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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 February 2007 (01.02.2007)

PCT

(10) International Publication Number
WO 2007/014314 A2

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/US2006/029355

(22) International Filing Date: 26 July 2006 (26.07.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/190,735 26 July 2005 (26.07.2005) US

(71) Applicant *(for all designated States except US)*: **APPLE COMPUTER, INC.** [US/US]; 1 Infinite Loop, Cupertino, Cupertino, California 95014 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **WYSOCKI, Christopher R.** [US/US]; 17610 Foster Road, Los Gatos, California 95030 (US). **WARD, Alan** [GB/US]; 5225 E. 130th Circle, Thornton, Colorado 80241 (US).

(74) Agents: **THOMAS, C. Douglass** et al.; P.o. Box 70250, Oakland, California 94607 (US).

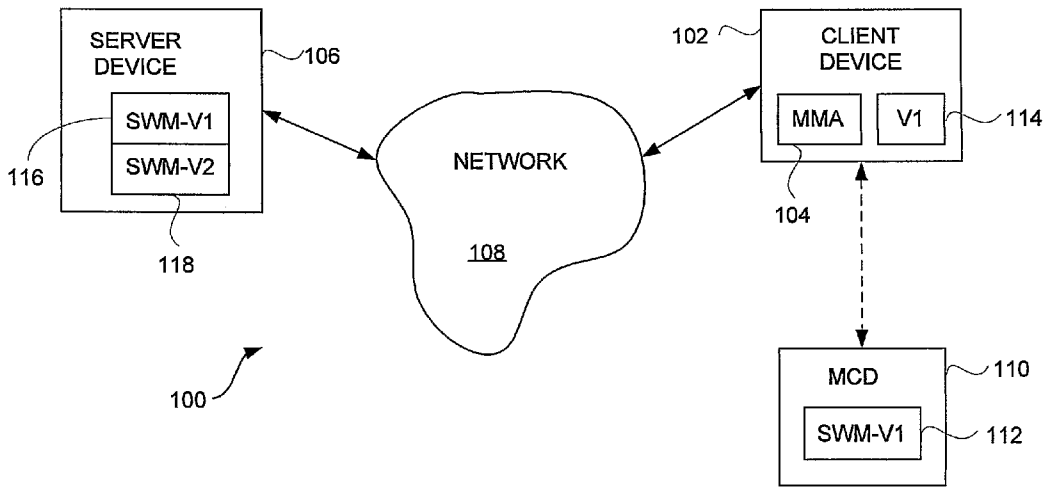
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, 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, HU, IE, IS, IT, LT, LU, LV, MC, NL, 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:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** SECURE SOFTWARE UPDATES



(57) **Abstract:** Improved techniques to update software in electronic devices that are already in use are disclosed. In one embodiment, software can be updated in a secure and controlled manner using cryptography. The authenticity of the updated software as well as its appropriateness for the particular electronic device can be confirmed prior to update. The software can also be updated on a per module basis. In one embodiment, a server hosts software updates for various electronic devices, and supplies the appropriate software update to the electronic devices via a data network.

WO 2007/014314 A2

SECURE SOFTWARE UPDATES

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention relates to updating software and, more particularly, to updating software at a client using updated software acquired from a remote server.

Description of the Related Art

[0002] It is common today for electronic devices to utilize software in their operation. Examples of electronic devices that utilize software include computers, personal digital assistants, media players and mobile telephones. However, at times, it is desirable to change or update the software being utilized by such electronic devices.

[0003] In the case of computers, updated software, such as a newer version, can be acquired from a remote server through a downloading process. Once acquired, the software can be installed on the computer. The installation process of the software can be controlled by requiring the user to enter an alphanumeric key or a registration code. Without the proper key or registration code, the updated software is unable to be installed. Still further, conventional approaches for updating software on computers requires substantial user participation. The need for user assistance is problematic given that users are concerned about downloading and installing software on computers given the propensity of computer viruses that exist today.

[0004] In the case of portable electronic devices (e.g., personal digital assistants, media assistants, mobile telephones) that utilize software, the software is typically initially installed during the manufacturing process. As a result, when the user receives the portable electronic device, the software is preinstalled and the portable electronic device is fully functional. However, when the software needs to be subsequently updated or modified, in many cases, the software installed on the portable electronic device cannot be

altered by the end user. More recently, some portable electronic devices permit the software to be updated. For example, a portable electronic device could be connected to a computer that could completely replace the existing software on the portable electronic device with updated software. One complication that results is that portable electronic devices often support multiple functionalities. These different functionalities can be controlled by different software modules which can be provided by different vendors. Hence, it is often not appropriate to completely replace all of the software on a portable electronic device. Consequently, there is a need to support software update techniques that enable different software modules to be updated without disturbing other modules.

[0005] Accordingly, there is a need for automated, secure solutions for updating software on electronic devices.

SUMMARY OF THE INVENTION

[0006] The invention pertains to improved techniques to update software in electronic devices that are already in use. In one embodiment, software can be updated in a secure and controlled manner using cryptography. The authenticity of the updated software as well as its appropriateness for the particular electronic device can be confirmed prior to update. The software can also be updated on a per module basis. In one embodiment, a server hosts software updates for various electronic devices, and supplies the appropriate software update to the electronic devices via a data network.

[0007] Although the invention is generally applicable to updating software of a wide variety of types, the invention is particularly well suited for updating digital rights management software. For security reasons, there can be a need to update DRM software in electronic devices that are in use. The improved techniques of the invention enable DRM software to be updated in a secure and controlled manner. In one implementation, the updating of the DRM software operates to modify a DRM software library provided at the electronic devices.

[0008] The invention is suitable for use with electronic devices that at least in part operate in accordance with software. The electronic devices, for example, can be computers, personal digital assistants, media players or mobile telephones.

[0009] The invention can be implemented in numerous ways, including as a method, system, device, apparatus, or computer readable medium. Several embodiments of the invention are discussed below.

[0010] As a method for upgrading software on an electronic device that operates at least partially in accordance with software, one embodiment of the invention includes at least the acts of: sending device information to a host device; receiving an encrypted software module at the electronic device, the encrypted software module being previously encrypted at the host device particularly for use by the electronic device; decrypting the encrypted software module at the electronic device; and thereafter installing the software module on the electronic device.

[0011] As a method for upgrading software on a portable electronic device, one embodiment of the invention includes at least the acts of: sending device information to a host device, the device information including device descriptive information, a public cryptographic key and a current version indicator; receiving an encrypted software module at the portable electronic device, the encrypted software module resulting from a software module available to the host device being selected based on the device descriptive information and the current version indicator and then encrypted using the public cryptographic key provided by the portable electronic device; decrypting the encrypted software module at the portable electronic device using a private cryptographic key known by the portable electronic device; authenticating the decrypted software module; and installing the software module on the portable electronic device after the decrypting and the authenticating have successfully completed.

[0012] As a computer readable medium including at least computer program code for upgrading software on a computing device, one

embodiment of the invention includes at least: computer program code for sending device information to a host device, the device information including device descriptive information, a first cryptographic key and a current version indicator; computer program code for receiving an encrypted software module at the computing device, the encrypted software module resulting from a software module available to the host device being selected based on the device descriptive information and the current version indicator and then encrypted using the first cryptographic key provided by the computing device; computer program code for decrypting the encrypted software module at the computing device using a second cryptographic key known by the computing device; computer program code for authenticating the decrypted software module; and computer program code for installing the software module on the computing device after the decrypting and the authenticating have successfully completed.

[0013] As a method for upgrading a software module on a portable electronic device, another embodiment of the invention includes at least the acts of: receiving device information at a network-based server device, the device information pertaining to the portable electronic device and including device descriptive information, a public cryptographic key and a current version indicator for the software module on the portable electronic device; determining whether an updated version of the software module is available from the server device, the determining being based on the device descriptive information pertaining to the portable electronic device; encrypting the updated version of the software module when the determining determines such to be available from the server device, the encrypting using the public cryptographic key provided by the portable electronic device; and transmitting the encrypted software module to the portable electronic device.

[0014] As a computer readable medium including at least computer program code for upgrading a software module on a computing device, another embodiment of the invention includes at least: computer program code for receiving device information at a network-based server device, the device information pertaining to the computing device and including device

descriptive information, a cryptographic key and a current version indicator for the software module on the computing device; computer program code for determining whether an updated version of the software module is available from the server device, the determining being based on the device descriptive information pertaining to the computing device; computer program code for encrypting the updated version of the software module when the determining determines such to be available from the server device, the encrypting using the cryptographic key provided by the computing device; and computer program code for transmitting the encrypted software module to the computing device.

[0015] As a computer readable medium including at least computer program code for upgrading software on an electronic device, one embodiment of the invention includes at least: computer program code for identifying, at a host device, an updated software module for the electronic device; computer program code for encrypting the updated software module for use on the electronic device; computer program code for transmitting the encrypted software module to the electronic device; computer program code for decrypting the encrypted software module at the electronic device; and computer program code for installing the software module on the electronic device.

[0016] As a network-based software update system, one embodiment of the invention includes at least: (i) a plurality of mobile client devices, each of the mobile client devices operating in accordance with at least one software module resident on the corresponding mobile client device; (ii) a server device having access to a plurality of software modules, each of the software modules being for use on specific one or more of the mobile client devices; and (iii) at least one client device operatively connectable to the server device and the mobile client devices, the client device operating a media management application for digital media assets. The digital media assets are protected by a digital rights management library having at least one of the software modules. The client device interacts with the server device over a first data link to retrieve an updated software module for the mobile client

device to be updated, the updated software module pertaining to the digital rights management library. The client device thereafter interacts with the mobile client device over a second data link to provide the updated software module to the mobile client device to be updated.

[0017] Other aspects and advantages of the invention will become apparent from the following detailed description taken in conjunction with the accompanying drawings which illustrate, by way of example, the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0018] The invention will be readily understood by the following detailed description in conjunction with the accompanying drawings, wherein like reference numerals designate like structural elements, and in which:

[0019] FIG. 1A is a block diagram of a software update system according to one embodiment of the invention.

[0020] FIG. 1B is a block diagram of the software update system after a software update has occurred.

[0021] FIG. 2 is a flow diagram of a server software update process according to one embodiment of the invention.

[0022] FIG. 3 is a flow diagram of a client software update process according to one embodiment of the invention.

[0023] FIGs. 4A and 4B are flow diagrams of a client software update process according to one embodiment of the invention.

[0024] FIG. 5A and 5B are flow diagrams of a server software update process according to one embodiment of the invention.

[0025] FIG. 6 is a flow diagram of a mobile client connection process according to one embodiment of the invention.

[0026] FIGs. 7A and 7B are flow diagrams of a mobile client disconnection process according to one embodiment of the invention.

DESCRIPTION OF THE INVENTION

[0027] The invention pertains to improved techniques to update software in electronic devices that are already in use. In one embodiment, software can be updated in a secure and controlled manner using cryptography. The authenticity of the updated software as well as its appropriateness for the particular electronic device can be confirmed prior to update. The software can also be updated on a per module basis. In one embodiment, a server hosts software updates for various electronic devices, and supplies the appropriate software update to the electronic devices via a data network.

[0028] Although the invention is generally applicable to updating software of a wide variety of types, the invention is particularly well suited for updating digital rights management software. For security reasons, there can be a need to update DRM software in electronic devices that are in use. The improved techniques of the invention enable DRM software to be updated in a secure and controlled manner. In one implementation, the updating of the DRM software operates to modify a DRM software library provided at the electronic device.

[0029] The invention is suitable for use with electronic devices that at least in part operate in accordance with software. The electronic devices, for example, can be computers, personal digital assistants, media players or mobile telephones.

[0030] Embodiments of the invention are discussed below with reference to FIGs. 1A-7B. However, those skilled in the art will readily appreciate that the detailed description given herein with respect to these figures is for explanatory purposes as the invention extends beyond these limited embodiments.

[0031] FIG. 1A is a block diagram of a software update system 100 according to one embodiment of the invention. The software update system

100 includes a client device 102 that includes a media management application (MMA) 104. The client device 102 is, for example, a computer, such as a desktop computer. The media management application 104 is an application program that operates to manage media assets available at the client device 102. The software update system 100 also includes a server device 106 that can couple to the client device 102 via a network 108. The network 108 can be a data network. The network 108 can include at least a portion of a global network, a wide area network or local area network. The network 108 can also be wired and/or wireless.

[0032] Still further, the software update system 100 includes a mobile client device (MCD) 110. The MCD 110 can be operatively coupled to the client device 102 by wired or wireless means. In one example, the MCD 110 can couple to the client device 102 over a peripheral bus cable, such as a USB cable. In another example, the MCD 110 can couple to the client device 102 via a wireless link over a wireless network (e.g., Bluetooth, WiFi, WiMax).

[0033] According to the invention, the client device 102 can facilitate updating software modules present on the MCD 110. In doing so, the client device 102 communicates with the server device 106. The server device 106 has access to a plurality of software modules that are available for distribution to appropriate mobile client devices. More specifically, the client device 102 interacts with the MCD 110 to identify a software module 112, namely, software module-version 1 (SWM-V1), that is installed on the MCD 110. The client device 102 then stores a version indication 114 associated with the identified software module 112. In the example illustrated in FIG. 1A, the version indication 114 indicates that the installed software module on the MCD 110 is version 1 (V1). The client device 102 can then communicate with the server device 106 via the network 108 to determine whether there is a newer or updated version of the software module for use on the MCD 110. In this example, the server device 106 includes software modules 116 and 118, with the software module 116 being version 1 (SWM-V1) and the software module 118 being version 2 (SWM-V2). In this example, both the software modules 116 and 118 are assumed to be suitable for use on the MCD 110. The server device 106 can then provide the updated software module 118,

namely, version 2 (SWM-V2), to the client device 102. Then, the client device 102 can forward the software module-version 2 (SWM-V2) to the MCD 110.

[0034] Although the software update system 100 illustrated in FIG. 1A illustrates a single client device and a single MCD, it should be understood that the software update system 100 is typically such that a single server can support updating software modules on a plurality of MCDs via a plurality of client devices. Moreover, although the software update system 100 illustrated in FIG. 1A utilizes one or more client devices, in another embodiment, the software update system need not utilize any client device in performing software updates. In such case, the MCDs can couple to the network 108 and communicate directly to the server device 106.

[0035] FIG. 1B is a block diagram of the software update system 100' after a software update has occurred. The software update system 100' represents the software update system 100 after the software module at the MCD 110 has been updated. Note that in FIG. 1B, the MCD 110 includes the software module 112' pertaining to the software module-version 2 (SWM-V2), and the version indicator 114' at the client device 102 indicates that the MCD 110 now utilizes version 2 (SWM-V2).

[0036] In one embodiment, the software can pertain to a digital rights management (DRM) software module. The software module can also pertain to a software library. As an example, the software module being updated can be referred to as a DRM library.

[0037] One example of a media management application is the iTunes® application, produced by Apple Computer, Inc. of Cupertino, CA. One example of a server device is the iTunes® Music Store server, also provided by Apple Computer, Inc. of Cupertino, CA.

[0038] FIG. 2 is a flow diagram of a server software update process 200 according to one embodiment of the invention. The server software update process 200 is, for example, performed by a server. The server pertains to a computing device that couples to a client, or a software program operating thereon. The server can couple to a client directly or via a network. For

example, the server can pertain to the client device 102 or the server device 106 illustrated in FIG. 1A

[0039] The server software update process 200 initially begins with a decision 202 that determines whether a software update is to be performed. When the decision 202 determines that a software update is not to be performed, the server software update process 200 awaits until a software update is to be performed. The software update can be automatically performed or performed at the request of a user. In any event, when the decision 202 determines that a software update is required, a software module (SWM) for the client is identified 204. After the software module has been identified 204, the software module is encrypted 206 for access by the client. It should be noted that the software module that was identified 204 is specifically designed for the client, and that the encryption of the software module is to restrict its usage to the client. Thereafter, the encrypted software module is sent 208 to the client. Following the operation 208, the server software update process 200 ends.

[0040] FIG. 3 is a flow diagram of a client software update process 300 according to one embodiment of the invention. The client software update process 300 is, for example, performed by a client operating in accordance with one embodiment of the invention. As an example, the client is typically an electronic device that utilizes software, or a software program operating thereon. For example, the client can pertain to the mobile client device 110 illustrated in FIG. 1A.

[0041] The client software update process 300 begins with a decision 302 that determines whether a software module is to be installed at the client. When the decision 302 determines that a software module is not to be installed, then the client software update process 300 awaits the need to install a software module on the client. In other words, the client software update process 300 can be deemed invoked whenever a software module is to be installed on the client. Once the decision 302 determines that a software module is to be installed, the encrypted software module is decrypted 304 at the client. Following the decryption 304, the software

module is installed 306 on the client. After the software module has been installed 306 at the client, the client software update process 300 ends.

[0042] FIGs. 4A and 4B are flow diagrams of a client software update process 400 according to one embodiment of the invention. The client software update process 400 is, for example, performed by a client operating in accordance with one embodiment of the invention. As an example, with reference to FIG. 1A, the client can pertain to the client device 102 or the media management application 104 operating thereon.

[0043] The client software update process 400 begins with a decision 402 that determines whether a media management application has been launched. When the decision 402 determines that a media management application has not been launched, then the client software update process 400 awaits such an event. On the other hand, once the decision 402 determines that a media management application has been launched, a decision 404 checks for an available software module. Here, an available software module is typically a newer version of the software module that is suitable for being utilized on the corresponding mobile client device (MCD). The client software update process 400 need not check for available software modules every time it is launched; instead, this can be done periodically (e.g., weekly).

[0044] When the decision 404 determines that checking for an available software module is to be performed, a version request is sent 406 to the server. The version request includes at least a current version identifier and MCD descriptive information. The MCD descriptive information is information that describes general characteristics, features or attributes of the MCD.

[0045] Next, a decision 408 determines whether a version response has been received from the server. When the decision 408 determines that a version response has not been received, the client software update process 400 can await such a response. However, the waiting period can be limited or processed in a separate non-blocking thread. In any case, once the decision 408 determines that a version response has been received, an available version indication is stored 410 at the client. The version response provides

the available version indication to the client. In one embodiment, the available version indication can indicate whether or not an updated software module for the MCD is available from the server.

[0046] At this point, the client software update process 400 effectively waits until the MCD connects to the client. While this is not necessary in other embodiments, the connection can allow the MCD to complete the balance of the client software update process 400. While waiting for the disconnection, the MCD can perform other operations unrelated to software update.

[0047] More particularly, as illustrated in FIGs. 4A and 4B, following the block 410 or following the decision 404 when no available software module is found, a decision 412 then determines whether the MCD is connected to the client. Typically, the decision 412 would be concerned with whether the MCD has recently been connected to the client. When the decision 412 determines that the MCD is not connected, other processing 414 can optionally be performed by the client. Such other processing 414 would normally be unrelated to upgrading a software module. A decision 416 then determines whether the client software update process 400 should be closed. When the decision 416 determines that the client software update process 400 should be closed, the client software update process 400 ends. Alternatively, when the decision 416 determines that the client software update process 400 should not be closed, the client software update process 400 returns to repeat the decision 412 so as to wait for the MCD to be connected to the client.

[0048] Once the decision 412 determines that the MCD is connected to the client, a decision 418 determines whether an available version indication is present. Recall, the available version indication was previously stored 410 at the client based on information provided in a version response from the server. When the decision 418 determines that there is an available version indication, a software module request is sent 420 for the available software module for the MCD. Here, the software module request is sent for 420 to the server and requests that the available software version module be provided to the client. The software module request can include a version identifier for the available software module desired and an encryption key, namely, a public encryption key, to be used to encrypt the available software module.

Next, a decision 422 determines whether a software module response has been received from the server. When the decision 422 determines that a software module response has not yet been received, the client software update process 400 can await such a response. Once the decision 422 determines that a software module response has been received, an encrypted software module provided by the software module response can be copied 424 to the MCD. Following the operation 424 or following the decision 418 when it is determined that there is no available version indication, the client software update process 400 is complete and ends.

[0049] FIG. 5A and 5B are flow diagrams of a server software update process 500 according to one embodiment of the invention. The server software update process 500 is, for example, performed by a server operating in accordance with one embodiment of the invention. As an example, with reference to FIG. 1A, the server can pertain to the server device 106 or a software application operating thereon.

[0050] Typically, the server is capable of performing a plurality of different processes. The server software update process 500 is considered one such process that can be performed by the server. Accordingly, the processing discussed in FIGs. 5A and 5B is processing directed at a software update for a client device (e.g., mobile client device) and such processing may be intertwined with other processing performed at the server.

[0051] The server software update process 500 begins with a decision 502 that determines whether a version request has been received. When the decision 502 determines that a version request has been received, a most current version of the software module for the MCD is determined 504 based on the MCD descriptive information. Here, the version request that has been received from the client includes an indication of the current version of the software module on the MCD as well as MCD descriptive information. The MCD descriptive information is information that describes general characteristics, features or attributes of the MCD.

[0052] Next, a decision 506 determines whether the current version of the software module on the MCD is the same as the most current version

available from the server. When the decision 506 determines that the current version of the software module on the MCD is the same as the most current version available at the server, a version response is sent 508 to the client indicating that there is no available version of the software module for the MCD. In other words, in this condition, there is no need to update the software module on the MCD. On the other hand, when the decision 506 determines that the current version of the software module on the MCD is not the same as the most current version available at the server, a version response is sent 510 to the client indicating that there is an available version of the software module for the MCD.

[0053] Following the blocks 508 and 510, as well as following the decision 502 when a version request has not been received, additional processing can be performed by the server software update process 500 when a software module request has been received. In particular, when a decision 512 determines that a software module request has been received, the most current version of the software module for the MCD is retrieved 514. Here, the most current version of the software module for the MCD is retrieved 514 from the server. In other words, the server centrally makes various versions of software modules for various MCDs available.

[0054] Next, the retrieved software module is encrypted 516 using a public-key for the MCD. Here, the software module request provides a public-key to be used in encrypting (directly or indirectly) the retrieved software module. The public-key is part of a key pair that is specifically associated with the MCD. In one embodiment, the key pair is stored on the MCD. After the retrieved software module is encrypted 516, a software module response is sent 518 to the client. The software module response includes at least the encrypted software module for the MCD.

[0055] Thereafter, other processing 520 may be optionally performed at the server. At some point thereafter, a decision 522 determines whether the server software update process 500 should close. When the decision 522 determines that the server software update process 500 should not close, then the server software update process 500 returns to the beginning of the server software update process 500. Alternatively, when the decision 522

determines that the server software update process 500 should close, then the server software update process 500 ends.

[0056] In general, client or the server can be considered a host device. In FIGs. 4A and 5A, the client interacts with the server to determine whether an updated version of the SWM is present. In this embodiment, the server determines whether an updated version of the SWM is present, and if so informs the client of the updated version. Thereafter, at the appropriate time, the client would retrieve the updated version of the SWM for the MCD.

[0057] However, in another embodiment, the client can determine whether an updated version of the SWM is present. This embodiment would represent an embodiment that differs from the embodiment of FIGs. 4A and 4B. In such an embodiment, the client can periodically query the server for a table (or list) of most current versions for a plurality of different devices. The client then stores the table (which can include version numbers representing the most current versions for the different devices). Thereafter, when the MCD is connected to the client, the client obtains the MCD descriptive information (including current version on the MCD) and compares such with the most current version available for that device as indicated in the stored table. If there is an available software version, the client requests the appropriate software update (e.g., using a version number) from the server. Once the appropriate software update is received, the available software module can be supplied to the MCD.

[0058] FIG. 6 is a flow diagram of a mobile client connection process 600 according to one embodiment of the invention. The mobile client connection process 600 is, for example, performed by a portable client operating in accordance with one embodiment of the invention. For example, the portable client can be a mobile client device (MCD). As an example, with reference to FIG. 1A, the MCD can pertain to the mobile client device 110 or a software application operating thereon.

[0059] The mobile client connection process 600 begins with a decision 602 that determines whether the MCD is connected to the client. When the decision 602 determines that the MCD is not connected to the client, either by

wired or wireless means, the mobile client connection process 600 awaits such a connection. In other words, the mobile client connection process 600 can be deemed invoked once a connection is established between the MCD and the client. In any event, once the decision 602 determines that a connection exists between the MCD and the client, MCD descriptive information and a current version identifier are provided 604 to the client. Here, the MCD descriptive information as well as the current version identifier are maintained by the MCD. Then, other processing 606 can be performed at the MCD. Such other processing 606 would typically not be part of the mobile client connection processing 600, but is illustrated in FIG. 6 for context. As an example, one type of other processing 606 that could be performed is a synchronization operation between the MCD and the client, e.g., to synchronize music libraries, calendars, etc. Additional details on synchronization of digital assets or data can be found in U.S. Patent Application No. 10/277,418, filed October 21, 2002, and entitled "INTELLIGENT INTERACTION BETWEEN MEDIA PLAYER AND HOST COMPUTER" [Att.Dkt.No.: APL1P228X1], which is hereby incorporated herein by reference.

[0060] At some point while the MCD is connected to the client, a software update will be performed. The software update is performed in a secure manner. Hence, according to the mobile client connection process 600, the MCD will receive an encrypted software module from the client. The mobile client connection processing 600 includes a decision 608 that determines whether an encrypted software module has been received. When the decision 608 determines that an encrypted software module has been received at the MCD, the encrypted software module is stored 610 in memory of the MCD. The memory can be of many different types, including Flash memory storage, disk drive storage, etc. Following the block 610 or following the decision 608 when an encrypted software module is not received, the mobile client connection process 600 ends.

[0061] FIGs. 7A and 7B are flow diagrams of a mobile client disconnection process 700 according to one embodiment of the invention. The mobile client disconnection process 700 is, for example, performed by a portable client

operating in accordance with one embodiment of the invention. For example, the portable client can be a mobile client device (MCD). As an example, with reference to FIG. 1A, the MCD can pertain to the mobile client device 110 or a software application operating thereon.

[0062] The mobile client connection process 700 begins with a decision 702 that determines whether the MCD has been disconnected from the client. When the decision 702 determines that the MCD has not been disconnected from the client, then the mobile client disconnection process 700 awaits such disconnection. In other words, the mobile client disconnection process 700 is initiated once the MCD is disconnected from the client. Hence, when the decision 702 determines that the MCD has been disconnected from the client, a decision 704 determines whether an encrypted software module is present on the MCD. Here, as noted in block 610 of FIG. 6, the mobile client connection process 600 operates to store the appropriate encrypted software module on the MCD. Here, at the decision 704, a determination is made as to whether an encrypted software module has been stored on the MCD.

[0063] When the decision 704 determines that an encrypted software module has been stored on the MCD, the encrypted software module is decrypted 706 using a private key provided within the MCD. Here, the MCD, as previously noted, includes a pair of cryptographic keys. These cryptographic keys include the public key noted above as well as a private key. The decryption of the encrypted software module is performed using the required private key. Hence, the encrypted software module is only able to be properly decrypted if the software module was encrypted for use on the MCD. In other words, the encryption of the software module was performed using the public key that is the counterpart of the private key stored in the MCD.

[0064] Assuming that the decryption 706 is successful, the software module can be validated 700. In one embodiment, the software module can be validated 700 using a digital signature. By verification of the digital signature, the validity of the software module is established. For example, the manufacturer of the MCD can ensure that the software module is authentic (i.e., approved by the manufacturer) before being permitted to be utilized thereon. A decision 710 then determines whether the software module is

valid. Here, to be valid, the software module must not only be properly decrypted but also successfully authenticated.

[0065] When the decision 710 determines that the software module is valid, a decision 712 determines whether the software module is suitable for the MCD. Here, the software module can be determined to be suitable for the MCD when the software module is affiliated with the MCD. The software module can be properly affiliated when the software module is suitable for use with the MCD. For example, the decision 712 can determine whether the software module is suitable for use on the model and/or hardware platform of the MCD. As a particular example, the software module can include one or more identifiers for the model and/or platform of the MCD, and these identifiers can be compared with like identifiers stored in the MCD.

[0066] When the decision 712 determines that the software module is suitable for the MCD, the software module can be installed 714 on the MCD. Next, a decision 716 determines whether the installation of the software module has been successful. When the decision 716 determines that the installation has not been successful, the installation 714 can be repeated. However, if the installation of the software module repeatedly fails, the mobile client disconnection process 700 can end without having installed the software module. On the other hand, when the decision 716 determines that the software module has been successfully installed on the MCD, the uninstalled software module can be deleted 718. Here, the uninstalled software module was stored in the memory of the MCD (e.g., block 610 of FIG. 6); hence, the deletion 718 of the uninstalled software module is performed for security reasons as well as to free-up memory of the MCD. In addition, a current version indicator is updated 720 for the MCD. The updating 720 of the current version indicator is appropriate because the software module on the MCD has been updated and is thus now the current version of the software module. The stored current version indicator also facilitates providing of current version information to the client as noted above (e.g., block 604 of FIG. 6). Following the block 720, as well as following any of the decisions 704, 710 and 712 when the evaluated conditions are not present, the mobile client disconnection process 700 is complete and ends.

[0067] With regards to authentication, the authentication of the software module (as discussed above), such as by a digital signature, can be utilized by a vendor. As an example, the updated software module can be achieved for a first vendor, but a second vendor can require that the software module be approved by them before being installed or otherwise provided to the electronic device. For example, if the first vendor is a software provider and the second vendor is a hardware platform provider, the first vendor can provide the updated software module to the electronic device in a secure manner, but the second vendor can require that the software module be authenticated or validated before being installed on the electronic device. Additionally, the second vendor might also provide their own level of encryption apart of any encryption provided by the first vendor. Hence, in one implementation, the software module of the first vendor can be packaged with a digital signature and/or encryption of the second vender before being made available to clients.

[0068] As noted above, a cryptographic key can be used to secure and control the software update process. For additional security or performance reasons, a combination of cryptographic keys can be used. As a result, to the extent that a public key is used, the public key need not be used to directly encrypt the software module. In one embodiment, the encryption process operates as follows. First, a random cryptographic key (random key) is generated. As an example, the random key can be a 128-bit AES key, which is a random symmetric key. The software module is first encrypted using the random key. This results in an encrypted software module. In addition, the random key is encrypted using the public key provided by the electronic device. This results in an encrypted cryptographic key. In one example, the encrypted cryptographic key is a 1024-bit RSA key. In this embodiment, the electronic device (e.g., MCD) receives the encrypted software module in a first electronic file, and receives the encrypted cryptographic key in a second electronic file. Thereafter, to install the software module on the electronic device, the encrypted cryptographic key in second electronic file is decrypted using a private key resident in the electronic device. The resulting cryptographic key is the random key which can then be used to decrypt the

encrypted software module in the first electronic file. The software module is then in the "clear" (i.e., unencrypted) and can be installed on the electronic device.

[0069] The software module update according to the invention can be provided in automatic fashion. Namely, as the client operatively connects to a server, the server can provide the client with any updated software modules without the participation of the user of the client. Alternatively, in another embodiment, the user could be prompted at the client (e.g., portable electronic device) for permission to install an updated software module.

[0070] The various aspects, embodiments, implementations or features of the invention can be used separately or in any combination.

[0071] The invention is preferably implemented by software, but can also be implemented in hardware or a combination of hardware and software. The invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can thereafter be read by a computer system. Examples of the computer readable medium include read-only memory, random-access memory, CD-ROMs, DVDs, magnetic tape, optical data storage devices, and carrier waves. The computer readable medium can also be distributed over network-coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

[0072] The advantages of the invention are numerous. Different aspects, embodiments or implementations may yield one or more of the following advantages. One advantage of the invention is that software updates can be performed over a network in a secure manner. The secure nature of the software updates prevents reverse-engineering of the software. For example, the security imposed secures against unauthorized interception and inspection of the software while being transmitted to an electronic device. Another advantage of the invention is that software used by an electronic device can be updated on a per-module basis, which is particularly useful when the electronic device uses software or hardware from different vendors. Still another advantage of the invention is that software updates can be

performed in an automated manner, and thus need not burden users of electronic devices with software updates.

[0073] The many features and advantages of the present invention are apparent from the written description. Further, since numerous modifications and changes will readily occur to those skilled in the art, the invention should not be limited to the exact construction and operation as illustrated and described. Hence, all suitable modifications and equivalents may be resorted to as falling within the scope of the invention.

What is claimed is:

CLAIMS

1. A method for upgrading software on an electronic device that operates at least partially in accordance with software, said method comprising the acts of:

(a) sending device information to a host device;

(b) receiving an encrypted software module at the electronic device, the encrypted software module being previously encrypted at the host device particularly for use by the electronic device;

(c) decrypting the encrypted software module at the electronic device; and

(d) thereafter installing the software module on the electronic device.

2. A method as recited in claim 1,

wherein the device information includes a cryptographic key, and

wherein the encrypted software module being received has been encrypted using the cryptographic key.

3. A method as recited in claim 2,

wherein the cryptographic key is a public key associated with the electronic device, and

wherein the encrypted software module being received has been encrypted directly or indirectly using the public key.

4. A method as recited in claim 3, wherein said decrypting (c) is performed directly or indirectly using a private cryptographic key stored in the electronic device.

5. A method as recited in claim 1, wherein the cryptographic key is a public key associated with the electronic device, and

wherein the encrypted software module being received has been encrypted using a randomly generated key, and the randomly generated key is encrypted using the public key.

6. A method as recited in claim 5, wherein said decrypting (c) initially decrypts the encrypted randomly generated key using a private cryptographic key stored in the electronic device, and then decrypts the encrypted software module using the randomly generated key.

7. A method as recited in claim 1, wherein the device information includes a version indicator and one or more of manufacturer information, model information or hardware platform information.

8. A method as recited in claim 1, wherein the host device is a server computer or a client computer, and

wherein the electronic device is a mobile telephone, a personal digital assistant or a media player.

9. A method as recited in claim 1, wherein said method is performed automatically without user interaction requesting software upgrading.

10. A method for upgrading software on a portable electronic device, said method comprising the acts of:

sending device information to a host device, the device information including device descriptive information, a public cryptographic key and a current version indicator;

receiving an encrypted software module at the portable electronic device, the encrypted software module resulting from a software module available to the host device being selected based on the device descriptive information and the current version indicator and then encrypted using the public cryptographic key provided by the portable electronic device;

decrypting the encrypted software module at the portable electronic device using a private cryptographic key known by the portable electronic device;

authenticating the decrypted software module; and

installing the software module on the portable electronic device after said decrypting and said authenticating have successfully completed.

11. A method as recited in claim 10, wherein said sending and said receiving are performed while the portable electronic device is operatively connected to the host device.

12. A method as recited in claim 11, wherein said decrypting, said authenticating and said installing are performed after the portable electronic device has been disconnected from the host device.

13. A method as recited in claim 10, wherein said method further comprises:

recognizing that the portable electronic device has been disconnected from the host device; and

performing said decrypting only after said recognizing recognizes that the portable electronic device has been disconnected from the host device.

14. A method as recited in claim 10, wherein said method further comprises:

updating the current version indicator after said installing.

15. A method as recited in claim 10, wherein the private cryptographic key is unique to the portable electronic device.

16. A method as recited in claim 10, wherein said authenticating of the decrypted software module utilizes a digital signature.

17. A computer readable medium including at least computer program code for upgrading software on a computing device, said computer readable medium comprising:

computer program code for sending device information to a host device, the device information including device descriptive information, a first cryptographic key and a current version indicator;

computer program code for receiving an encrypted software module at the computing device, the encrypted software module resulting from a software module available to the host device being selected based on the device descriptive information and the current version indicator and then encrypted using the first cryptographic key provided by the computing device;

computer program code for decrypting the encrypted software module at the computing device using a second cryptographic key known by the computing device;

computer program code for authenticating the decrypted software module; and

computer program code for installing the software module on the computing device after said decrypting and said authenticating have successfully completed.

18. A computer readable medium as recited in claim 17, wherein the computing device is a portable computing device.

19. A computer readable medium as recited in claim 18, wherein the portable computing device is a mobile telephone, a personal digital assistant or a media player.

20. A method for upgrading a software module on a portable electronic device, said method comprising the acts of:

receiving device information at a network-based server device, the device information pertaining to the portable electronic device and including device descriptive information, a public cryptographic key and a current version indicator for the software module on the portable electronic device;

determining whether an updated version of the software module is available from the server device, said determining being based on the device descriptive information pertaining to the portable electronic device;

encrypting the updated version of the software module when said determining determines such to be available from the server device, said encrypting using the public cryptographic key provided by the portable electronic device; and

transmitting the encrypted software module to the portable electronic device.

21. A method as recited in claim 20, wherein said method further comprises:

receiving the encrypted software module at the portable electronic device;

accessing a private cryptographic key resident on the portable client device; and

decrypting the encrypted software module at the portable electronic device using the private cryptographic key.

22. A method as recited in claim 21, wherein said method further comprises:

installing the software module on the electronic device after said decrypting has successfully completed.

23. A method as recited in claim 21, wherein said method further comprises:

authenticating the decrypted software module; and

installing the software module on the electronic device after said decrypting and said authenticating have successfully completed.

24. A method as recited in claim 20,

wherein said encrypting comprises: (i) encrypting the updated version of the software module using a random key, and (ii) encrypting the random key using the public cryptographic key,

wherein said transmitting operates to transmit the encrypted software module and the encrypted random key.

25. A method as recited in claim 24, wherein said method further comprises:

receiving the encrypted software module and the encrypted random key at the portable electronic device;

accessing a private cryptographic key resident on the portable client device;

decrypting the encrypted random key using the private cryptographic key to provide an acquired random key; and

decrypting the encrypted software module at the portable electronic device using the acquired random key.

26. A method as recited in claim 25, wherein said method further comprises:

installing the software module on the electronic device after said decrypting has successfully completed.

27. A computer readable medium including at least computer program code for upgrading a software module on a computing device, said computer readable medium comprising:

computer program code for receiving device information at a network-based server device, the device information pertaining to the computing device and including device descriptive information, a cryptographic key and a current version indicator for the software module on the computing device;

computer program code for determining whether an updated version of the software module is available from the server device, said determining being based on the device descriptive information pertaining to the computing device;

computer program code for encrypting the updated version of the software module when said determining determines such to be available from the server device, said encrypting using the cryptographic key provided by the computing device; and

computer program code for transmitting the encrypted software module to the computing device.

28. A computer readable medium including at least computer program code for upgrading software on an electronic device, said computer readable medium comprising:

computer program code for identifying, at a host device, an updated software module for the electronic device;

computer program code for encrypting the updated software module for use on the electronic device;

computer program code for transmitting the encrypted software module to the electronic device;

computer program code for decrypting the encrypted software module at the electronic device; and

computer program code for installing the software module on the electronic device.

29. A computer readable medium as recited in claim 28,

wherein said computer readable medium further comprises:

computer program code for receiving device information pertaining to the electronic device, and

wherein the identification of the software module for the electronic device by said computer program code for identifying is dependent on the device information.

30. A network-based software update system, comprising:

a plurality of mobile client devices, each of the mobile client devices operating in accordance with at least one software module resident on the corresponding mobile client device;

a server device having access to a plurality of software modules, each of the software modules being for use on specific one or more of the mobile client devices; and

at least one client device operatively connectable to the server device and the mobile client devices, the client device operating a media management application for digital media assets,

wherein the digital media assets are protected by a digital rights management library having at least one of the software modules, and

wherein the client device interacts with the server device over a first data link to retrieve an updated software module for the mobile client device to be updated, the updated software module pertaining to the digital rights management library, and wherein the client device thereafter interacts with the mobile client device over a second data link to provide the updated software module to the mobile client device to be updated.

31. A network-based software update system as recited in claim 30, wherein subsequently the mobile client device replaces an existing software with the updated software module.

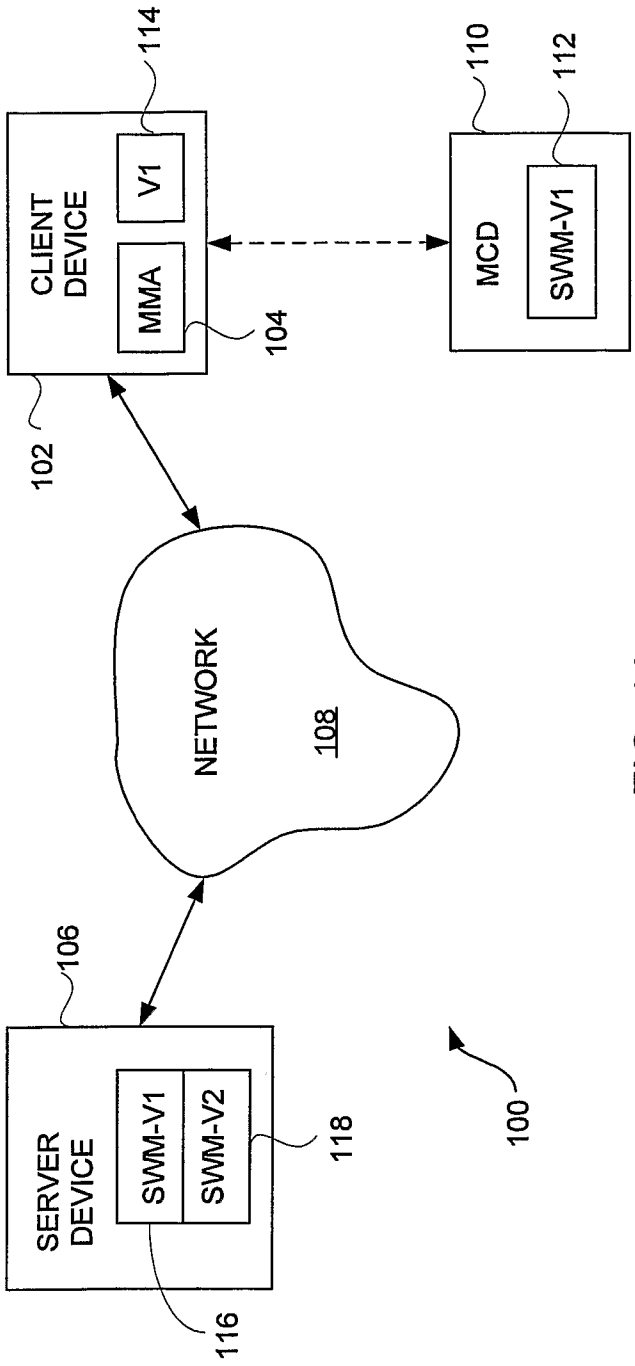

CLIENT DEVICE
V1
MMA
104
114
102
MCD
SWM-V1
110
112
NETWORK
108
SERVER DEVICE
SWM-V1
SWM-V2
118
116
106
100

FIG. 1A

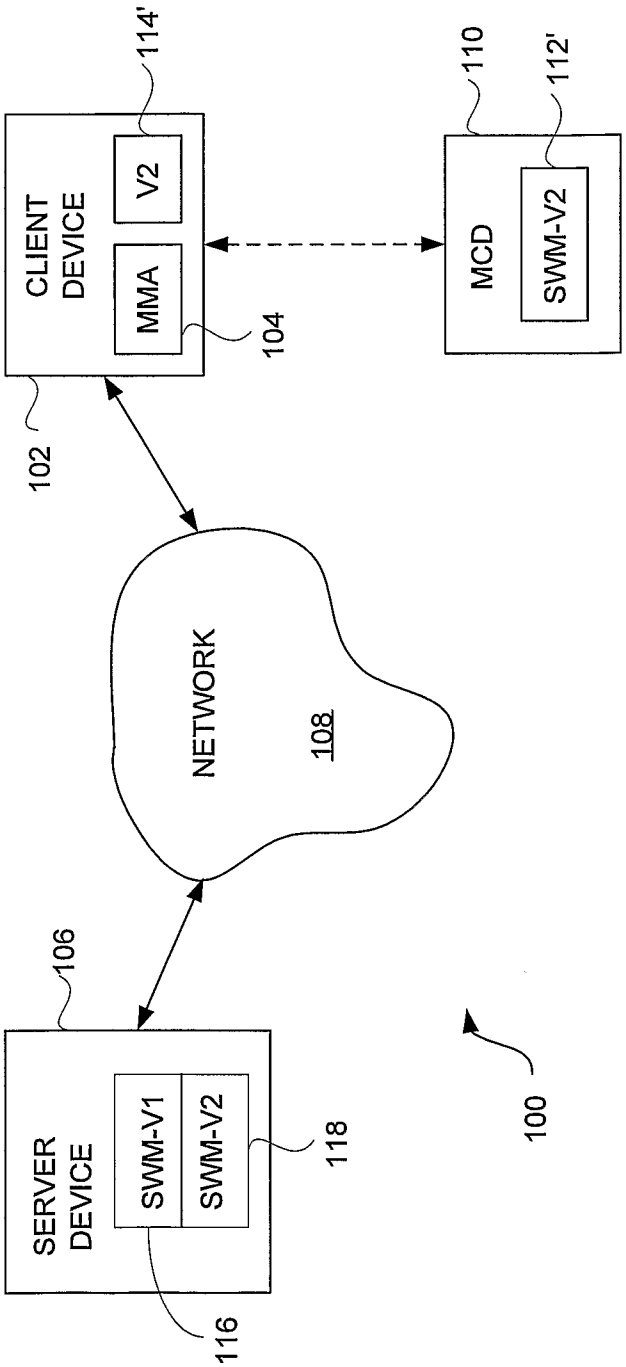

CLIENT DEVICE
V2
MMA
114'
104
102
MCD
SWM-V2
110
112'
NETWORK
108
SERVER DEVICE
SWM-V1
SWM-V2
106
116
118
100

FIG. 1B

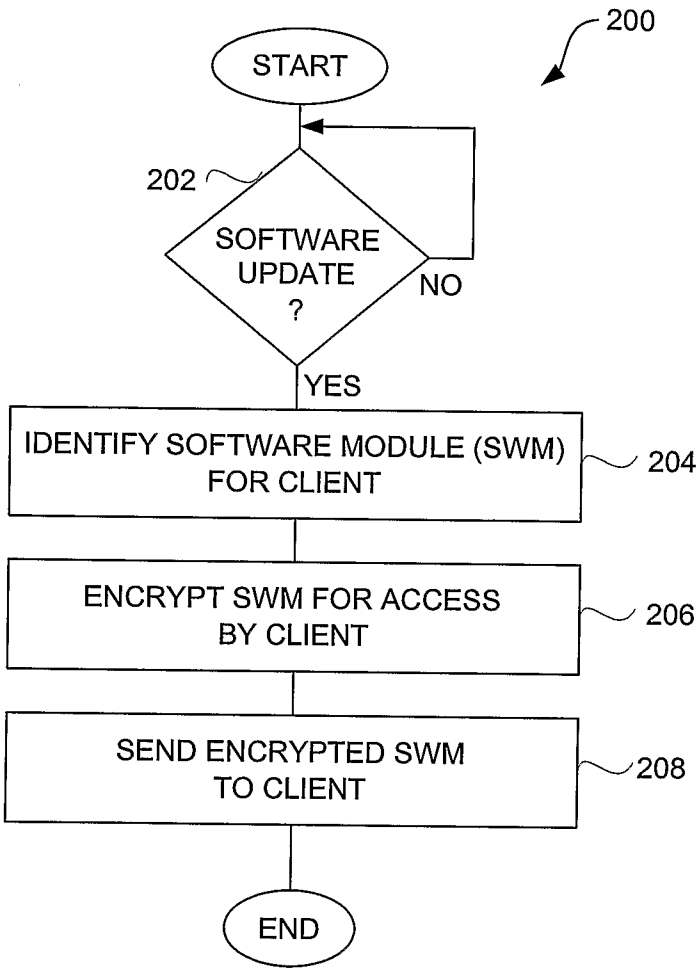

200
START
202
SOFTWARE UPDATE ?
NO
YES
IDENTIFY SOFTWARE MODULE (SWM) FOR CLIENT
204
ENCRYPT SWM FOR ACCESS BY CLIENT
206
SEND ENCRYPTED SWM TO CLIENT
208
END

FIG. 2

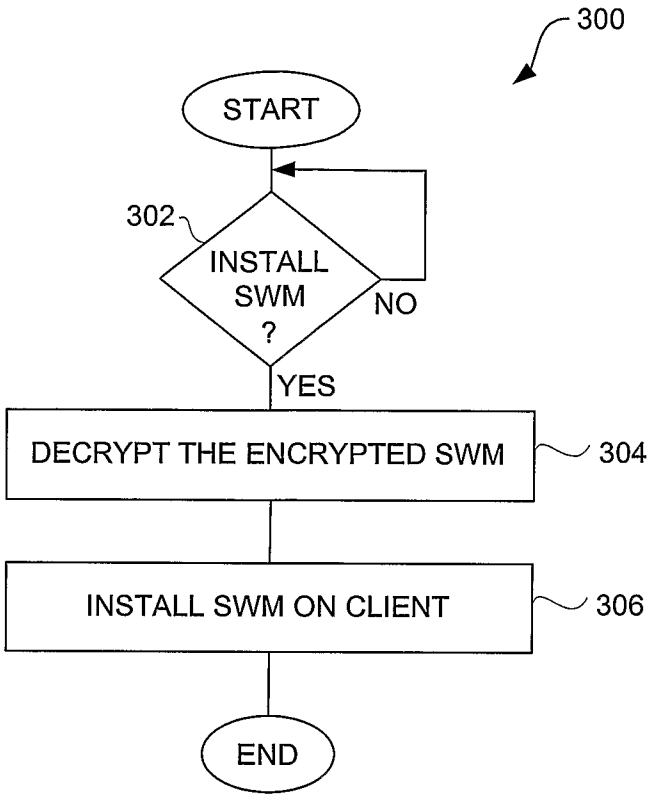

300
START
302
INSTALL SWM ?
NO
YES
DECRYPT THE ENCRYPTED SWM
304
INSTALL SWM ON CLIENT
306
END

FIG. 3

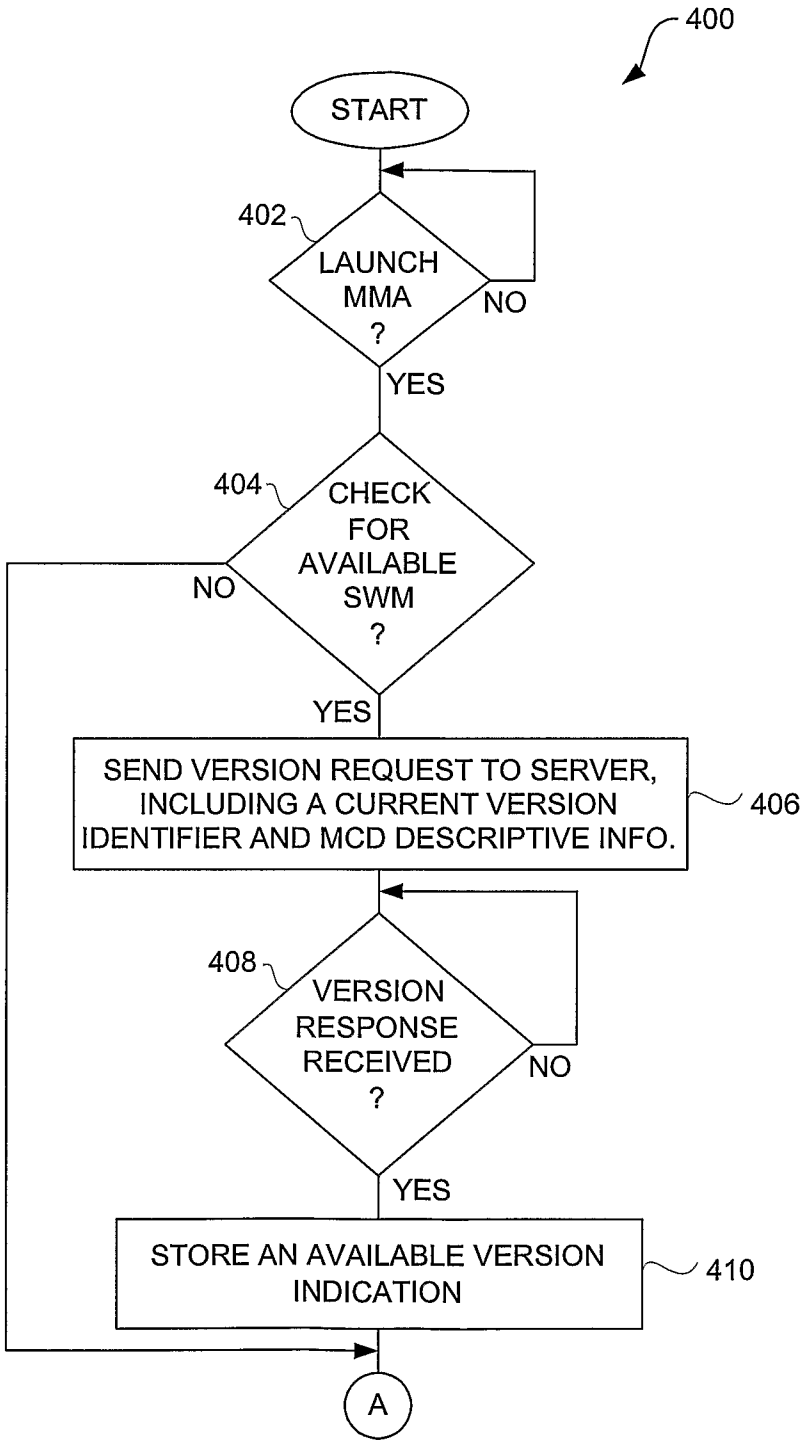

400
START
402
LAUNCH MMA ?
NO
YES
404
CHECK FOR AVAILABLE SWM ?
NO
YES
SEND VERSION REQUEST TO SERVER, INCLUDING A CURRENT VERSION IDENTIFIER AND MCD DESCRIPTIVE INFO.
406
408
VERSION RESPONSE RECEIVED ?
NO
YES
STORE AN AVAILABLE VERSION INDICATION
410
A

FIG. 4A

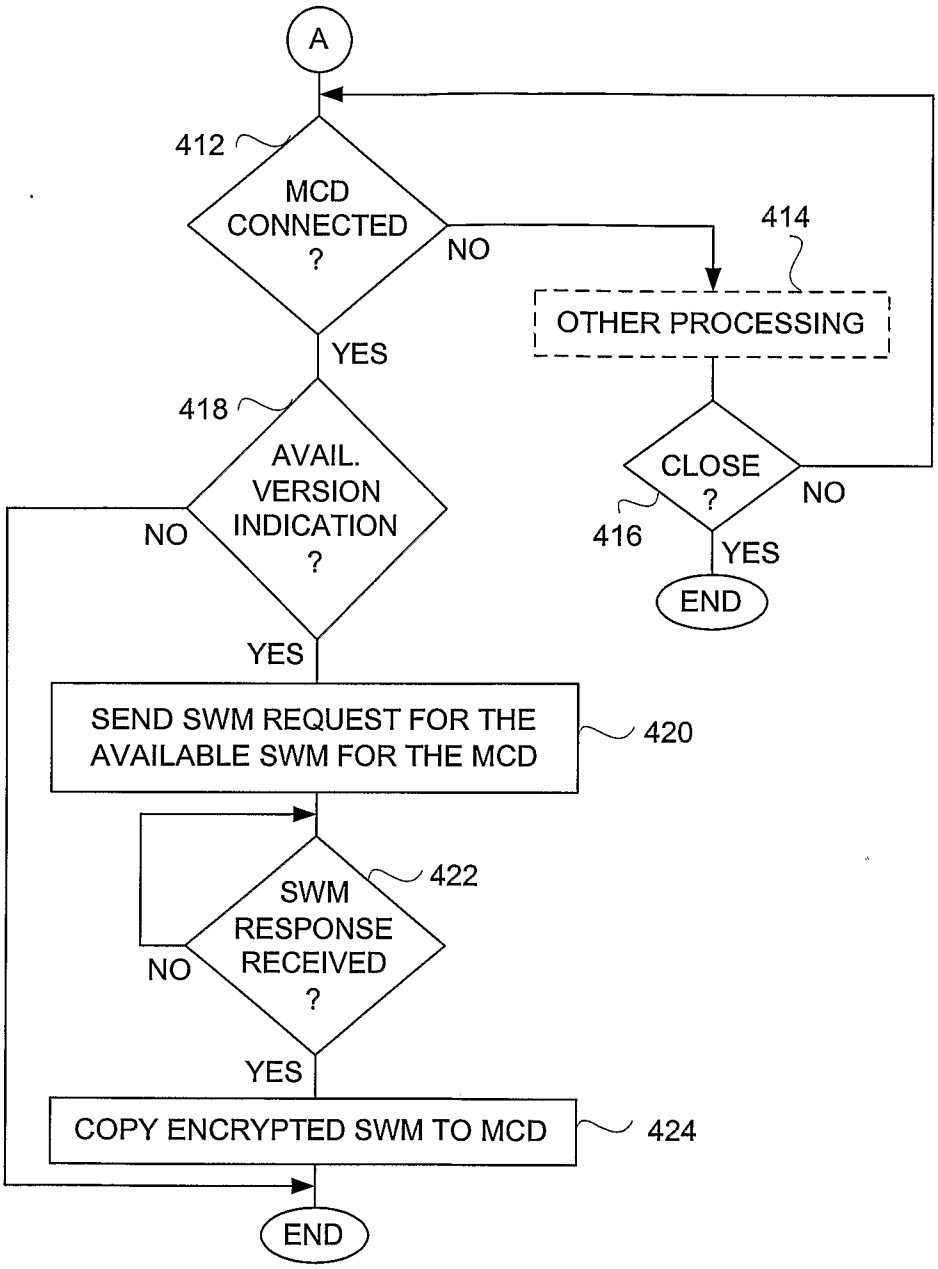

A
412
MCD CONNECTED ?
NO
414
OTHER PROCESSING
YES
418
AVAIL. VERSION INDICATION ?
NO
CLOSE ?
NO
416
YES
END
YES
SEND SWM REQUEST FOR THE AVAILABLE SWM FOR THE MCD
420
422
SWM RESPONSE RECEIVED ?
NO
YES
COPY ENCRYPTED SWM TO MCD
424
END

FIG. 4B

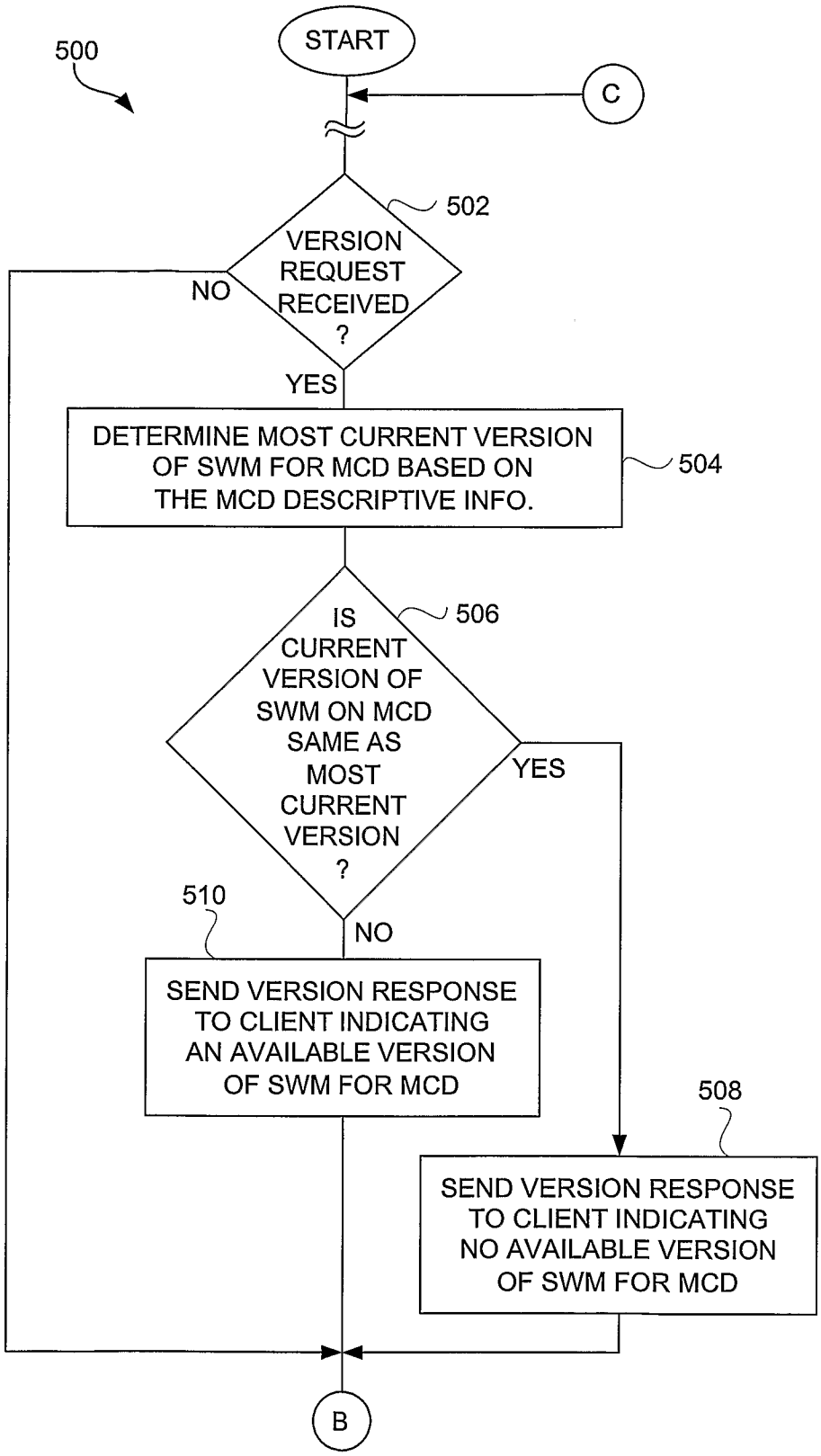

500
START
C
502
VERSION REQUEST RECEIVED ?
NO
YES
DETERMINE MOST CURRENT VERSION OF SWM FOR MCD BASED ON THE MCD DESCRIPTIVE INFO.
504
IS CURRENT VERSION OF SWM ON MCD SAME AS MOST CURRENT VERSION ?
506
YES
NO
510
SEND VERSION RESPONSE TO CLIENT INDICATING AN AVAILABLE VERSION OF SWM FOR MCD
508
SEND VERSION RESPONSE TO CLIENT INDICATING NO AVAILABLE VERSION OF SWM FOR MCD
B

FIG. 5A

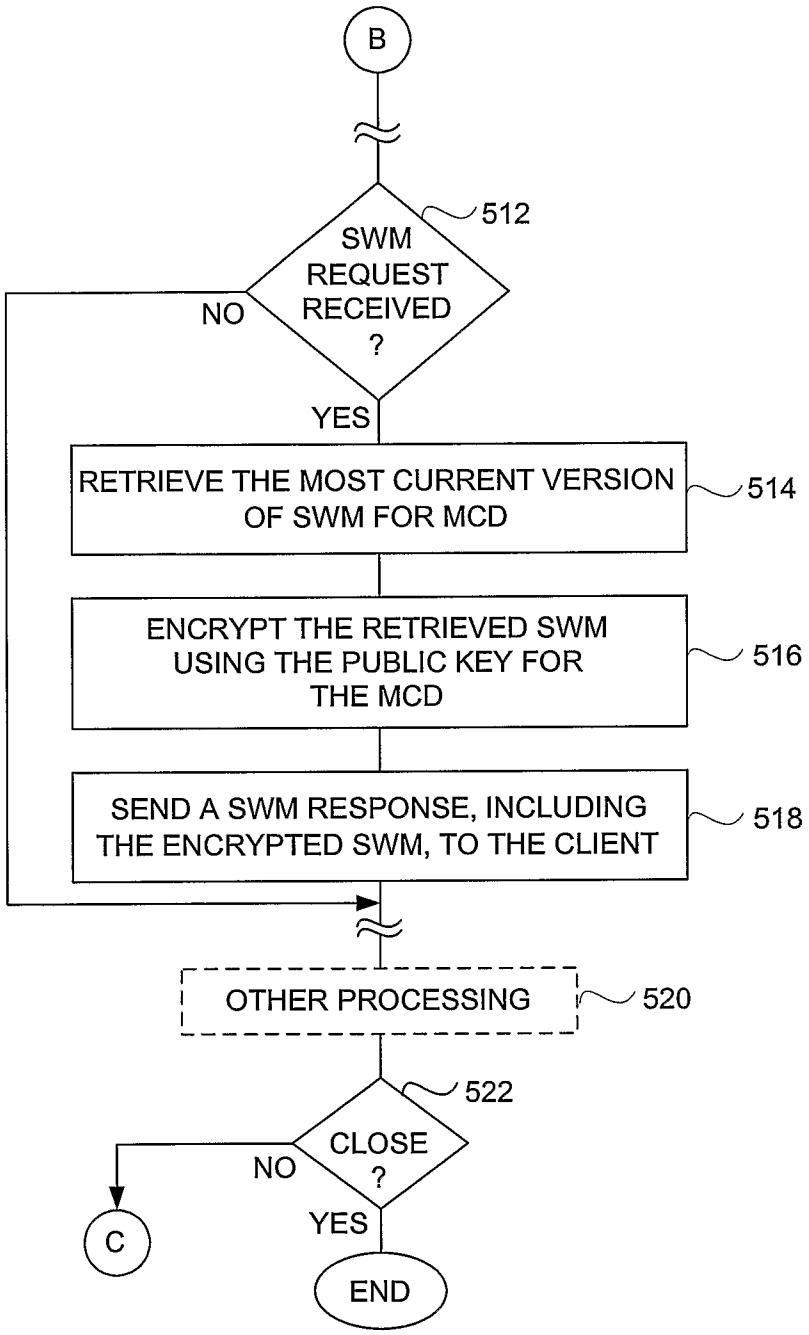

B
512
SWM REQUEST RECEIVED ?
NO
YES
RETRIEVE THE MOST CURRENT VERSION OF SWM FOR MCD
514
ENCRYPT THE RETRIEVED SWM USING THE PUBLIC KEY FOR THE MCD
516
SEND A SWM RESPONSE, INCLUDING THE ENCRYPTED SWM, TO THE CLIENT
518
OTHER PROCESSING
520
522
CLOSE ?
NO
YES
C
END

FIG. 5B

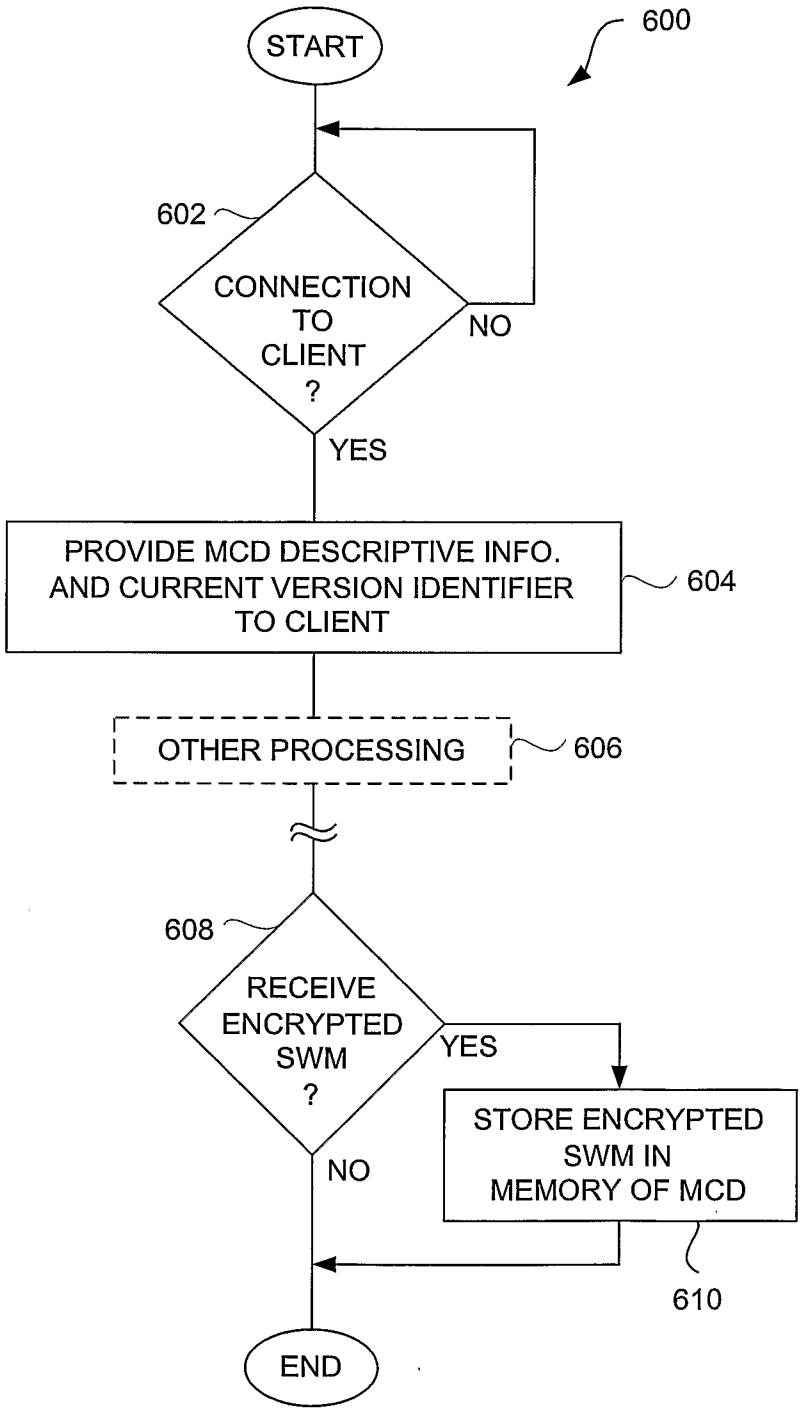

600
START
602
CONNECTION TO CLIENT ?
NO
YES
PROVIDE MCD DESCRIPTIVE INFO. AND CURRENT VERSION IDENTIFIER TO CLIENT
604
OTHER PROCESSING
606
608
RECEIVE ENCRYPTED SWM ?
YES
NO
STORE ENCRYPTED SWM IN MEMORY OF MCD
610
END

FIG. 6

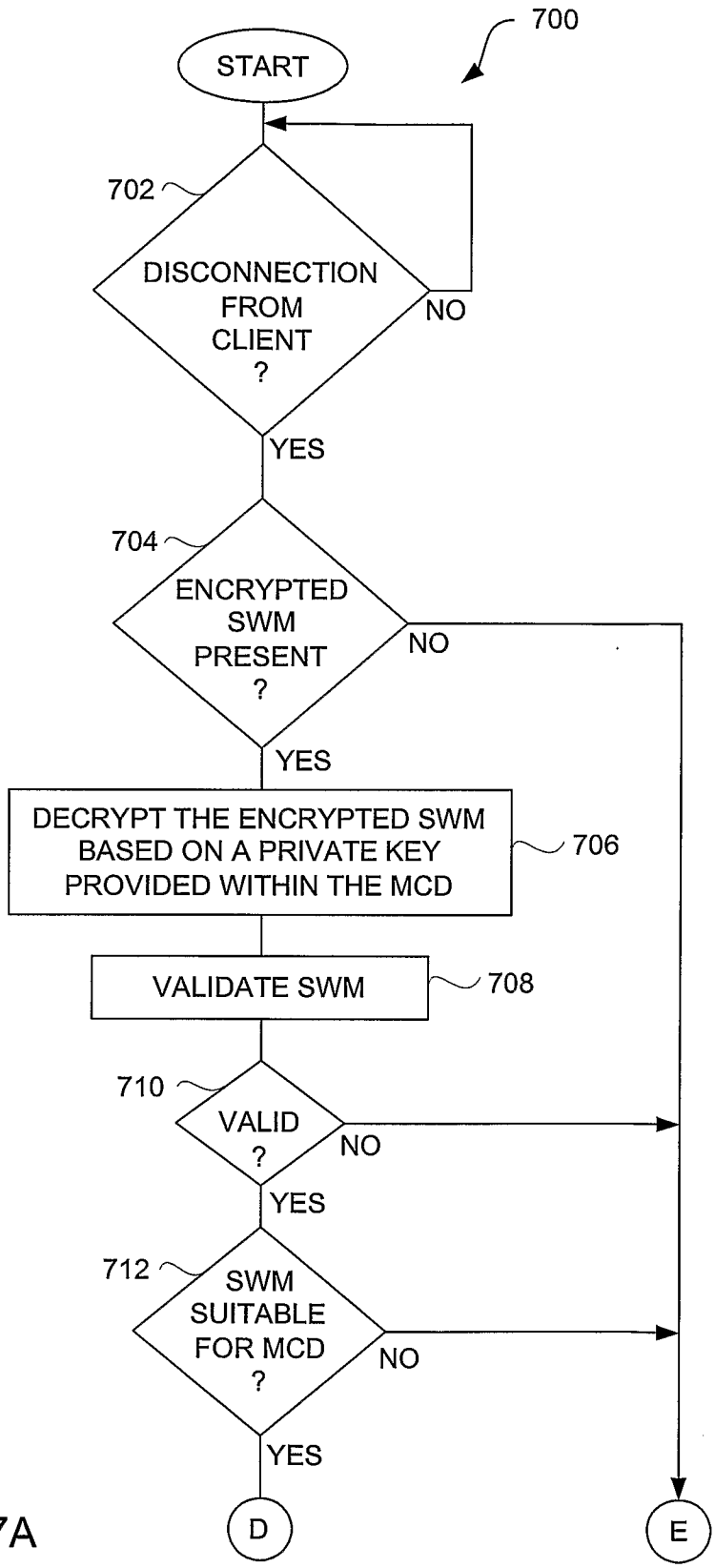

700
START
702
DISCONNECTION FROM CLIENT ?
NO
YES
704
ENCRYPTED SWM PRESENT ?
NO
YES
DECRYPT THE ENCRYPTED SWM BASED ON A PRIVATE KEY PROVIDED WITHIN THE MCD
706
VALIDATE SWM
708
710
VALID ?
NO
YES
712
SWM SUITABLE FOR MCD ?
NO
YES
D
E

FIG. 7A

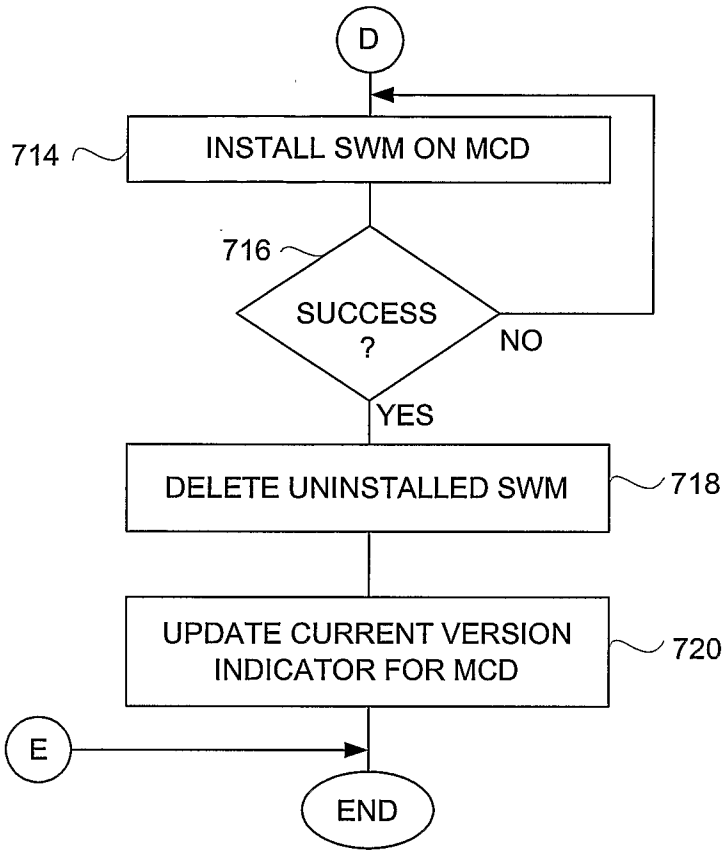

D
714
INSTALL SWM ON MCD
716
SUCCESS ?
NO
YES
DELETE UNINSTALLED SWM
718
UPDATE CURRENT VERSION INDICATOR FOR MCD
720
E
END

FIG. 7B