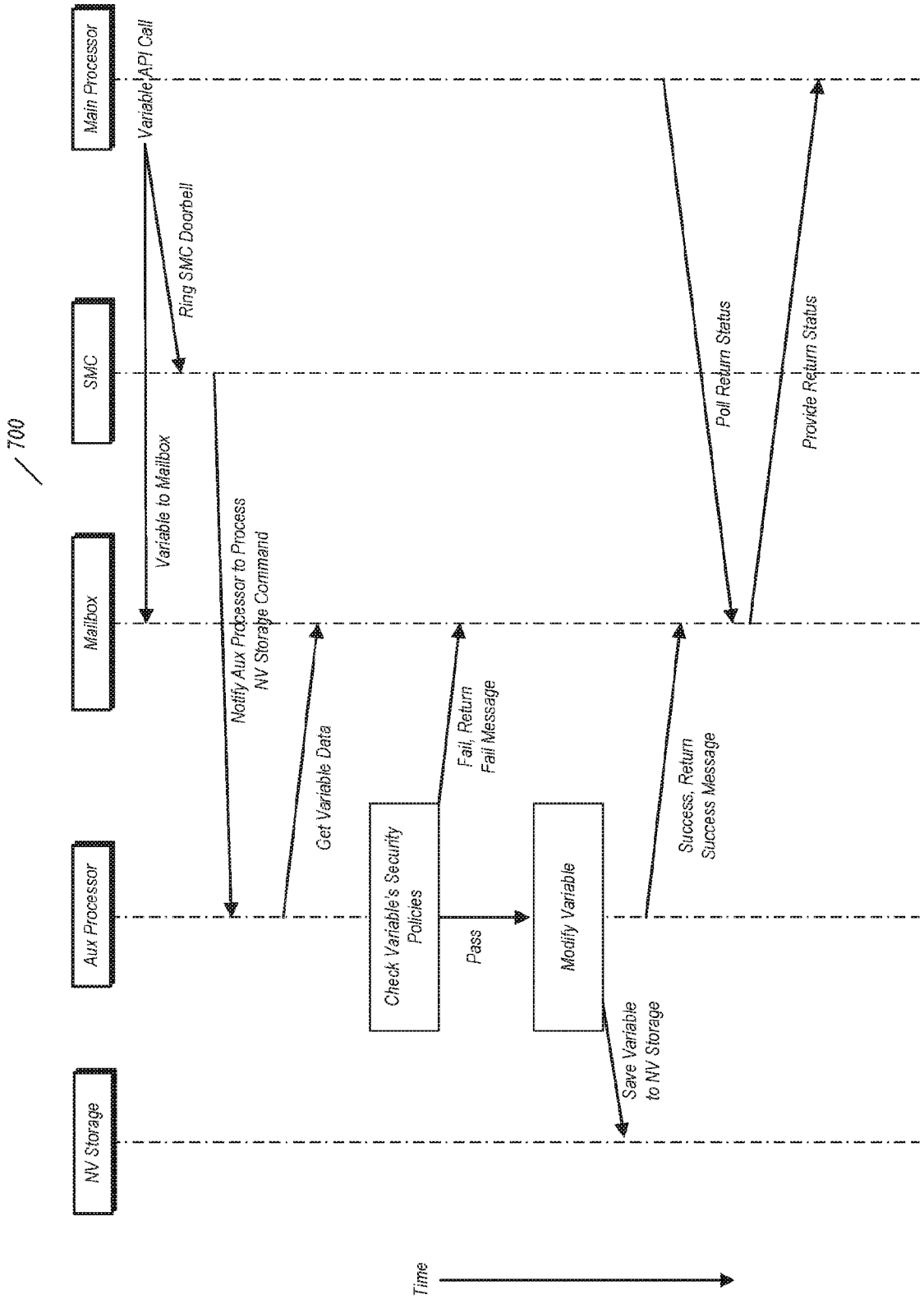
700
Main Processor
SMC
Mailbox
Aux Processor
NV Storage
Variable API Call
Ring SMC Doorbell
Variable to Mailbox
Notify Aux Processor to Process NV Storage Command
Get Variable Data
Check Variable's Security Policies
Fail, Return Fail Message
Pass
Modify Variable
Save Variable to NV Storage
Success, Return Success Message
Poll Return Status
Provide Return Status
Time

Fig. 7B

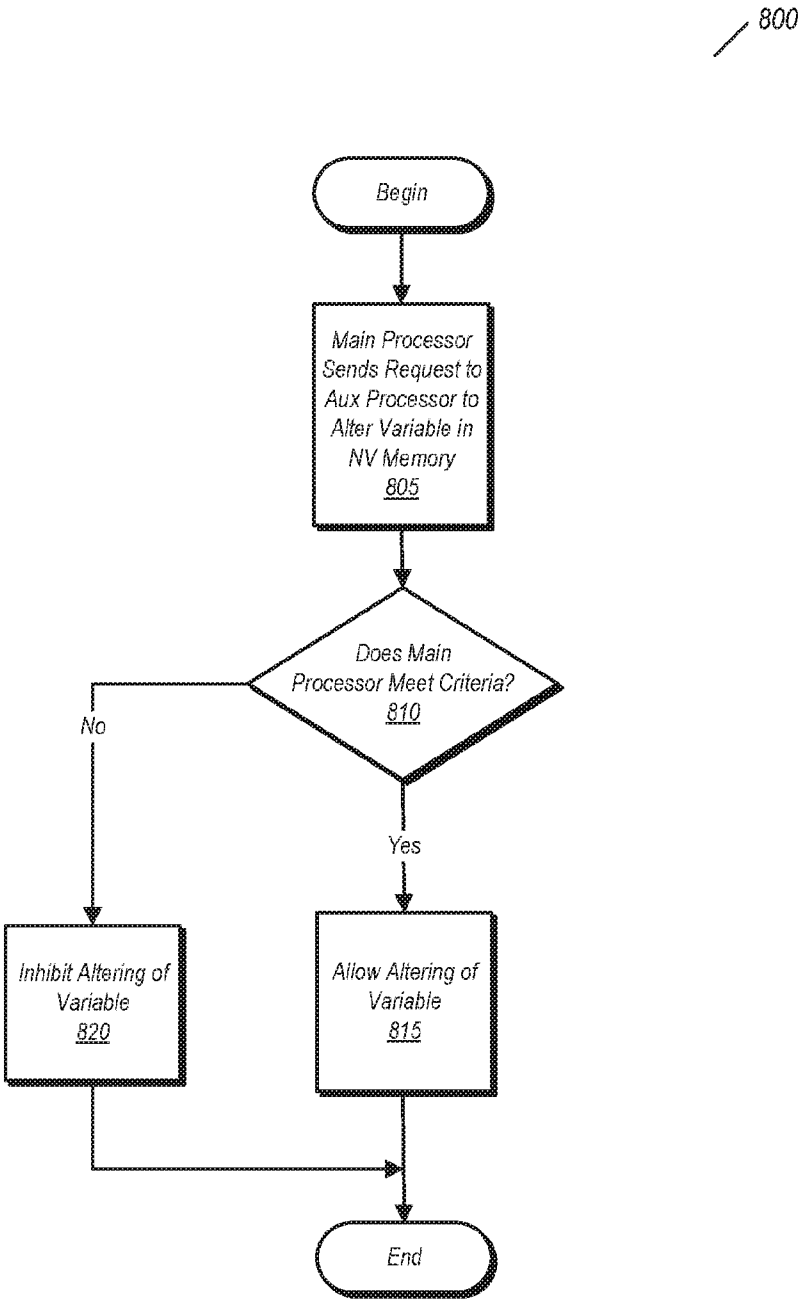
800
Begin
Main Processor Sends Request to Aux Processor to Alter Variable in NV Memory
805
Does Main Processor Meet Criteria?
810
No
Yes
Inhibit Altering of Variable
820
Allow Altering of Variable
815
End

Fig. 8

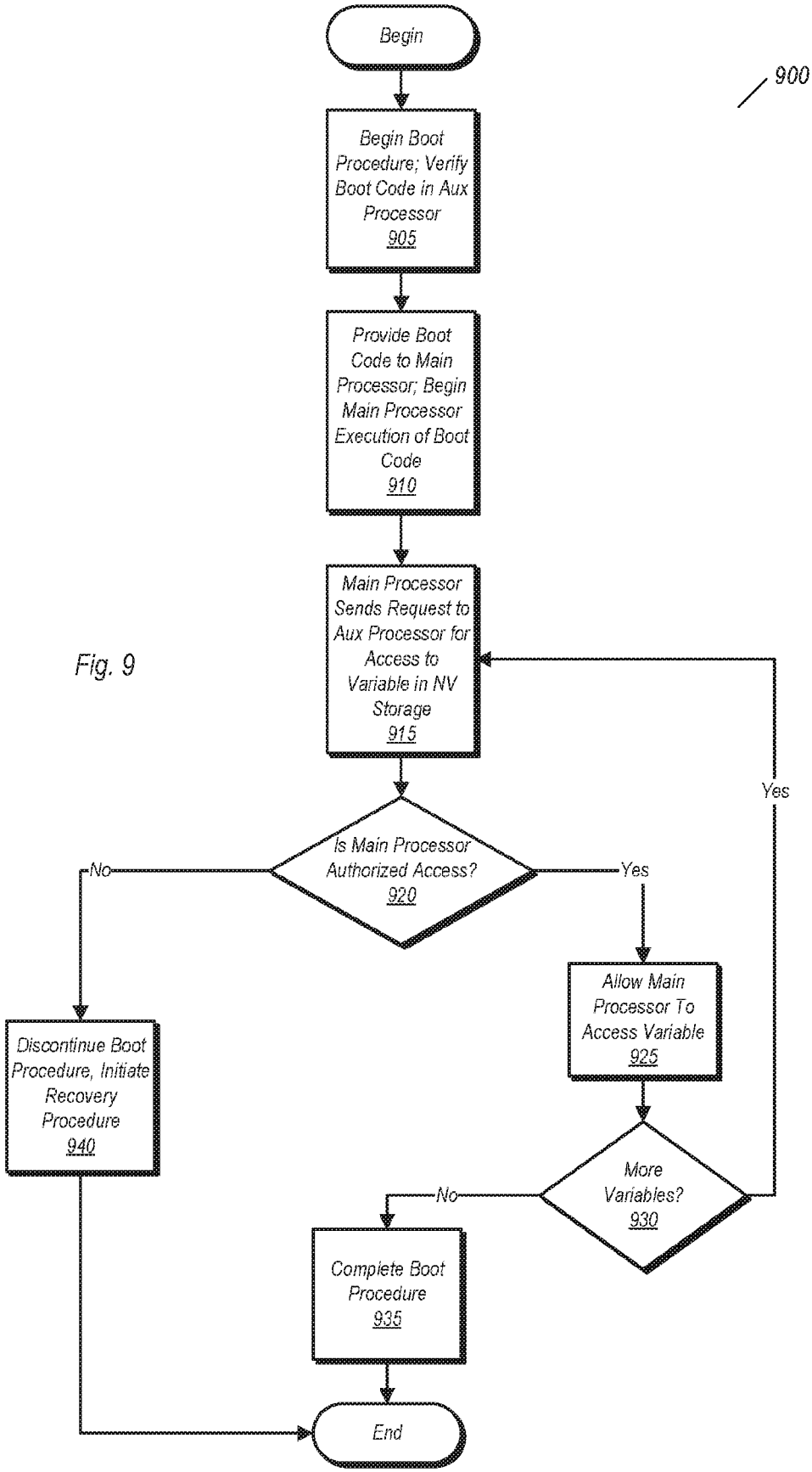
Begin
900
Begin Boot Procedure; Verify Boot Code in Aux Processor
905
Provide Boot Code to Main Processor; Begin Main Processor Execution of Boot Code
910
Main Processor Sends Request to Aux Processor for Access to Variable in NV Storage
915
Is Main Processor Authorized Access?
920
No
Yes
Discontinue Boot Procedure, Initiate Recovery Procedure
940
Allow Main Processor To Access Variable
925
More Variables?
930
Yes
No
Complete Boot Procedure
935
End

Fig. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2018/063686

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/57
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	US 9 202 061 B1 (POLZIN R STEPHEN [US] ET AL) 1 December 2015 (2015-12-01)	1-20
Y	column 3, line 44 - column 21, line 17	1-20
Y	US 2013/212369 A1 (IMTIAZ IMRAN [GB] ET AL) 15 August 2013 (2013-08-15) paragraph [0024] - paragraph [0069]	1-20

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

☒ See patent family annex.

* Special categories of cited documents :

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

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

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of 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 February 2019	21/03/2019

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	Pinto, Raúl

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2018/063686

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 9202061 B1	01-12-2015	US 9202061 B1	01-12-2015
		US 2014089650 A1	27-03-2014
US 2013212369 A1	15-08-2013	CN 103124973 A	29-05-2013
		EP 2619701 A1	31-07-2013
		JP 5745061 B2	08-07-2015
		JP 2013538404 A	10-10-2013
		US 2013212369 A1	15-08-2013
		WO 2012038211 A1	29-03-2012