Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2694201 C 2017/01/31

(11)(21) **2 694 201**

(12) **BREVET CANADIEN**
CANADIAN PATENT

(13) **C**

(86) **Date de dépôt PCT/PCT Filing Date:** 2008/07/22
(87) **Date publication PCT/PCT Publication Date:** 2009/01/29
(45) **Date de délivrance/Issue Date:** 2017/01/31
(85) **Entrée phase nationale/National Entry:** 2010/01/20
(86) **N° demande PCT/PCT Application No.:** US 2008/070707
(87) **N° publication PCT/PCT Publication No.:** 2009/015116
(30) **Priorité/Priority:** 2007/07/23 (US11/781,412)

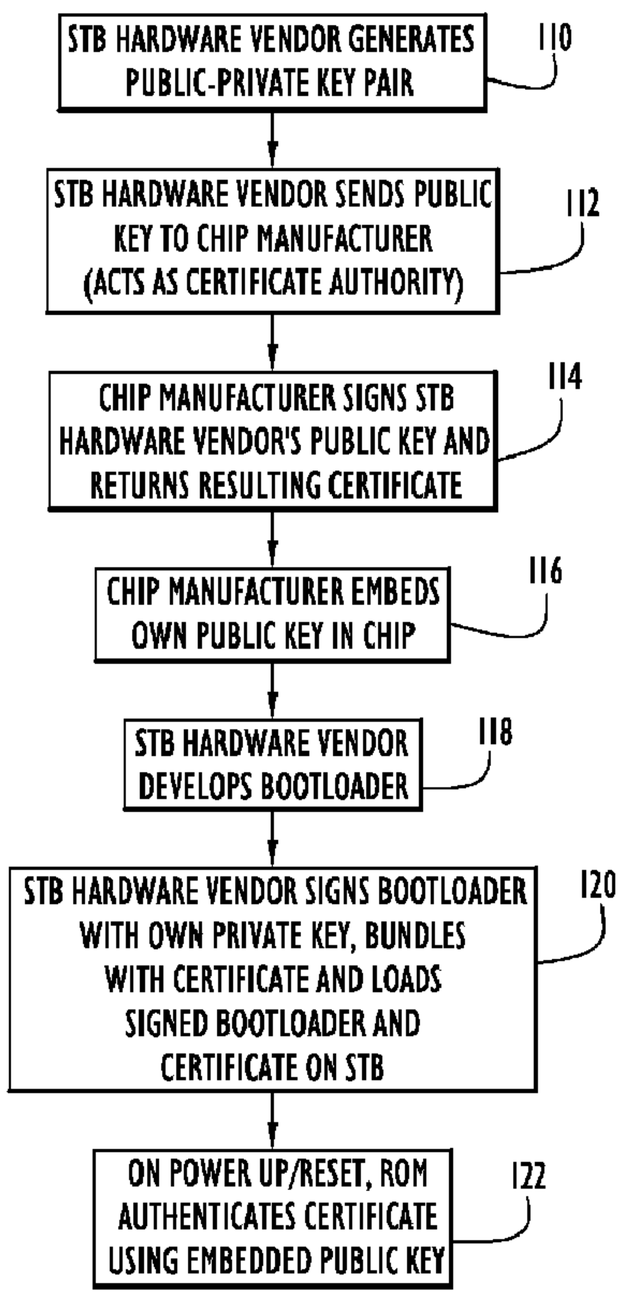
(51) **Cl.Int./Int.Cl.** ***H04N 5/44*** (2011.01),
G06F 21/44 (2013.01), ***G06F 21/57*** (2013.01),
H04L 9/30 (2006.01), ***H04L 9/32*** (2006.01),
H04N 21/40 (2011.01)

(72) **Inventeur/Inventor:**
MURRAY, MARK R., US

(73) **Propriétaire/Owner:**
CISCO TECHNOLOGY, INC., US

(74) **Agent:** RIDOUT & MAYBEE LLP

(54) **Titre : EMPECHER DES MANEUVRES FRAUDULEUSES D'ACTIFS DE BOITIER DE DECODEUR**
(54) **Title: PREVENTING UNAUTHORIZED POACHING OF SET TOP BOX ASSETS**



(57) **Abrégé/Abstract:**
To prevent poaching of an Internet Protocol (IP) set top box (STB) asset or similar network computing device from one system operator to another, code executing in the IP STB not only authenticates downloaded software images using a public key provided

CA 2694201 C 2017/01/31

(11)(21) **2 694 201**

(13) **C**

(57) **Abrégé(suite)/Abstract(continued):**
in a serial-number assigned digital certificate, but also confirms that the serial number appears on a signed whitelist, or does not appear on a signed blacklist. The code exe-cuting in the STB further preferably enforces a rule that only the authority that signed the already-loaded whitelist/blacklist may replace it with a new list. Such a sticky whitelist/blacklist ensures that if the STB boots or resets in a new network, the existing authentication list will not be replaced by a list that is valid for a new or different network, and, as a result, that new software code images will not be authenticated.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 January 2009 (29.01.2009)

PCT

(10) International Publication Number
WO 2009/015116 A1

(51) International Patent Classification:
G06F 21/00 (2006.01) *H04N 7/16* (2006.01)

(21) International Application Number:
PCT/US2008/070707

(22) International Filing Date: 22 July 2008 (22.07.2008)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/781,412 23 July 2007 (23.07.2007) US

(71) Applicant *(for all designated States except US)*: **SCIENTIFIC-ATLANTA, INC.** [US/US]; 5030 Sugarloaf Parkway, Lawrenceville, Georgia 30044 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: **MURRAY, Mark R.** [US/US]; P.O. Box 421637, Atlanta, GA 30342 (US).

(74) Agents: **BARNHARDT, Hubert J., III** et al.; Scientific-Atlanta, Inc., Intellectual Property Dept., 5030 Sugarloaf Parkway, Lawrenceville, Georgia 30044 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, 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, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL,

[Continued on next page]

(54) Title: PREVENTING UNAUTHORIZED POACHING OF SET TOP BOX ASSETS

(57) Abstract: To prevent poaching of an Internet Protocol (IP) set top box (STB) asset or similar network computing device from one system operator to another, code executing in the IP STB not only authenticates downloaded software images using a public key provided in a serial-number assigned digital certificate, but also confirms that the serial number appears on a signed whitelist, or does not appear on a signed blacklist. The code executing in the STB further preferably enforces a rule that only the authority that signed the already-loaded whitelist/blacklist may replace it with a new list. Such a sticky whitelist/blacklist ensures that if the STB boots or resets in a new network, the existing authentication list will not be replaced by a list that is valid for a new or different network, and, as a result, that new software code images will not be authenticated.

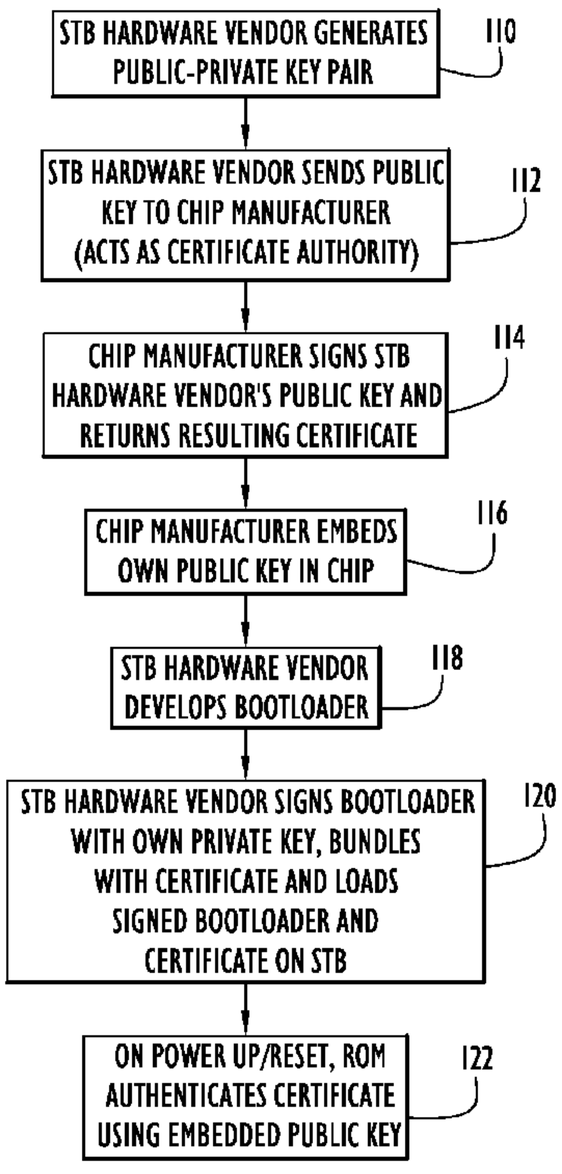


FIG.1

WO 2009/015116 A1

WO 2009/015116 A1

NO, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

PREVENTING UNAUTHORIZED POACHING OF SET TOP BOX ASSETS

TECHNICAL FIELD

[0001] The present disclosure relates generally to methods for providing enhanced control over set top boxes.

BACKGROUND

[0002] System operators, such a cable television service providers or content providers, generally, often provide set top boxes (STBs) to customers. STBs are generally connected between an incoming physical cable, wire, or other broadband connection and a nearby television set (or computer). As is well-known, STBs are conventionally used to, for example, demodulate and unscramble (as necessary) signals for standard television, pay per view, video on demand, gaming data, and other content that is broadcast from a head end of the system operator. The incoming data may be encoded in accordance with, for example, the Internet Protocol (IP) and be compliant with emerging IP television ("IPTV") systems.

[0003] System operators invest significantly in purchasing STBs, and then installing the STBs on customer premises. While a system operator may, over time, recoup the cost of the STBs through subscription fees, it may be months or even years before the cost associated with a given STB and its installation is fully recaptured. It is therefore particularly frustrating for an incumbent system operator when a competing system operator is not only able to convince a given customer to switch service to the competing system operator, but is also able to use (or "poach") the incumbent system operator's STB that is already in place on the customer premises. To the extent the incumbent system operator has not already recaptured the cost of the STB, that cost may be forever lost.

[0004] It is therefore desirable to provide a methodology or technique to better control STB assets belonging to an incumbent system operator.

SUMMARY

[0005] Embodiments of the present invention reduce or eliminate the possibility that a competing system operator that uses Internet Protocol set top boxes supplied from the same hardware vendor can poach the incumbent system operator's IP STBs into the competing system operator's network.

[0006] In accordance with embodiments of the invention, an IP STB preferably includes a code authentication mechanism that is built into an operating system-like program, referred to herein as "Bootloader code" or "the Bootloader." One function of the Bootloader is to ensure that any code executed on the STB is authenticated by a hardware vendor authorized signing key. The Bootloader recognizes a downloadable "whitelist" or "blacklist" that lists serial numbers associated with keys that are permitted (white) or not permitted (black) to be authenticated on the STB. The authentication list may be signed and authenticated by the hardware vendor's authorized signing key.

[0007] As noted, it is possible that an STB may be poached (i.e., transferred without functional impairment) from one system to another. For this to occur, the Bootloader may download a new version of software (properly authorized by the hardware vendor signing keys) and thus allow the transition from one operator network to another. Even if an authentication list is used, a new authentication list could be downloaded and enforced by the STB when it is powered on or is reset in the new network.

[0008] To reduce the possibility of poaching, the Bootloader in the STB preferably enforces a rule that only the authority that signed the current whitelist may replace it with a new whitelist. Such a "sticky whitelist" ensures that if the STB boots or resets in a new network, the existing authentication list will not be replaced by the list valid for the new network, and will not, accordingly, authenticate any code images from that network. This technique assumes that system operators require that their software providers use different signing keys for actual code images.

[0009] In an embodiment, a special hardware vendor-signed message can be used to remove an existing list to facilitate an authorized transition of STB asset from one system operator to another in the event of key loss, asset purchase, corporate merger, or other authorized instance.

[0010] These and other features of the several embodiments of the invention along with their attendant advantages will be more fully appreciated upon a reading of the following detailed description in conjunction with the associated drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] Figure 1 depicts a hardware-enforced authentication mechanism in accordance with embodiments of the present invention.

[0012] Figures 2 and 3 illustrate how a chain of trust is established in accordance with embodiments of the present invention.

[0013] Figure 4 illustrates a key signing process in accordance with embodiments of the present invention.

[0014] Figure 5 illustrates a code signing process in accordance with embodiments of the present invention.

[0015] Figure 6 illustrates an authentication process in accordance with embodiments of the present invention.

[0016] Figure 7 illustrates a list signing process in accordance with embodiments of the present invention.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[0017] An Internet Protocol (IP) set top box (STB) ("IP STB" or, more simply "STB") typically includes basic operating code that is embedded within, e.g., a ROM of an integrated circuit or chip and that is operable to load other, executable, software code into PvAM, such as a flash memory. In the case of an IP STB, an operating system-like routine is typically initially loaded on the STB and operates to load and execute still other software code. The operating system-like code of the IP STB is referred to herein as "Bootloader code," or, more simply, "the Bootloader." STBs that include such

Bootloader code are designed and sold by, e.g., Scientific-Atlanta™ (Lawrenceville, Georgia). These STBs are installed, e.g., by cable television or telephone service providers or the like, on customer premises. Those skilled in the art will appreciate that

the embodiments described herein are applicable to any "provider" that delivers content to a STB, or similar device.

[0018] In addition to the embedded operating code and the Bootloader code, service providers may also load other software on the IP STB to, for example, interact with a headend of the service provider, among other things. This additional software may be the service provider's own code, or more frequently, code generated by a third party. If the STBs are already in the field, the additional software may be delivered to the STBs using a broadcast technique. Notably, in allowing third party software to be loaded on STBs, there is the potential that corrupted, or worse, malicious software may be proliferated.

[0019] To ensure that only authorized software is loaded and executed on the STB, a software code image authentication mechanism may be implemented using a public-private key pair scheme along with digital certificates. Initially, in a preferred implementation, a root chain-of-trust on the set top box is established by a hardware-enforced mechanism that begins by authenticating the Bootloader itself before any other code execution is permitted.

[0020] Figure 1 illustrates a series of steps for implementing the hardware-enforced authentication mechanism. At step **110**, a STB hardware vendor generates a public-private key pair, referred to as the "Bootloader Signing Key," and then at step **112**, sends the public key portion thereof to, e.g., a chip manufacturer whose chips are to be incorporated in the STB asset. As those skilled in the art will appreciate, the chip manufacturer may then function as a "certificate authority" whereby, at step **114**, the chip manufacturer signs the public key portion with its own chip manufacturer private key and returns a resulting digital certificate to the STB hardware vendor.

[0021] At step **116**, in the course of manufacturing the integrated circuit (i.e., chip) for the STB, the manufacturer embeds its own public key in the circuit hardware. Meanwhile, at step **118**, the STB hardware vendor develops the Bootloader code and at step **120** signs the code with its own private key, and then bundles the signed Bootloader code with the digital certificate previously received from the chip manufacturer, and loads these components in the flash memory or RAM of the STB. At step **122**, when the STB thereafter powers up or is reset, the chip manufacturer's public key that is embedded

in the chip is used to authenticate the digital certificate that is bundled with the Bootloader code, thereby authenticating the public key portion of the Bootloader signing key. The Bootloader code can then be properly authenticated with that public key. The procedure described above establishes a first link in a chain-of-trust, where the first link is considered to be a hardware-based authentication link.

[0022] Stated alternatively, if the Bootloader is executing on the STB, then the hardware-enforced security mechanism must have "passed" and the Bootloader is thereafter "trusted." As will be explained in more detail below, a further chain-of-trust thereafter flows from this first hardware-enforced authentication mechanism to different keys, certificates and code.

[0023] Figure 2 depicts a series of steps that illustrate a chain of trust for authenticating different software components, lists and keys that are intended to be run or used on the STB. Step **202** represents the hardware enforced authentication that is based on the STB hardware vendor's Bootloader Signing Key and that was described with respect to Figure 1. The chain-of-trust may then be extended to other authorities through a series of key signing processes. In an embodiment, and as depicted by step **204**, the STB hardware vendor itself may act as a certificate authority by signing a third Party Signing Key (i.e., signing a third party's public key) using a Key Signing Key ("KSK") and returning a digital certificate, referred to as a "Key Signing Key Certificate" or "KSK Certificate," to an authorized third party. In accordance with an embodiment, each such KSK Certificate includes an embedded serial number that is associated with the third party. As will be explained more fully below, the KSK may also be used to sign a List Signing Key (LSK) that itself is used to authenticate a list of serial numbers.

[0024] An authorized third party may then sign its own software code (or image) using its own Third Party Signing Key and embed the KSK Certificate in the resulting software image, as shown by step **206**. At run-time, as represented by step **208**, the Bootloader extends trust to the software image by first authenticating the embedded KSK Certificate, and then using the public key from the certificate to authenticate the software image itself. The Bootloader may also, in a similar fashion, authenticate a whitelist or blacklist. Details of such a process are described later herein.

[0025] Figures 3-7 depict in more detail the several signing and authentication processes, and practical effects thereof, in accordance with embodiments of the invention.

[0026] As shown in Figure 3, a KSK public key **312a** is compiled as part of the Bootloader and is used to establish the authenticity of the third party public key **310a**, which is treated as data at this point. If the key signature **314** (which has been encrypted with a KSK private key **312b**) passes, then the third party public key can be trusted as authentic (since it was signed by the private KSK). The third party public key **310a** and KSK Signature **314** are elements of the KSK Certificate **404** (described below).

[0027] The third party now-"trusted" public key **310a** is then used to establish the authenticity of a code image **318**. If a code signature **320** (which has been encrypted using the third party's private key **310b**) passes, then trust can be extended to the code image **318** (since it was signed by the third party private key, whose public key portion has been authenticated).

[0028] Key Signing Process

[0029] As shown by Figure 4, a key signing process **400** is employed to sign the third party public key **310a** (its private key **312b** being held in confidence) and a serial number **402** using the KSK private key **312b** (recalling that the KSK public key **312a** is compiled in the Bootloader). The resulting output file is a "KSK Certificate" **404**, which includes a copy of the third party public key **310a**, the serial number **402**, and a signature of the data **314**. Those skilled in the art will appreciate that there may be additional header fields associated with KSK Certificate **404** to ensure file corruption can be detected before any "bad" data is used.

[0030] The KSK Certificate **404** is provided to the third party and is used in subsequent code-signing steps, as explained below. The KSK Certificate need not be kept secret.

[0031] Code Signing Process

[0032] When a third party has been given a serial number-controlled KSK Certificate **404**, the third party then has the ability to sign its own code images, as shown by the process **500** depicted in Figure 5. Specifically, software code (or simply "code") **318** is signed by the third party's private key **310b** and the data from the KSK Certificate **404** is

copied into the output image file **510**. The third party private key **310b** is used to generate a signature **320** for the code block. The code **318**, KSK Certificate data **404**, and code signature **320** are all packaged into a single image file **510** along with additional header bytes (not shown) that can be used to detect file corruption that can occur in transport. This signed code image file **510** can then be provided to a Download Server for broadcasting and loading into, e.g., flash memory on STBs.

[0033] Code Validation Process

[0034] As shown in Figure 6, the Bootloader authenticates the signed code image file **510** immediately after download, before committing it to memory, or allowing it to execute on the STB, or combinations thereof. More specifically, the Bootloader uses the code header data along with the public key of the KSK, compiled-in the Bootloader, to authenticate the code image. This process was described in connection with Figure 3.

[0035] As will be explained in more detail below, an optional blacklist/whitelist listing unique serial numbers of KSK Certificates can be used to reject authority for any particular serial number that is associated with a KSK Certificate, thereby rejecting the authority of a given third party vendor to have its code loaded on the STB.

[0036] List Signing Process

[0037] Figure 7 shows how signing authority can be revoked/blacklisted (or expressly given/whitelisted) using a whitelist/blacklist mechanism. As shown, a list of serial numbers (previously assigned to respective third party vendors) in a whitelist or blacklist **701** is signed by process **700** using a private key portion **710b** of a List Signing Key ("LSK"), resulting in a signed whitelist/blacklist **702** that is then downloaded and stored in the memory of the STB for reference by the Bootloader. Figure 2 shows how the LSK is used to authenticate a whitelist/blacklist.

[0038] Referring back to Figure 6, when signed code image file **510** is received at the STB, the Bootloader not only authenticates the code image, but also may consult the signed whitelist/blacklist **702** to determine if a serial number on that list corresponds to the serial number **402** of the code image **510**. If the serial number is listed in a whitelist, then the Bootloader will allow the code to be installed and/or executed, otherwise it may

be rejected. If the serial number is listed on a blacklist, then the Bootloader will reject the code image **510**, otherwise it may be accepted.

[0039] Whitelist/Blacklist Management

[0040] The whitelist or blacklist **701** comprising serial numbers that can be used to revoke signing authority to particular third party keys must be available to the Bootloader at the time a given code image is authenticated. Consequently, the list may be compiled-in the Bootloader or downloaded any time before code is downloaded. A compiled-in list is secure in that it cannot be tampered with because it is part of the hardware enforced authentication mechanism.

[0041] If the list is downloaded, the LSK private key used for signing the list ensures the content is not altered. On the other hand, the download mechanism or flash memory storage table could possibly be interfered with. For example, if the list were prevented from being loaded at all into flash memory, then the Bootloader would not know to reject a given serial number that was listed on the list that was blocked. Thus, a downloaded blacklist is a "best effort" method of authority revocation. A compiled-in list may be considered more secure.

[0042] Another approach, but still possibly vulnerable, is to require all serial numbers to be whitelisted. Thus, if the external list is removed or tampered with then authority can only be lost and not gained. However, if authority were granted to a given serial number in one version of the file and later revoked in a later version of the file, authority could be restored by re-flashing the original list back into flash memory.

[0043] List Control

[0044] In an example implementation, the STB recognizes a whitelist on a STB Download server associated with the headend of the system provider. As explained above, upon download this list is authenticated using the same third party trust mechanism that is used to authenticate software images. In accordance with a particular embodiment of the invention, after a whitelist is loaded into the STB, a new list only from the same signing authority (serial number) can replace the earlier-downloaded list. As mentioned earlier, the initial whitelist that is loaded into a STB can be put in place

explicitly before STB delivery, or may be loaded on first-boot in the host network. In one possible implementation, when an STB boots without any whitelist in place, then any STB-authorized Third Party Certificate may be accepted. However, once a whitelist has been received and stored, only the third party key serial number for the LSK used to sign the stored list will be recognized for list replacement.

[0045] White List Removal

[0046] A List Maintenance Code Signing Key ("LMCSK") may always be authorized even if not included in the whitelist. More specifically, a utility may be provided to account for when it is necessary to remove the whitelist in, for example, a customer repair facility in order to allow diagnostic software to be loaded or to allow for repaired STBs to be re-deployed in a new network. For example, a "Whitelist Delete Application" may be created and signed by the LMCSK. Once the application is executing, it may contact a secure server on a local network to identify the STB and receive permission to remove the whitelist. The messages exchanged ensure mutual authentication and generate a delete request for that single STB. In a preferred implementation, only authorized repair centers will have such a secure server. Alternatively, the server may be operated by a third party, e.g., the STB hardware vendor, and the authentication may be performed over a network. The LMCSK is the same as a 3rd party Code Signing Key except that it has a special or unique serial number recognized by the Bootloader.

[0047] On occasion, it may also be desirable to remove the whitelist of a large population of STBs in the field to facilitate the authorized acquisition of assets containing conflicting whitelists or to recover from a lost LSK. For example, one service provider may legitimately acquire the customers and assets of another service provider and thus acquire all of the STBs of the other service provider.

[0048] In this scenario, a "WhiteList Delete Message" may be signed by the List Removal Key ("LRMK") and be placed on the STB Download Server. The message preferably contains the list of the LSK serial numbers targeted for removal. For security purposes, after the use of such a WhiteList Delete Message the targeted LSK Keys mentioned in the message should no longer be used. The LRMK is the same as a LSK except that it has a special or unique serial number recognized by the Bootloader.

[0049] To summarize, when no whitelist or blacklist is loaded in an STB, any authentic third party certificate can be used to download Software or whitelists/blacklists. When either list is already loaded in an STB, it is preferable that only the third party certificate that signed the saved list can be used to replace it. Further, in accordance with a preferred embodiment, only third party certificate serial numbers listed in the whitelist/blacklist may be accepted for authentication.

[0050] There is also a scenario where a repair facility may obtain STBs from several different system operators, where the STBs would have different whitelists/blacklists stored. It may be desirable for the repair facility to restore the STBs back to their "factory-fresh" state (i.e. no whitelist/blacklist at all). However, it would be unlikely that all operators could be convinced to white-list the serial number of the repair software (and, in fact, from a security perspective, it would be prudent if they excluded it). Further, if the STB hardware vendor had a master-key to remove the lists (or any software for that matter), that would represent a security vulnerability. To address these issues, embodiments of the present invention provide the following method to remove whitelists/blacklists at a repair facility.

[0051] In accordance with an embodiment, the list-removal software application, before it will actually remove the list, communicates with a PC-based "Authentication Server" to get permission. The Authentication Server can be located anywhere physically, e.g., at the repair facility or at a central location.

[0052] The server itself may include complex rules regarding which serial numbers are permitted. An audit trail of the server's activities is preferably kept. The server also preferably has a private-key token card of some sort that can be used to encrypt or sign data. The practical effect of the token is that "private key" encryption allows a client (i.e., the STB in this case) to authenticate that the server it is talking to has a valid private key by using the compiled-in public key to authenticate it. Private keys are presumed to be tightly controlled, never exposed, and mathematically impractical to reverse-engineer. Accordingly, if there is evidence that an entity correctly performed an encryption using the private key, that entity can be trusted.

[0053] The removal software application running in the STB may generate a random block of data, send it to the authentication server along with serial number information and challenge it to return a valid response. The Authentication server then uses the private key to encrypt (or sign) the random data and send it back to the STB software. The STB software uses the public key to decrypt (or signature check) the response to confirm that whomever it is talking to does indeed have the private key. The random data prevents a "record-and-playback" attack - ensuring every exchange will be unique.

[0054] Once the STB software authenticates, it can communicate with the Authorization Server and, with permission, it will then execute the function to remove the list(s) from secure storage.

[0055] A goal of the sequence described above is to make sure that the list-removal software application in the STB that is signed with the LMCSK (always white-listed) becomes inert in the absence of the authentication server. In other words, if someone at a repair facility were to steal the software signed with the LMCSK, it would be of no use without the Authentication Server. The Authentication server is preferably secured in a locked room and may be local or at an offsite location.

[0056] In one implementation, a STB hardware vendor-Signed Delete Message may be used to remove a specific whitelist, and a STB hardware vendor-Signed Delete Application may be used to remove individual whitelists in a repair facility with a secure server.

What is claimed is:

1. A method for maintaining control over a set-top-box (STB) asset, comprising: authenticating, using a hardware-based authentication process, a first software application, the first software application having been generated by a first party, wherein the STB asset is operated by a second party; once authenticated, using the first software application to authenticate a public key received with a second software application that was generated by a third party and is intended to be run on the STB asset; once authenticated, using the public key of the third party to authenticate the second software application generated by the third party; and confirming that the second software application is authorized to be run on the STB asset by consulting a list of authorized third parties, wherein consulting the list of authorized third parties comprise utilizing a list maintenance code signing key (LMCSK) having a unique serial number always recognized by the hardware-based authentication process, and locating the third party that generated the second software application, wherein the list of authorized third parties is itself authenticated by the first software application using the LMCSK.

2. The method of claim 1, wherein the list comprises a plurality of serial numbers associated with the third parties.

3. The method of claim 1, further comprising restricting the ability to replace the list.

4. The method of claim 3, further comprising allowing replacement of the list only by a same signing authority that signed the list.

5. The method of claim 1, wherein said hardware-based authentication process comprises authenticating a digital certificate using a public key embedded in hardware.

6. The method of claim 1, wherein the list is compiled in the STB asset.

7. The method of claim 1, wherein the list is downloaded from a server.

8. The method of claim 1, wherein the list is digitally signed with a private key belonging to the second party.

9. The method of claim 8, further comprising authenticating the digitally signed list using a public key compiled in the first software application.

10. The method of claim 8, wherein the first party is the asset hardware vendor.

11. The method of claim 8, wherein the second party is a content service provider.

12. The method of claim 11, wherein the second party is an IPTV service provider.

13. A method, comprising: storing a first digitally signed list on an electronic device, the first digitally signed list including identification data for a plurality of software vendors authorized to load and run software on the electronic device; receiving, at the electronic device, a second digitally signed list, the second digitally signed list also including identification data for a plurality of software vendors authorized to load and run software on the electronic device, wherein the identification data comprises a list maintenance code signing key (LMCSK) having a serial number unique to a signing authority that is always recognized by a bootloader; determining that a second signing authority that signed the second digitally signed list is the same as a first signing authority that signed the first digitally signed list, wherein determining that the second signing authority that signed the second digitally signed list is the same as the first signing authority that signed the first digitally signed list comprises authorizing the LMCSK; and replacing, utilizing the bootloader, the first digitally signed list with the second digitally signed list when it is determined that the second signing authority that signed the second digitally signed list is the same as the first signing authority that signed the first digitally signed list.

14. The method of claim 13, wherein the electronic device comprises a set top box.

15. The method of claim 13, wherein the first and second digitally signed lists are whitelists.

16. The method of claim 13, wherein said replacing is performed, at least in part, by code stored on a computer readable memory as instructions executable by a computer processor that has been authenticated via the bootloader.

17. A method comprising: receiving at a set top box a signed list of serial numbers, wherein the serial numbers respectively identify authorized digital certificates, and a digital certificate includes a public key associated with a developer of a code image; storing the list of serial numbers in a memory of the set top box; receiving at the set top box a signed replacement list of serial numbers and a list maintenance code signing key (LMCSK) having a serial number unique to a signing authority that is always recognized by a bootloader; determining if a first signing authority that signed the signed replacement list of serial numbers is the same as a second signing authority that signed the list of serial numbers; and replacing, utilizing the bootloader, the list of serial numbers with the list of serial numbers from the signed replacement list of serial numbers when the first signing authority that signed the signed replacement list of serial numbers is the same as the second signing authority that signed the list of serial numbers.

18. The method of claim 17, wherein the same signing authority is a system operator for the set top box.

19. The method of claim 17, wherein said replacing is performed, at least in part, by code stored on a computer readable memory as instructions executable by a computer processor that has been authenticated via a hardware-enforced authentication mechanism.

20. A set-top-box (STB), comprising: a public key embedded in a hardware device in the STB and code stored in a memory as instructions executable by a computer processor of the STB to:

authenticate, based at least in part on the public key embedded in the hardware device, a stored list of authorized software vendors and a list maintenance code signing key (LMCSK) having a serial number unique to a signing authority that is always recognized by the hardware device, a software image downloaded from a server and prepared by one of a plurality of software vendors; and

replace the stored list of authorized software vendors only when a new list of authorized software vendors is signed by the same signing authority that signed the stored list of authorized software vendors and recognized by the LMCSK.

21. The set-top-box of claim 20, wherein the set-top-box communicates over an internet protocol (IP) network.

22. The set-top-box of claim 20, wherein the set-top-box is operable with an internet protocol (IP) television (IPTV) system.

23. The set-top-box of claim 20, wherein the code includes instructions to receive the new list of authorized software vendors from a download server.

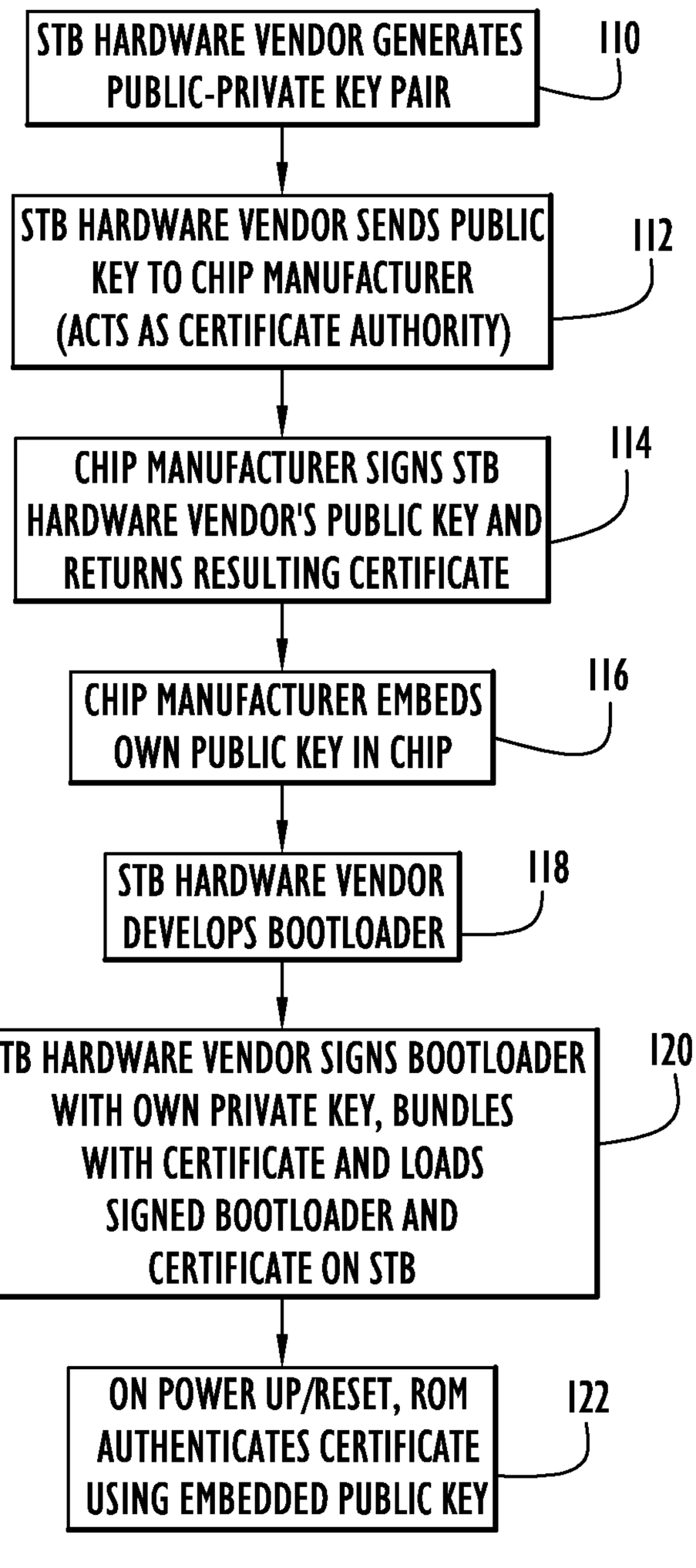

STB HARDWARE VENDOR GENERATES PUBLIC-PRIVATE KEY PAIR
110
STB HARDWARE VENDOR SENDS PUBLIC KEY TO CHIP MANUFACTURER (ACTS AS CERTIFICATE AUTHORITY)
112
CHIP MANUFACTURER SIGNS STB HARDWARE VENDOR'S PUBLIC KEY AND RETURNS RESULTING CERTIFICATE
114
CHIP MANUFACTURER EMBEDS OWN PUBLIC KEY IN CHIP
116
STB HARDWARE VENDOR DEVELOPS BOOTLOADER
118
STB HARDWARE VENDOR SIGNS BOOTLOADER WITH OWN PRIVATE KEY, BUNDLES WITH CERTIFICATE AND LOADS SIGNED BOOTLOADER AND CERTIFICATE ON STB
120
ON POWER UP/RESET, ROM AUTHENTICATES CERTIFICATE USING EMBEDDED PUBLIC KEY
122

FIG.1

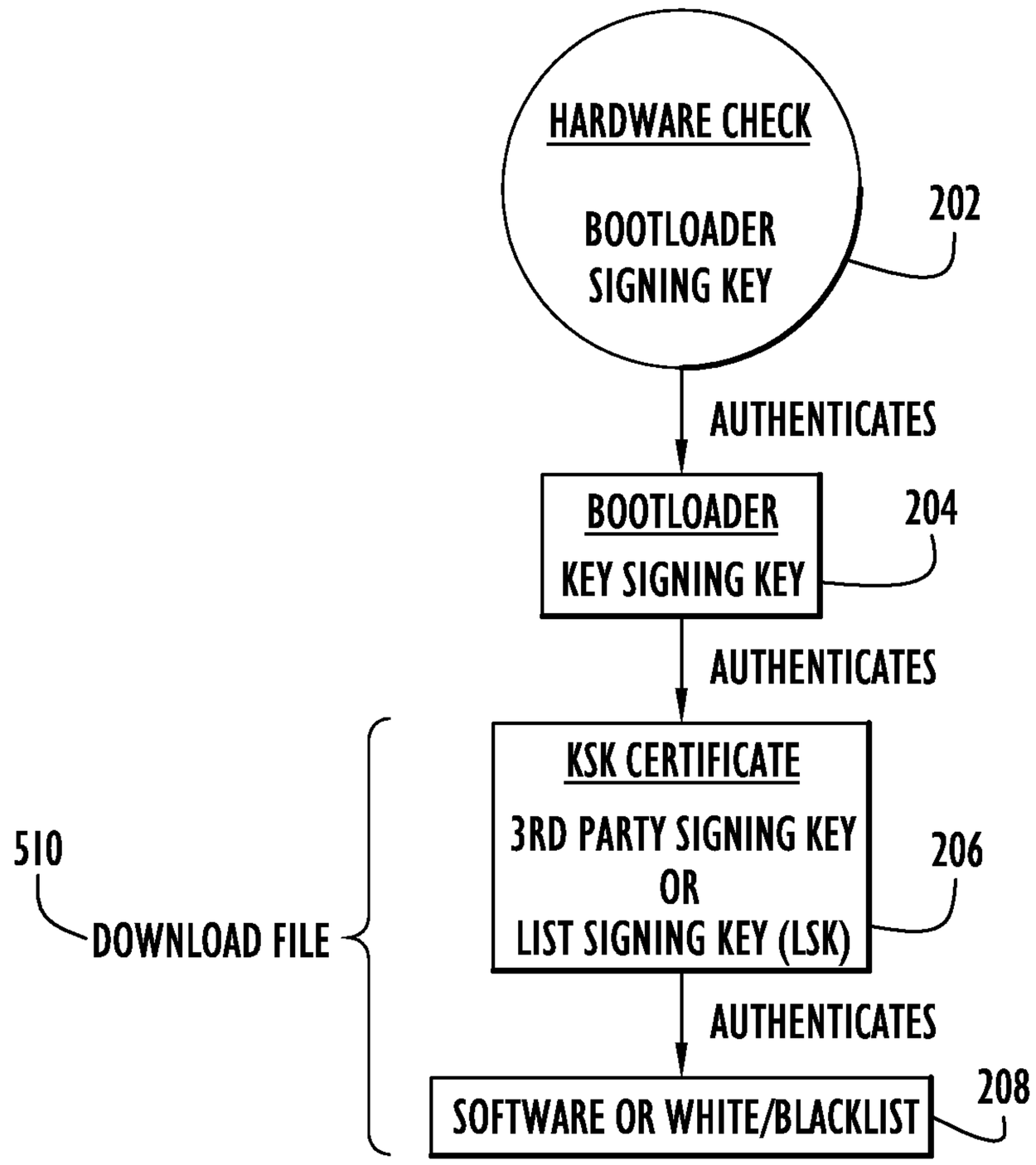

HARDWARE CHECK
BOOTLOADER
SIGNING KEY
202
AUTHENTICATES
BOOTLOADER
KEY SIGNING KEY
204
AUTHENTICATES
KSK CERTIFICATE
3RD PARTY SIGNING KEY
OR
LIST SIGNING KEY (LSK)
206
AUTHENTICATES
SOFTWARE OR WHITE/BLACKLIST
208
510
DOWNLOAD FILE

FIG.2

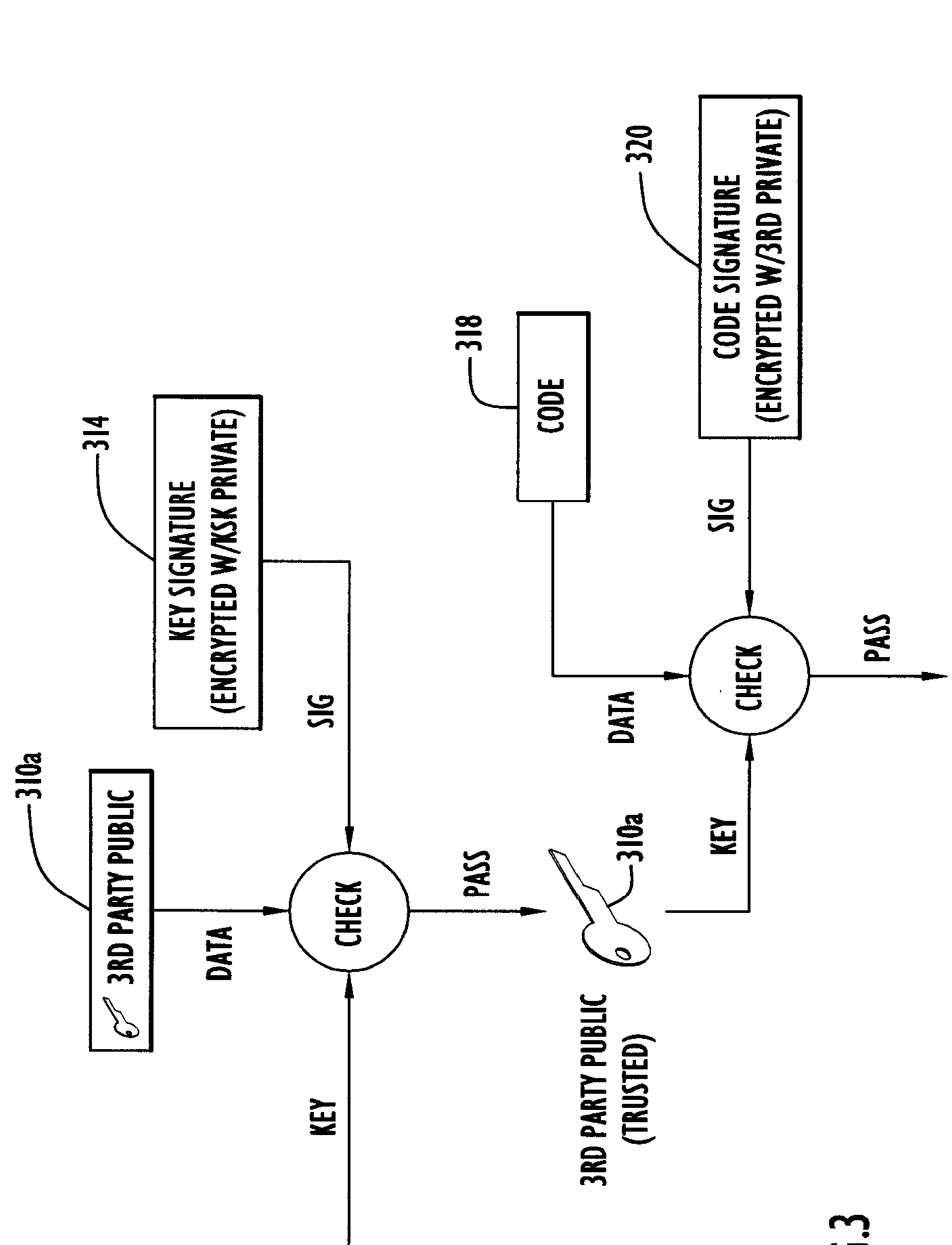

KSK PUBLIC (COMPILED-IN BOOTLOADER)
312a
KEY
3RD PARTY PUBLIC (TRUSTED)
310a
KEY
DATA
CHECK
PASS
3RD PARTY PUBLIC
310a
DATA
CHECK
PASS
KEY SIG (ENCRYPTED W/KSK PRIVATE)
314
SIG
CODE
318
CODE SIGNATURE (ENCRYPTED W/3RD PRIVATE)
320
SIG

FIG.3

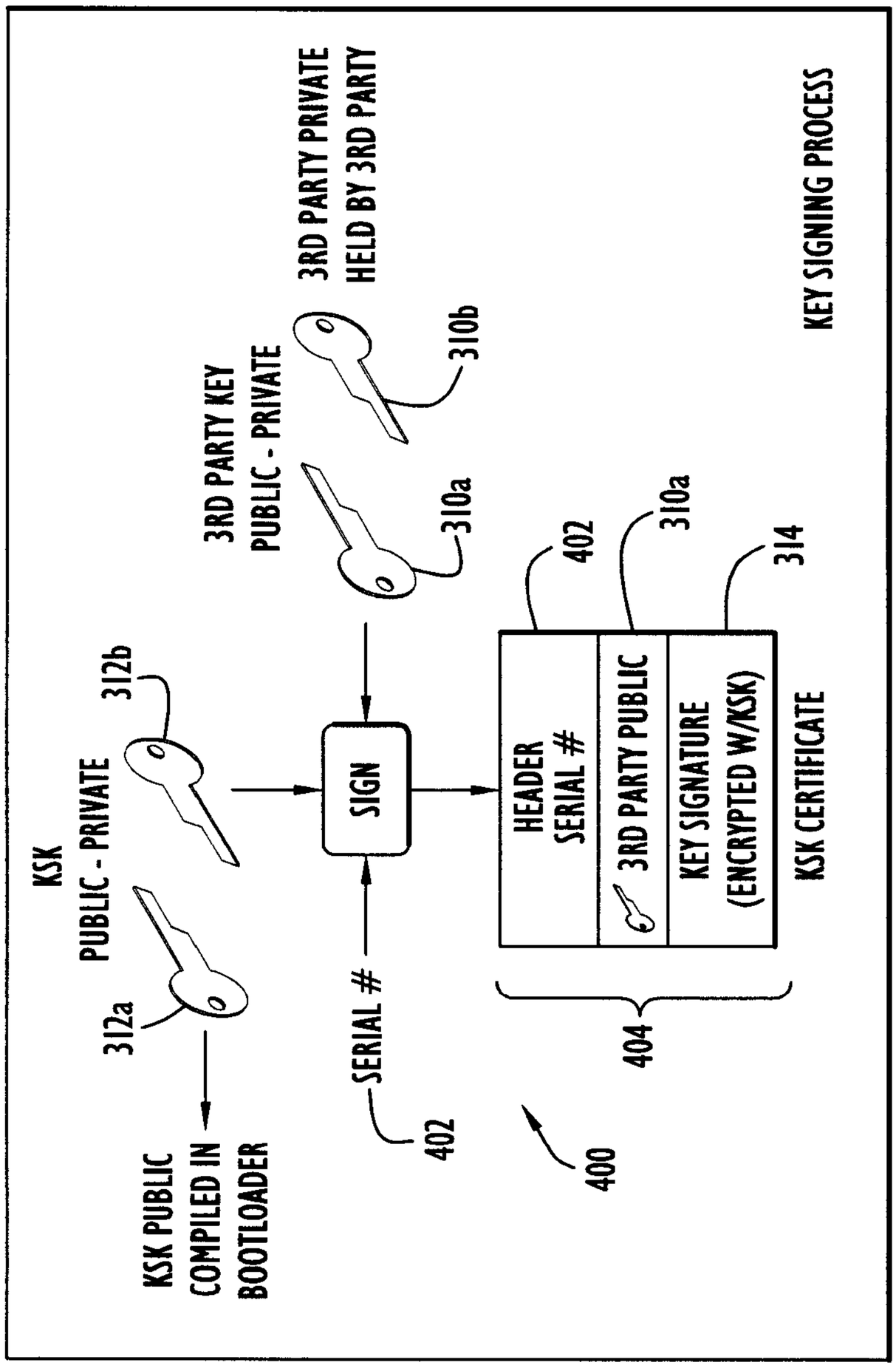

KEY SIGNING PROCESS
3RD PARTY PRIVATE
HELD BY 3RD PARTY
310b
3RD PARTY KEY
PUBLIC - PRIVATE
310a
KSK
PUBLIC - PRIVATE
312b
312a
KSK PUBLIC
COMPILED IN
BOOTLOADER
SIGN
SERIAL #
402
HEADER
SERIAL #
402
3RD PARTY PUBLIC
310a
KEY SIGNATURE
(ENCRYPTED W/KSK)
314
KSK CERTIFICATE
404
400

FIG.4

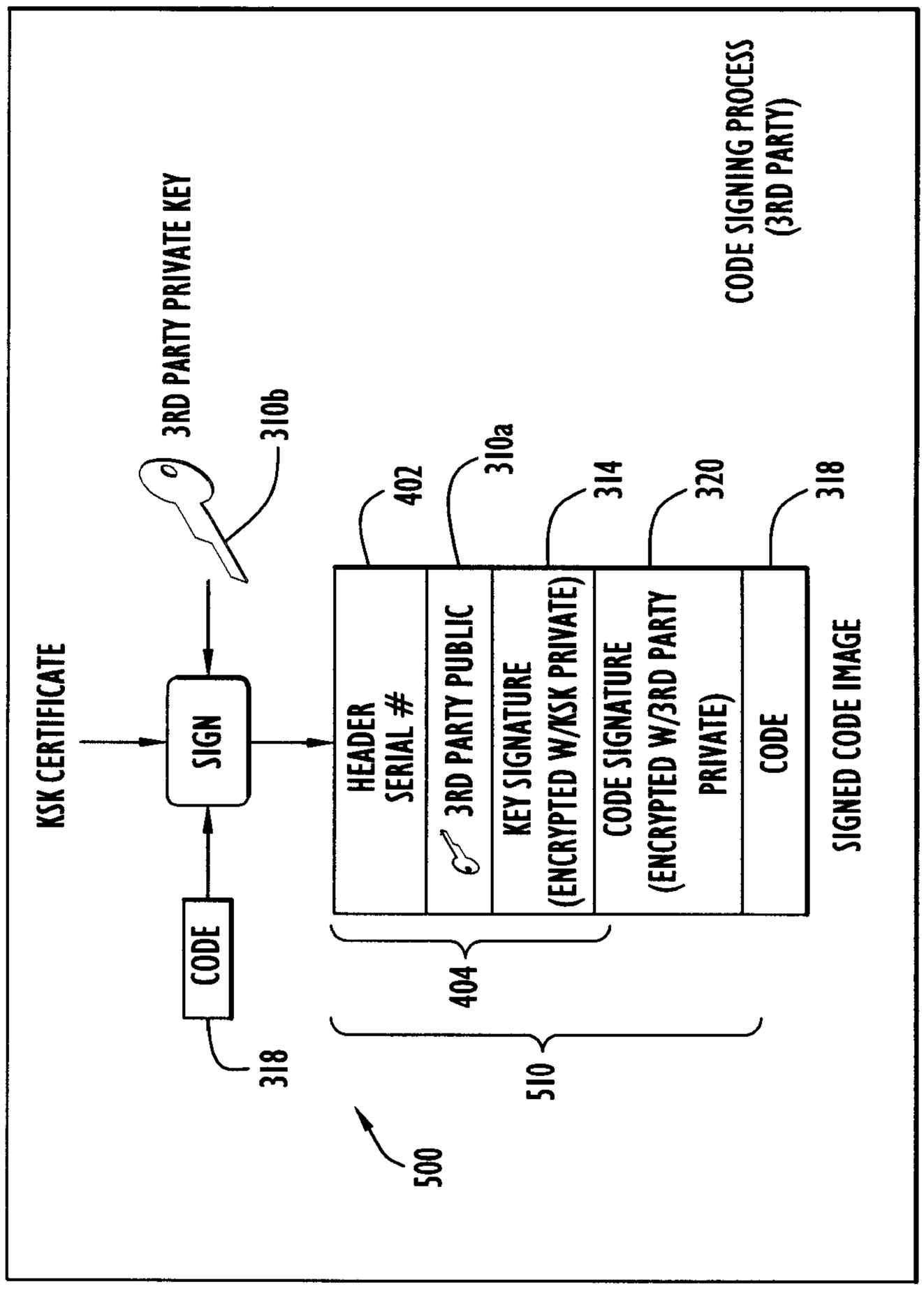

KSK CERTIFICATE
SIGN
CODE
318
3RD PARTY PRIVATE KEY
310b
HEADER
SERIAL #
402
3RD PARTY PUBLIC
310a
KEY SIGNATURE
(ENCRYPTED W/KSK PRIVATE)
314
CODE SIGNATURE
(ENCRYPTED W/3RD PARTY
PRIVATE)
320
CODE
318
SIGNED CODE IMAGE
404
510
CODE SIGNING PROCESS
(3RD PARTY)
500

FIG.5

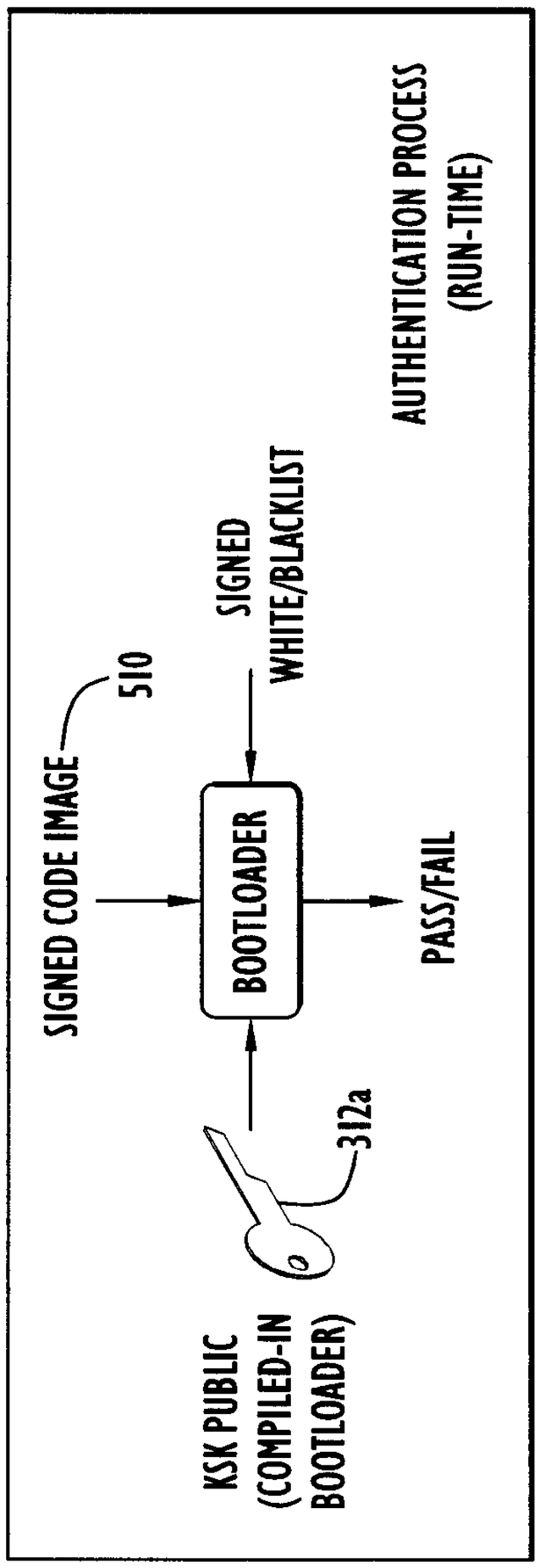

AUTHENTICATION PROCESS
(RUN-TIME)
SIGNED CODE IMAGE
510
SIGNED
WHITE/BLACKLIST
BOOTLOADER
PASS/FAIL
312a
KSK PUBLIC
(COMPILED-IN
BOOTLOADER)

FIG.6

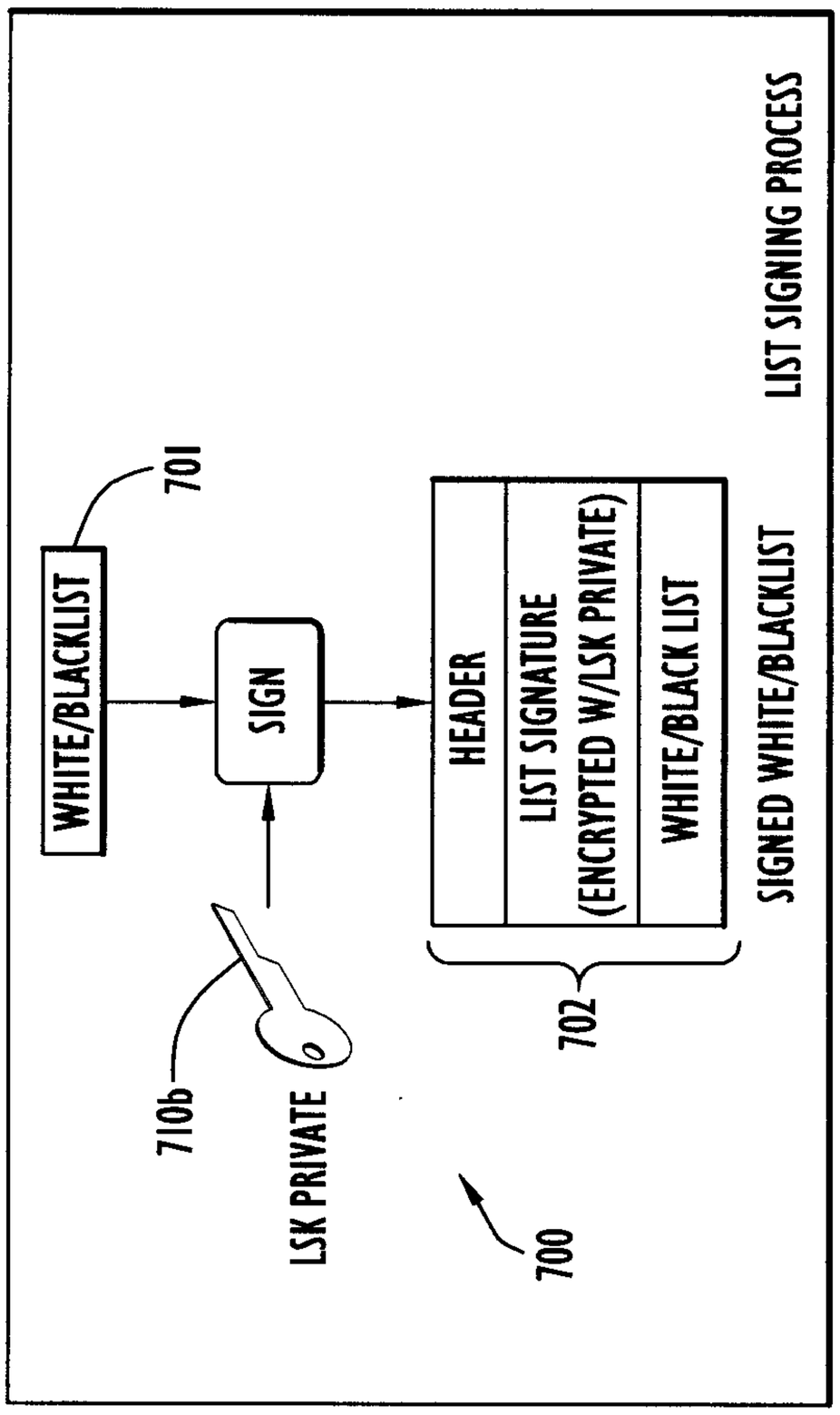
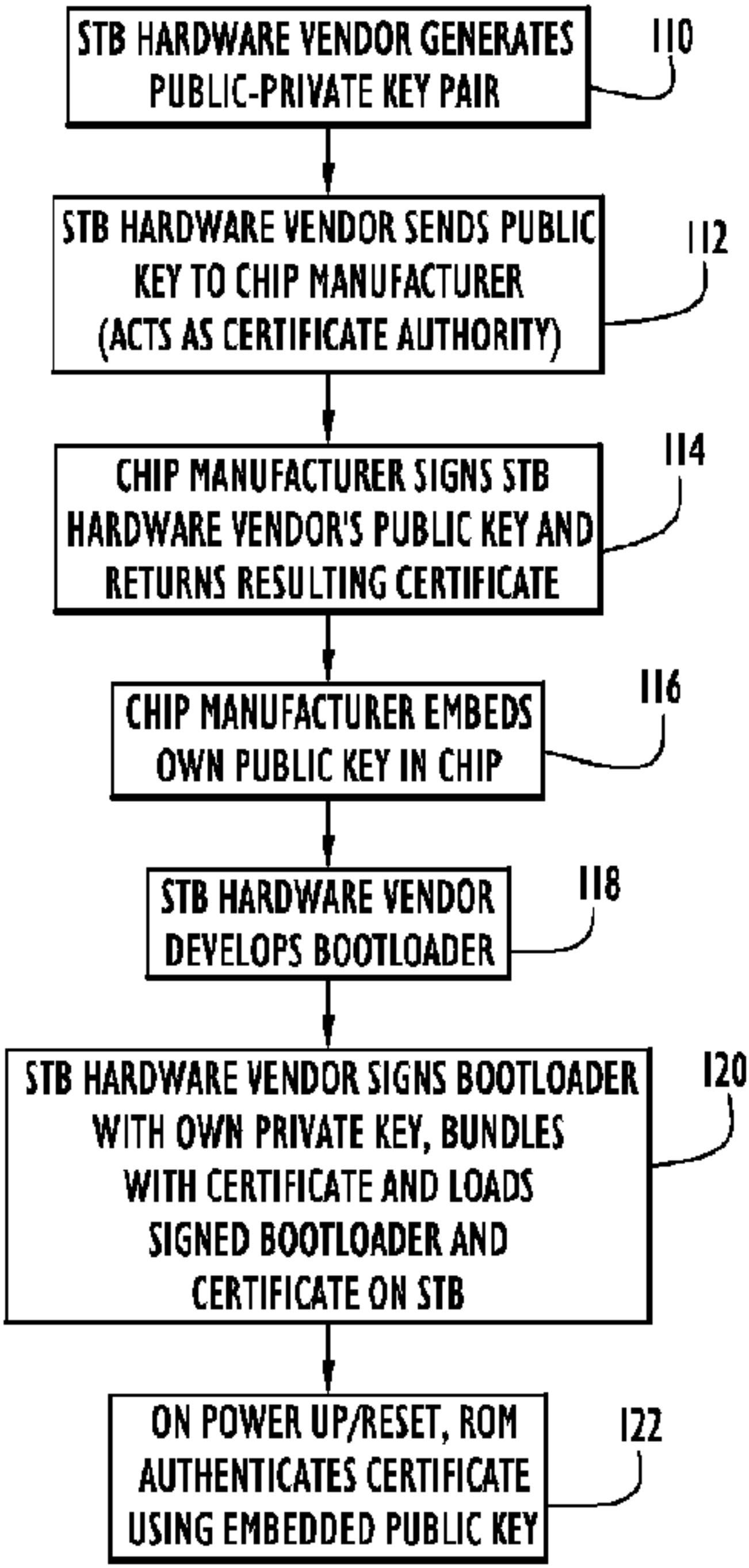

LIST SIGNING PROCESS
701
WHITE/BLACKLIST
SIGN
HEADER
LIST SIGNATURE
(ENCRYPTED W/LSK PRIVATE)
WHITE/BLACK LIST
SIGNED WHITE/BLACKLIST
710b
LSK PRIVATE
702
700

FIG.7


STB HARDWARE VENDOR GENERATES PUBLIC-PRIVATE KEY PAIR
110
STB HARDWARE VENDOR SENDS PUBLIC KEY TO CHIP MANUFACTURER (ACTS AS CERTIFICATE AUTHORITY)
112
CHIP MANUFACTURER SIGNS STB HARDWARE VENDOR'S PUBLIC KEY AND RETURNS RESULTING CERTIFICATE
114
CHIP MANUFACTURER EMBEDS OWN PUBLIC KEY IN CHIP
116
STB HARDWARE VENDOR DEVELOPS BOOTLOADER
118
STB HARDWARE VENDOR SIGNS BOOTLOADER WITH OWN PRIVATE KEY, BUNDLES WITH CERTIFICATE AND LOADS SIGNED BOOTLOADER AND CERTIFICATE ON STB
120
ON POWER UP/RESET, ROM AUTHENTICATES CERTIFICATE USING EMBEDDED PUBLIC KEY
122