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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
06 August 2020 (06.08.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/159533 A1

(51) International Patent Classification:
G06F 21/60 (2013.01) *G06F 21/78* (2013.01)

(21) International Application Number:
PCT/US2019/016261

(22) International Filing Date:
01 February 2019 (01.02.2019)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 10300 Energy Drive, Spring, Texas 77389 (US).

(72) Inventors: **BRADUKE, Rosilet Retnamoni**; 11445 Compaq Center West Drive, Houston, Texas 77070 (US). **ANBAZHAGAN, Baraneedharan**; 11445 Compaq Center West Drive, Houston, Texas 77070 (US). **STEWART, Christopher H.**; 11445 Compaq Center West Drive, Houston, Texas 77070 (US).

(74) Agent: **SU, Benjamin** et al.; HP Inc., Mail Stop 35 3390 E. Harmony Road, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, 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).

(54) Title: SECURITY CREDENTIAL DERIVATION

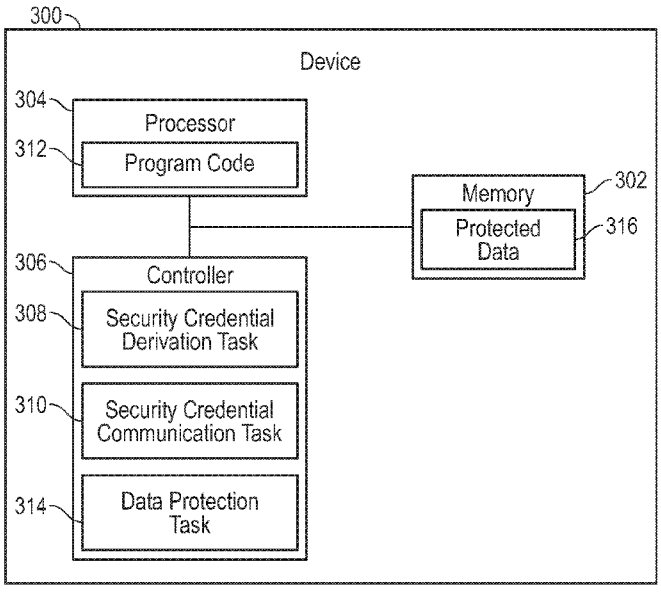

FIG. 3

(57) Abstract: In some examples, a device includes a memory, a processor, and a controller separate from the processor to derive a security credential based on information comprising a key accessible by the controller. The controller communicates the derived security credential in a secure manner to a program code executable on the processor, and uses the derived security credential to protect data stored in the memory against unauthorized access.

WO 2020/159533 A1

[Continued on next page]

Declarations under Rule 4.17:

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

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

Published:

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

SECURITY CREDENTIAL DERIVATION

Background

[0001] Electronic devices can include various components for performing different tasks. For example, the components can include a processor, a memory, an embedded controller, an input/output (I/O) device, and other components. Various code (in the form of machine-readable instructions including firmware and/or software) are executable on the embedded controller, the processor, and other components.

Brief Description of the Drawings

[0002] Some implementations of the present disclosure are described with respect to the following figures.

[0003] Fig. 1 is a block diagram of an electronic device according to some examples.

[0004] Fig. 2 is a flow diagram of a process according to some examples.

[0005] Fig. 3 is a block diagram of a device according to some examples.

[0006] Fig. 4 is a block diagram of a storage medium storing machine-readable instructions according to some examples.

[0007] Fig. 5 is a flow diagram of a process according to further examples.

[0008] Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements. The figures are not necessarily to scale, and the size of some parts may be exaggerated to more clearly illustrate the example shown. Moreover, the drawings provide examples and/or implementations consistent with the description; however, the description is not limited to the examples and/or implementations provided in the drawings.

Detailed Description

[0009] In the present disclosure, use of the term "a," "an", or "the" is intended to include the plural forms as well, unless the context clearly indicates otherwise. Also, the term "includes," "including," "comprises," "comprising," "have," or "having" when used in this disclosure specifies the presence of the stated elements, but do not preclude the presence or addition of other elements.

[0010] An electronic device is vulnerable to an attack by an unauthorized entity, such as malware loaded into the electronic device, or a user that attempts to gain unauthorized access of the electronic device. Examples of electronic devices include any or some combination of the following: computers (e.g., desktop computers, notebook computers, tablet computers, server computers, etc.), handheld devices (e.g., smartphones, game appliances, etc.), wearable devices (e.g., smart watches, head-mounted devices, smart eyeglasses, etc.), Internet-of-Things (IoT) devices, controllers in vehicles, storage systems, communication nodes, and so forth.

[0011] An electronic device can include a memory that is used to store data. In some examples, the memory may be in the form of a nonvolatile memory, which is a memory that retains stored data even if power is removed from the memory. In other examples, a memory can include volatile memory, which is a memory that loses its stored data if power is removed from the memory.

[0012] An unauthorized entity may attempt to access certain information stored in a memory. The unauthorized entity can be in the form of malware executing in an electronic device. Alternatively, the unauthorized entity can be a user who may attempt to gain access to information stored in a memory.

[0013] To protect against unauthorized access of data stored in a memory, the data may be protected such that access of the data is possible if a requester of the data presents a security credential, such as a password, an encryption key, or another type of credential.

[0014] As an example, a boot code may store, in the memory, data associated with establishing a wireless connection with a wireless network (e.g., a wireless local area network or WLAN, a cellular network, etc.). The data for establishing the wireless connection with the wireless network may include a password or private key that is used to gain access to the wireless network. The boot code may be configured to automatically establish the wireless connection with the wireless network, such as during a pre-boot phase of the electronic device. The pre-boot phase can refer to a phase prior to starting of an operating system of the electronic device. It may be inconvenient to seek user input of the password or private key each time the boot code attempts to establish a wireless connection. Thus, the password or private key may be stored in the memory for use by the boot code in automatically performing pre-boot establishment of the wireless connection.

[0015] Although an example is described in the context of pre-boot establishment of a wireless connection with a wireless network, it is noted that the data stored in the memory can also be used to perform a wireless connection with the wireless network after the electronic device has completed booting.

[0016] Also, in other examples, other types of data (used for other purposes) stored in a memory may be subject to unauthorized access.

[0017] An unauthorized entity may attempt to gain access of the password or other private data stored in the memory that the unauthorized entity can then use to gain access of the wireless network, or for another purpose.

[0018] To protect against unauthorized access of the data stored in the memory, the data may be encrypted using an encryption key. Some example techniques of generating encryption keys may be associated with various issues. For example, a key derivation technique that prompts a user to enter a password or other information for deriving an encryption key may reduce user convenience since the user has to remember the information to be entered to generate the encryption key. In other examples, an electronic device may have to implement a specialized

cryptographic subsystem to generate the encryption key, which may be associated with increased complexity and cost of the electronic device.

[0019] In accordance with some implementations of the present disclosure, as shown in Fig. 1, an electronic device 100 includes an embedded controller 102 to generate a security credential 118 that can be used to protect data stored in a memory against unauthorized access. The electronic device 100 further includes a processor 104 that is separate from the embedded controller 102.

[0020] In some examples, the embedded controller 102 can be used to perform specific tasks. The tasks of the embedded controller 102 can be performed by embedded controller (EC) code 106, in the form of machine-readable instructions such as EC firmware or EC software, executed on the embedded controller 102. In other examples, the tasks of the embedded controller 102 can be performed by a hardware processing circuit of the embedded controller 102. Examples of tasks that can be performed by the embedded controller 102 include any one or some combination of the following: power supply control in the electronic device 100 (for controlling a power supply that supplies power supply voltages to various components in the electronic device 100), charging and control of a battery in the electronic device 100, thermal monitoring (to monitor a temperature in the electronic device 100), fan control (to control a fan in the electronic device 100), and interaction with a user input device (such as performing a scan of a keyboard of the electronic device 100 or interaction with a pointing device such as a mouse, touchpad, touchscreen, and so forth). In other examples, the embedded controller 102 can perform additional or alternative tasks. The embedded controller 102 can be implemented with a microcontroller, an application-specific integrated circuit (ASIC), a programmable gate array (PGA), or any other type of programmable circuit.

[0021] The security credential 118 generated by the embedded controller 102 can be in the form of an encryption key, a password, or any other information that can be used to encrypt data or otherwise protect data against unauthorized access. For example, a requester is unable to access encrypted information unless the requester presents an encryption key that can be used to decrypt the encrypted

information. In other examples, an access mechanism prevents access of the information unless a security credential (e.g., a password, a security code, etc.) is presented that matches a specified security credential.

[0022] The embedded controller 102 has access to specified information that is inaccessible to the processor 104 or any other entity in the electronic device 100. The specified information can include a key or any other type of information that the embedded controller 102 is able to access. Another component (such as the processor 104) is unable to access the specified information. The embedded controller 102 is able to generate the security credential 118 based on the specified information that no other entity in the electronic device 100 can access. The derived credential 118 can be shared with a program code (including machine-readable instructions) executable on the processor 104, so that the embedded controller 102 does not have to share the specified information (such as the key in the specified information) with another entity.

[0023] In some examples, the specified information can be hardcoded into the embedded controller 102. For example, the specified information can be hardcoded by fusing the specified information into the embedded controller 102. Fusing refers to programming fuses of the embedded controller 102 to provide a collection of values that make up the specified information. In further examples, the embedded controller 102 can include an internal nonvolatile storage to store the specified information. As yet further examples, the specified information can be stored in an external nonvolatile storage that is external to and accessible by the embedded controller 102. The internal or external nonvolatile storage is electrically isolated way from any other entities (other than the embedded controller 102), and thus inaccessible, to the other entities of the electronic device. The internal or external nonvolatile storage in such examples may include a one-time programmable (OTP) storage, which is a storage that can be programmed once with data values. Once the OTP storage is programmed, the data values stored in the OTP storage cannot be changed.

[0024] In the foregoing examples in which the specified information is fused into the embedded controller 102 or stored in an (internal or external) OTP storage, the specified information can be referred to as OTP information 122. Such OTP information 122 can also be referred to as "hardware information." Although reference is made to "OTP information" in the present discussion, in other examples, other specified information stored in re-writable storage can be used for deriving an encryption key or other security credential.

[0025] Fig. 1 shows that the OTP information 122 can be part of the embedded controller 102 (either fused or stored in an internal OTP storage), or alternatively, can be stored in an external OTP storage 124. The OTP information 122 can be programmed in the embedded controller 102 or in the external OTP storage 124 at build time of the electronic device 100 at a factory or as part of the configuration of the electronic device 100 prior to delivery to an end user.

[0026] In some examples, the OTP information 122 includes a key 126. This key that is part of the OTP information 122 may be referred to as a "hardware key" that the embedded controller 102 can use to encrypt information.

[0027] As discussed further below, a security credential generation logic 120 of the embedded controller 102 can use the OTP information 122 to derive the security credential 118. The security credential generation logic 120 can be implemented using a portion of the hardware processing circuit of the embedded controller 102, or alternatively, can be implemented as machine-readable instructions (e.g., part of the EC code 106) executable by the embedded controller 102.

[0028] The security credential 118 being derived based on the OTP information 122 can refer to the security credential 118 being derived based on the entirety of the OTP information 122 or a portion of the OTP information 122, and possibly based on additional information (discussed further below).

[0029] The embedded controller 102 can transmit the derived security credential 118 in a secure manner to a program code executable by the processor 104. A

discussion of how information can be communicated in a "secure manner" is provided further below. Note that the derived security credential 118 that is transferred to the program code in the secure manner is inaccessible by any other entity of the electronic device 100 outside the context of the secure transfer. Further, no other entity (aside from the embedded controller 102) can derive the derived security credential 118 because no other entity can access the OTP information 122.

[0030] Note that the OTP information 122 (or a portion of the OTP information 122) is not communicated to the program code, so that the OTP information (portion) 122 is not exposed to the program code executable by the processor 104. Consequently, even if the electronic device 100 is subject to an attack by an entity that has access to the processor 104 (e.g., malware executable on the processor 104), the OTP information 122 is protected against unauthorized access. The derived security credential 118 can be considered to be equivalent in cryptographic integrity with the OTP information 122. However, the derived security credential 118 is not identical to the OTP information 122 (or to a key in the OTP information 122).

[0031] The program code executable on the processor 104 can encrypt data stored in a memory (e.g., shared memory 108) using the derived security credential 118. Such encrypted data is represented as protected data 116 in Fig. 1. "Protected data" refers to any data that is to be protected against unauthorized access by a malicious entity, such as malware, an unauthorized user, and so forth. In other examples, the security credential 118 can protect the data (protected data 116) in the shared memory 108 in a different manner—for example, an access mechanism (e.g., a memory controller, a file system, etc.) that manages access of the data stored in the shared memory 108 may prompt for the security credential 118 before allowing access of the data.

[0032] In examples according to Fig. 1, the EC code 106 executable by the embedded controller 102 is stored in the shared memory 108. The shared memory 108 can include a nonvolatile memory, such as a flash memory or other type of nonvolatile memory. The shared memory 108 is accessible by various components of the electronic device 100, including the embedded controller 102, the processor

104, and other components. The shared memory 108 can also store a boot code 110, which is used for booting the electronic device 100 or to resume the electronic device 100 from a low power state. The boot code 110 can be in the form of a Basic Input/Output System (BIOS) code.

[0033] The BIOS code can perform checking of hardware components to ensure that the hardware components are present and functioning properly. This can be part of a power-on self-test (POST) procedure, for example. After the POST procedure, the BIOS code can progress through the remainder of a booting sequence, after which the BIOS code can load and pass control to an operating system (OS) 112. BIOS code can also refer to Unified Extensible Firmware Interface (UEFI) code. In some examples, the BIOS code can also include a runtime portion that is executed after the OS 112 loads.

[0034] Although Fig. 1 shows the entirety of the boot code 110 stored in the shared memory 108, it is noted that in other examples, a first portion of the boot code 110 is stored in the shared memory 108, while another portion of the boot code 110 is stored in a separate storage, such as a storage 114. The storage 114 can include a persistent storage, such as a persistent storage implemented using a disk-based storage, a solid state storage, and so forth. The OS 112 is also stored in the storage 114. In other examples, if the shared memory 108 has sufficient capacity, the OS 112, or a portion of the OS 112, can also be stored in the shared memory 108.

[0035] In the example of Fig. 1, the protected data 116, the boot code 110 (or a portion of the boot code 110), and the EC code 106 are stored in the shared memory 108. In other examples, the foregoing pieces of data or program code can be stored in different memories or storage. Note that the boot code 110 and the EC code 106 (or portions of the boot code 110 and EC code 106) can be considered to be part of the protected data 116 in some examples.

[0036] Further examples of the protected data 116 can include any or some combination of the following: data useable to establish a wireless connection with a

wireless network (e.g., a password or other private data such as a private key that is used as part of the process of establishing the wireless connection), other passwords and/or keys, configuration information for the electronic device 100 or components of the electronic device 100, program code, or any other information deemed to be sensitive such that it is to be protected against unauthorized access.

[0037] To generate the security credential 118, the OTP information (portion) 122 is input as a seed to a security credential derivation function 123 executed by the security credential generation logic 120. As an example, the security credential generation function 123 executed by the security credential generation logic 120 is a key derivation function, such as a Password-Based Key Derivation Function 2 (PBKDF2). The key derivation function can apply a pseudorandom function, such as a hash-based message authentication code (HMAC), to an input seed (which in this case includes the OTP information 122 or a portion of the OTP information 122) to generate a derived encryption key. The derived encryption key is an example of the security credential 118.

[0038] In some examples, the portion of the OTP information 122 used by the key derivation function can include the hardware key 126 that is part of the OTP information 122. Another input to the key derivation function can include a salt value, which can include a random number produced by a random number generator (RNG) 128, which can be part of (or be coupled to) the security credential generation logic 120. Thus, the OTP information (portion) 122 and the random number produced by the RNG 120 are input to the key derivation function, which produces a derived encryption key.

[0039] In other examples, another type of the security credential derivation function 123 can be used by the security credential generation logic 120 to produce the security credential 118.

[0040] The security credential 118 can be communicated in a secure manner over a secure channel 130 to a program code executed by the processor 104. For example, the program code executed by the processor 104 can include the boot

code 110. Communicating the security credential 118 in the secure channel 130 protects the security credential 118 against unauthorized access by an entity that executes in the electronic device 100 or that has access to the electronic device 100. The protection against the unauthorized access in the secure channel 130 can be based on use of a protection code and/or use of a trusted interface between the embedded controller 102 and the processor 104.

[0041] In some examples, the protection code to protect the security credential 118 communicated in the secure channel 130 includes a value that can be used to verify the integrity of the security credential 118. For example, the protection code can be in the form of an HMAC, which can be used to verify the integrity of the security credential 118, as well as the authenticity of information (e.g., a message) that includes the security credential 118. Note that the HMAC is not used to encrypt the information including the security credential 118, but rather, an HMAC value is transferred with the information including the security credential 118, and the HMAC value (e.g., an HMAC hash value) can be used to verify the integrity and the authenticity of the information including the security credential 118.

[0042] In further examples, as an alternative to or in addition to using a protection code such as an HMAC to protect communication of the security credential 118 over the secure channel 130, a trusted interface 132, such as in the form of a trusted application programming interface (API), can be used as part of the secure channel 130. The trusted interface 132 can have an open state or a closed state. In the open state, the trusted interface 132 allows for information to be passed between the embedded controller 102 and a program code (e.g., the boot code 110) executing on the processor 104. In the closed state, the trusted interface 132 does not allow for communication of information between the embedded controller 102 and the program code executing on the processor 104.

[0043] Whether the trusted interface 132 is open or closed is based on a flag 134 stored in a volatile memory 136 connected to the embedded controller 102. A volatile memory can include a dynamic random access memory (DRAM), a static

random access memory (SRAM), or any other type of memory that loses its content if power is removed from the memory.

[0044] The flag 134 can include a bit or multiple bits stored in the volatile memory 136. If the flag 134 is set to a first value, then the trusted interface 132 is in the open state. However, if the flag 134 is set to a different second value, then the trusted interface 132 is in the closed state. In some examples, when the electronic device 100 initially starts, the flag 134 can default to the first value that corresponds to the trusted interface 132 being in the open state. During the time that the flag 134 is set to the first value, the trusted interface 132 can be used to perform communication between the embedded controller 102 and a program code (e.g., the boot code 110) executing on the processor 104.

[0045] During the time that the flag 134 is set to the first value and the trusted interface 132 is in the open state, the program code that executes on the processor 104 is deemed to be secure. For example, during this time, the part of the boot code 110 that executes on the processor 104 is from a secure portion of the shared memory 108 or is subject to verification, such as by the embedded controller 102, so that the electronic device 100 can trust the part of the boot code 110 that executes while the trusted interface 132 is in the open state.

[0046] The boot code 110 executing on the processor 104 can subsequently inform the embedded controller 102 to set the flag 134 to the second value that corresponds to the trusted interface 132 being in the closed state. For example, the boot code 110 can inform the embedded controller 102 to set the flag 134 to the second value in response to the boot code 110 exiting a specified phase.. Alternatively, the embedded controller 102 can close the trusted interface 132 by setting the flag 134 to the second value after the derived security credential 118 has been retrieved once.

[0047] Thus, more generally, the trusted interface 132 is available for communicating information during an initial boot phase of the boot code 110, and

unavailable for communicating information after the initial boot phase of the boot code 110.

[0048] For example, the trusted part of the boot code 110 that can execute while the trusted interface 132 is open can include a Platform Initialization (PI) or Pre-EFI (PEI) code of the BIOS. The PI or PEI code can retrieve the derived security credential 118 over the open trusted interface 132, after which either the BIOS can close the trusted interface 132 or the embedded controller 102 closes the trusted interface 132 after the derived security credential 118 has been retrieved once. The PI/PEI stage of execution is deemed a trusted platform state, since a small number of entities are running at this point, and none are deemed capable of intruding on the embedded controller 102 to BIOS communication. The PEI code can make the derived security credential 118 available in a piece of memory referred to in the UEFI context as a hand-off block (HOB). An early Driver eXecution Environment (DXE) code of the BIOS can read the HOB; store the content of the HOB in a System Management Random Access Memory (SMRAM); and finally, securely delete the HOB's contents from memory using a cryptographically secure overwrite method. The DXE phase is the phase of the boot code 110 when the boot code 110 loads drivers for configured components in an electronic device 100. The early DXE code is deemed to be just as secure as the PI/PEI code. Before other code is allowed to execute, the DXE code locks the SMRAM so that no other entity can read from or write to the content (which includes the derived security credential 118) of the SMRAM. The BIOS code can read, update, or delete the content of the SMRAM—however, no other entity is able to access the SMRAM. As a result, storage of the derived security credential 118 in the SMRAM is deemed relatively secure.

[0049] After the trusted interface 132 is closed, the trusted interface 132 can be re-opened by a power on/off cycle of the electronic device 100, and the foregoing process can be re-iterated.

[0050] The secure channel 130 can refer to either or both of: a channel (such as a bus or other communication link) over which information is protected by a protection code (e.g., an HMAC), or the trusted interface 132.

[0051] For example, if the HMAC is used (and the trusted interface 132 is not used) to protect the security credential 118, the embedded controller 102 generates an HMAC, and transmits the security credential 118 along with the HMAC to the boot code 110 (or other program code) executable on the processor 104.

[0052] If the trusted interface 132 is used (but the HMAC is not used) to protect the security credential 118, the embedded controller 102 transmits the security credential 118 to the boot code 110 (or other program code) executable on the processor 104 while the trusted interface 132 is in the open state. Once the trusted interface 132 transitions to the closed state, the embedded controller 102 does not transmit the security credential 118 to the boot code 110 (or other program code).

[0053] In other examples, both the HMAC and the trusted interface 132 can be used. In such examples, key information used for deriving the HMAC can be exchanged between the embedded controller 102 and boot code 110 (or other program code) executing on the processor 104 while the trusted interface 132 is in the open state. Once the trusted interface 132 transitions to the closed state, exchanging key information between the embedded controller 102 and a program code executing on the processor 104 is not performed. However, in the closed state of the trusted interface 132, the HMAC can be used to protect the security credential 118 communicated between the embedded controller 102 and the boot code 110 (or other program code) executing on the processor 104.

[0054] After receiving the security credential 118 from the embedded controller 102, the boot code 110 (or other program code) executing on the processor 104 can use the security credential 118 to encrypt data or otherwise protect data that is stored as the protected data 116 in the shared memory 108. For example, if the security credential 118 is a derived encryption key, the derived encryption key can be used by the boot code 110 (or other program code) to encrypt data that is stored as the protected data 116. The encrypted protected data 116 is not accessible by an entity that does not have the encryption key.

[0055] As another example, the protected data 116 stored in the shared memory 108 is not encrypted. However, the boot code 110 (or other program code) can manage access of the protected data 116 using the security credential 118.

[0056] Fig. 2 is a message flow diagram of a process according to some examples, which can be performed by the embedded controller 102 and the boot code 110 executed on the processor 104. The embedded controller 102 accesses (at 202) the OTP information 122. The embedded controller 102 generates (at 204) a salt value. For example, the salt value is a random number, and the embedded controller 102 uses the RNG 120 to generate the random number.

[0057] The OTP information (or a portion of the OTP information 122) and the salt value are input (at 206) to the security credential derivation function 123 to derive the security credential 118.

[0058] The embedded controller 102 sends (at 208) the derived security credential 118 over the secure channel 130 to the boot code 110. The boot code 110 protects (at 210) data using the derived security credential 118, and the protected data is stored as the protected data 116 in the shared memory 108.

[0059] Fig. 3 is a block diagram of a device 300 (e.g., an electronic device) that includes a memory 302, a processor 304, and a controller 306 (similar to the embedded controller 102, for example) separate from the processor 304. A processor can include a microprocessor, a core of a multi-core microprocessor, a microcontroller, a programmable integrated circuit, a programmable gate array, a digital signal processor, or another hardware processing circuit.

[0060] The controller 306 can perform various tasks. The tasks include a security credential derivation task 308 that derives a security credential based on information (e.g., the OTP information 122) including a key (e.g., the hardware key 126) accessible by the controller 306. Note that security credential can be derived based on just the hardware key 126, or based on the hardware key 126 and other information, such as the remaining portion of the OTP information 122.

[0061] The tasks further include a security credential communication task 310 that communicates the derived security credential in a secure manner to a program code 312 executable on the processor 304.

[0062] The tasks further include a data protection task 314 that uses the derived security credential to protect data 316 stored in the memory 302 against unauthorized access.

[0063] Fig. 4 is a block diagram of a non-transitory machine-readable or computer-readable storage medium 400 that stores machine-readable instructions executable by a controller to perform respective tasks. The machine-readable instructions include key information access instructions 402 to access information comprising a key accessible by the controller and inaccessible by a processor that is separate from the controller. The machine-readable instructions further include security credential derivation instructions 404 to input the information including the key into a security credential derivation function to produce a derived security credential.

[0064] The machine-readable instructions additionally include security credential communication instructions 406 to communicate the derived security credential over a secure channel to a program code executable on the processor, the derived security credential for use in protecting data stored in a memory.

[0065] Fig. 5 is a flow diagram of a process according to some examples. The process includes accessing (at 502), by an embedded controller of an electronic device, one-time programmable information.

[0066] The process includes using (at 504), by the embedded controller, the one-time programmable information as a seed in producing a derived security credential. The process further includes communicating (at 506), by the embedded controller, the derived security credential in a secure manner to a program code executable on a processor of the electronic device. The process further includes protecting (at 508), using the derived security credential, data stored in a memory.

[0067] The storage medium 400 (Fig. 4) can include any or some combination of the following: a semiconductor memory device such as a dynamic or static random access memory (a DRAM or SRAM), an erasable and programmable read-only memory (EPROM), an electrically erasable and programmable read-only memory (EEPROM) and flash memory; a magnetic disk such as a fixed, floppy and removable disk; another magnetic medium including tape; an optical medium such as a compact disc (CD) or a digital video disc (DVD); or another type of storage device. Note that the instructions discussed above can be provided on one computer-readable or machine-readable storage medium, or alternatively, can be provided on multiple computer-readable or machine-readable storage media distributed in a large system having possibly plural nodes. Such computer-readable or machine-readable storage medium or media is (are) considered to be part of an article (or article of manufacture). An article or article of manufacture can refer to any manufactured single component or multiple components. The storage medium or media can be located either in the machine running the machine-readable instructions, or located at a remote site from which machine-readable instructions can be downloaded over a network for execution.

[0068] In the foregoing description, numerous details are set forth to provide an understanding of the subject disclosed herein. However, implementations may be practiced without some of these details. Other implementations may include modifications and variations from the details discussed above. It is intended that the appended claims cover such modifications and variations.

What is claimed is:

1. A device comprising:

a memory;

a processor; and

a controller separate from the processor to:

derive a security credential based on information comprising a key accessible by the controller;

communicate the derived security credential in a secure manner to a program code executable on the processor; and

use the derived security credential to protect data stored in the memory against unauthorized access.

2. The device of claim 1, wherein the controller comprises an embedded controller.

3. The device of claim 1, wherein the information comprising the key is hardcoded into the controller, the information comprising the key inaccessible to the processor.

4. The device of claim 3, wherein the information comprising the key is programmed at build time of the device at a factory or as part of a configuration of the device prior to delivery to an end user.

5. The device of claim 1, wherein the communication of the derived security credential is protected against unauthorized access using a protection code.

6. The device of claim 5, wherein the protection code comprises a hash-based message authentication code (HMAC).

7. The device of claim 1, wherein the program code comprises a boot code for booting the device, and wherein the derived security credential is communicated in the secure manner using a trusted interface between the controller and the boot code, the trusted interface being available for communicating information during an initial boot phase of the boot code, and unavailable for communicating information after the initial boot phase of the boot code.

8. The device of claim 1, wherein the controller is to use the information comprising the key as a seed to generate the derived security credential.

9. The device of claim 8, wherein the controller is to input the information comprising the key and a salt value to a security credential derivation function that in response is to output the derived security credential.

10. The device of claim 9, wherein the controller comprises a random number generator to generate a random number used as the salt value.

11. A non-transitory machine-readable storage medium comprising instructions that upon execution cause a controller of a device.to:

access information comprising a key accessible by the controller and inaccessible by a processor of the device, the processor separate from the controller;

input the information comprising the key into a security credential derivation function to produce a derived security credential; and

communicate the derived security credential over a secure channel to a program code executable on the processor, the derived security credential for use in protecting data stored in a memory.

12. The non-transitory machine-readable storage medium of claim 11, wherein the communication of the derived security credential over the secure channel uses a

protection code that is useable to verify an integrity of information comprising the derived security credential.

13. The non-transitory machine-readable storage medium of claim 11, wherein the secure channel comprises a trusted interface between the controller and the boot code, the trusted interface settable to an open state to enable communication of selected information between the controller and the program code, and to a closed state to disable communication of the selected information between the controller and the program code.

14. A method comprising:

accessing, by an embedded controller of an electronic device, one-time programmable information;

using, by the embedded controller, the one-time programmable information as a seed in producing a derived security credential;

communicating, by the embedded controller, the derived security credential in a secure manner to a program code executable on a processor of the electronic device; and

protecting, using the derived security credential, data stored in a memory.

15. The method of claim 14, wherein the one-time programmable information is included in the embedded controller or stored in an one-time programmable storage external of the embedded controller.

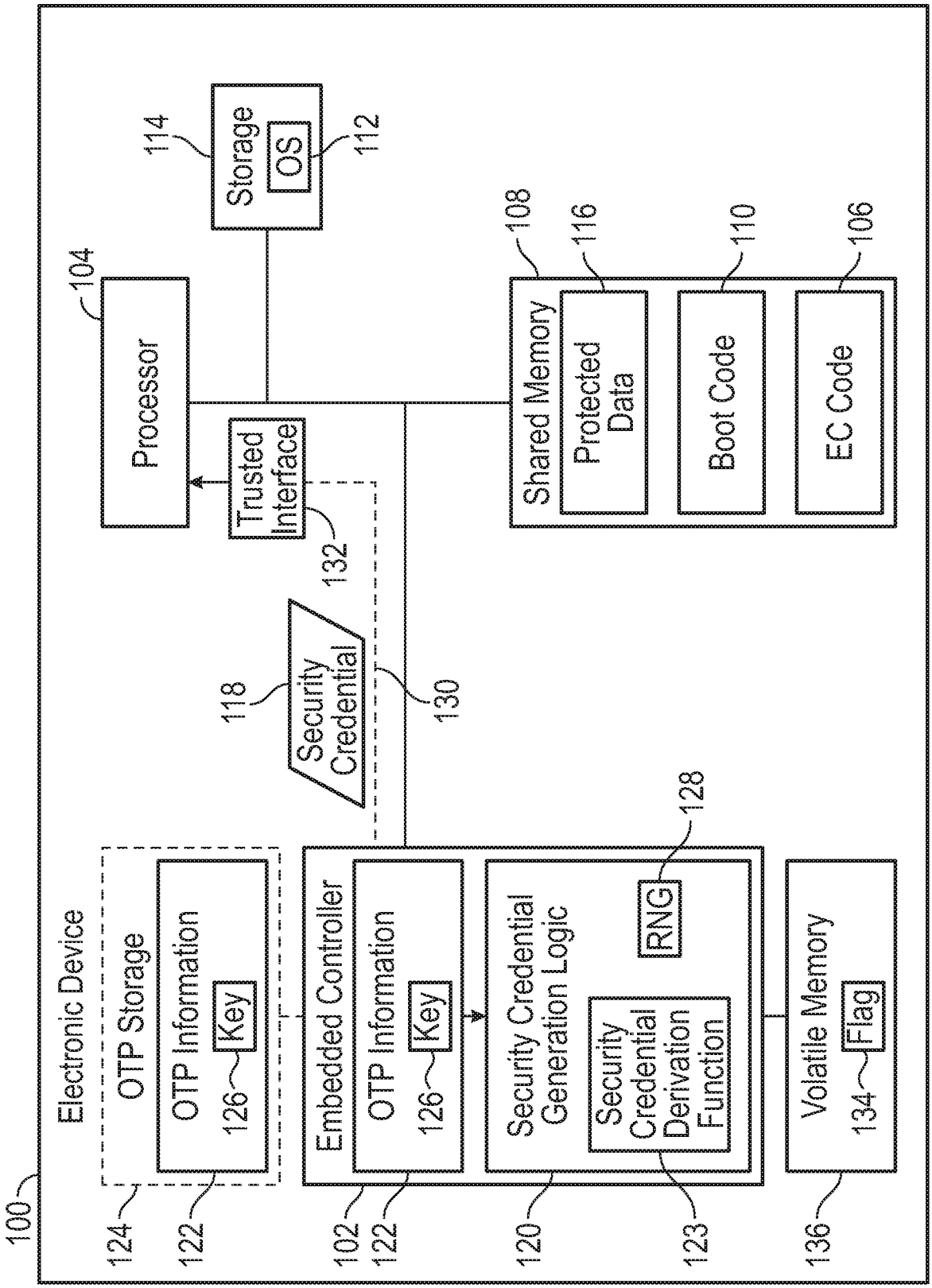
Electronic Device
100
OTP Storage
124
OTP Information
122
Key
126
Embedded Controller
102
OTP Information
122
Key
126
Security Credential Generation Logic
120
Security Credential Derivation Function
123
RNG
128
Volatile Memory
136
Flag
134
Security Credential
118
130
132
Trusted Interface
Processor
104
Storage
114
OS
112
Shared Memory
108
Protected Data
116
Boot Code
110
EC Code
106

FIG. 1

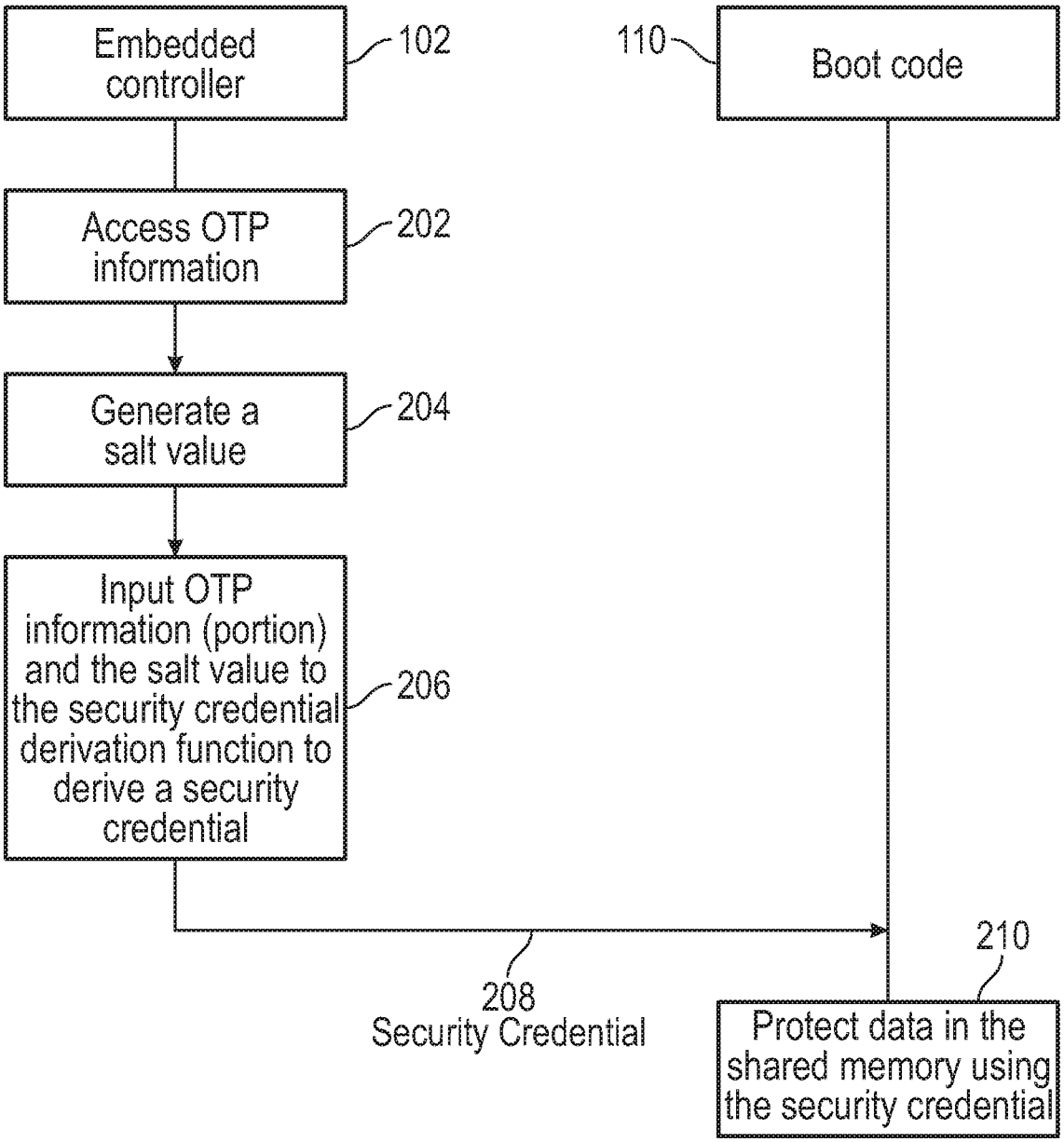
Embedded controller
102
110
Boot code
Access OTP information
202
Generate a salt value
204
Input OTP information (portion) and the salt value to the security credential derivation function to derive a security credential
206
208
Security Credential
210
Protect data in the shared memory using the security credential

FIG. 2

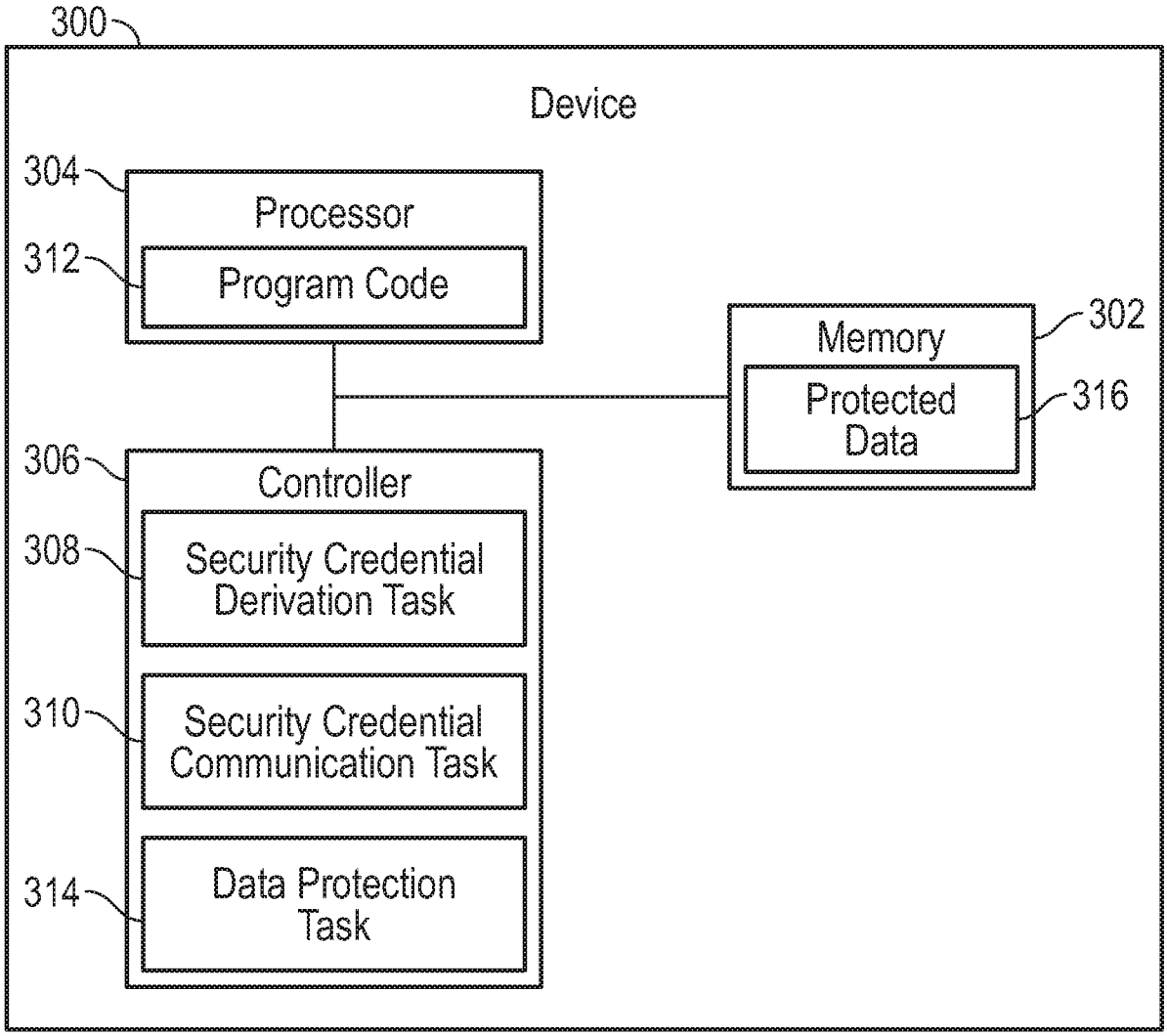
300
Device
304
Processor
312
Program Code
Memory
302
Protected Data
316
306
Controller
308
Security Credential Derivation Task
310
Security Credential Communication Task
314
Data Protection Task

FIG. 3

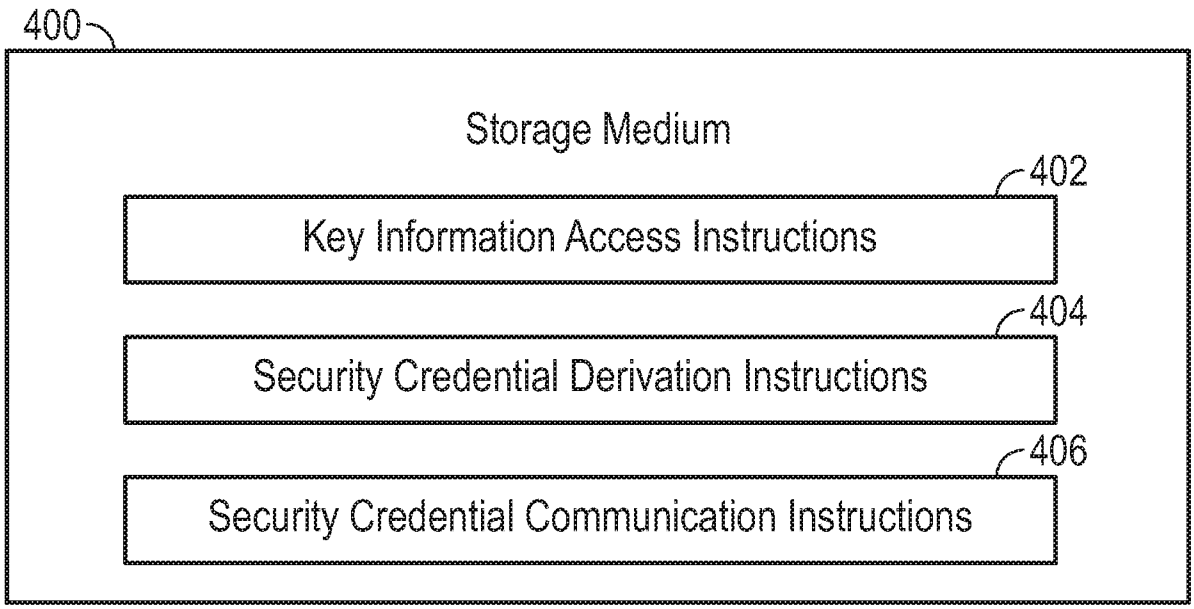
400
Storage Medium
402
Key Information Access Instructions
404
Security Credential Derivation Instructions
406
Security Credential Communication Instructions

FIG. 4

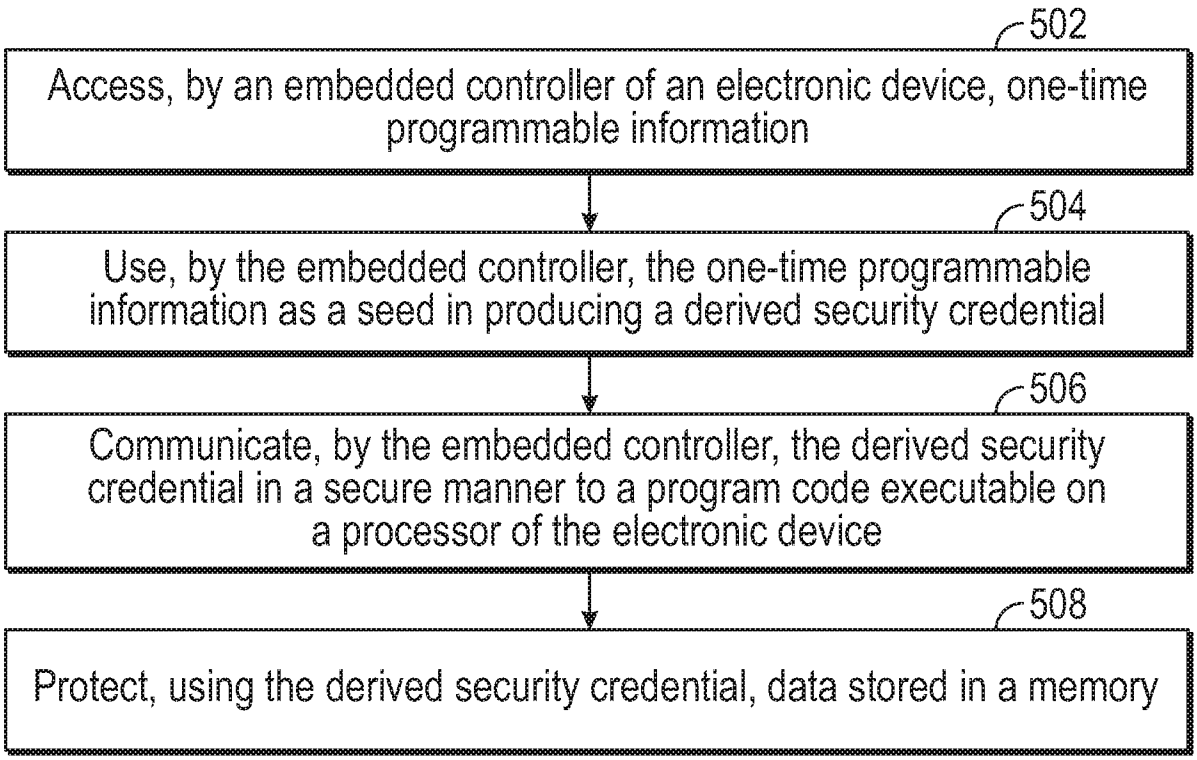
502
Access, by an embedded controller of an electronic device, one-time programmable information
504
Use, by the embedded controller, the one-time programmable information as a seed in producing a derived security credential
506
Communicate, by the embedded controller, the derived security credential in a secure manner to a program code executable on a processor of the electronic device
508
Protect, using the derived security credential, data stored in a memory

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.

PCT/US 2019/016261

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/60 *(2013.01)*
G06F 21/78 *(2013.01)*

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 21/60, 21/78

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)

PatSearch (RUPTO Internal), USPTO, PAJ, Espacenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2016/0055332 A1 (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 25.02.2016, abstract, paragraphs [0008], [0012], [0013], [0019], [0021], [0025], [0044], claim 6	1-15
Y	EP 2034780 A2 (HONEYWELL INTERNATIONAL INC.) 11.03.2009, paragraphs [0012], [0013], [0058], [0066], [0076], [0079]	1-15
Y	US 2018/0279394 A1 (ZTE CORPORATION) 27.09.2018, paragraph [0026]	5, 6
Y	US 2016/0294802 A1 (QUALCOMM INCORPORATED) 06.10.2016, paragraphs [0037], [0038]	9, 10

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

☐ See patent family annex.

* Special categories of cited documents:

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

"E" earlier document 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
31 October 2019 (31.10.2019)	07 November 2019 (07.11.2019)

Name and mailing address of the ISA/RU:
Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer

E. Korolev

Telephone No. 8(495) 531-64-81