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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2018 (29.11.2018)

WIPO | PCT

(10) International Publication Number
WO 2018/217224 A1

(51) International Patent Classification:
G06F 21/10 (2013.01) *H04L 9/32* (2006.01)
G06F 21/12 (2013.01) *H04L 29/06* (2006.01)

(21) International Application Number:
PCT/US2017/052567

(22) International Filing Date:
20 September 2017 (20.09.2017)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
15/601,909 22 May 2017 (22.05.2017) US

(71) Applicant: **MACPAW, INC.** [US/US]; 5201 Great America Parkway, Suite 320, Santa Clara, California 95054 (US).

(72) Inventor: **KOSOVAN, Oleksandr**; C/o MacPaw, 5201 Great America Parkway, Suite 320, Santa Clara, California 95054 (US).

(74) Agent: **PLATI, Francis G.** et al.; c/o Procopio, Cory, Hargreaves & Savitch LLP, 525 B Street, #2200, San Diego, California 92101 (US).

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

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

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

(54) Title: SYSTEM AND METHOD FOR SOFTWARE ACTIVATION AND LICENSE TRACKING

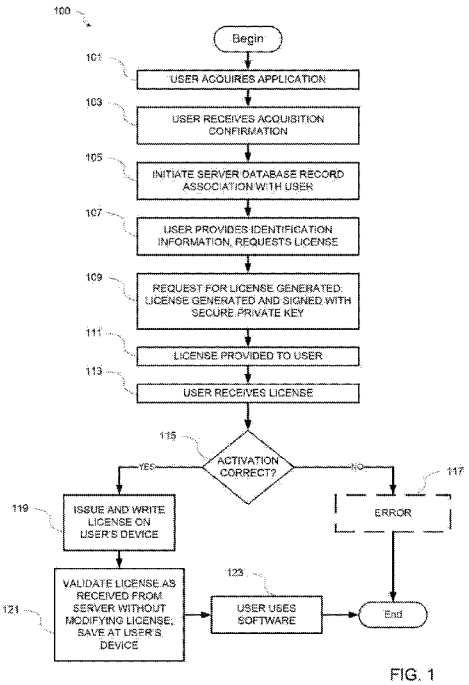

FIG. 1

(57) Abstract: System and method for software activation and further tracking of its states on an end-user computing device (computer) was developed to provide software developers a flexible and secure tool for software distribution and gathering statistics of usage of software activation. The method consists of the following logical steps: (a) obtaining an acquisition confirmation; (b) requesting for a license; (c) issuing and delivering the license to End User, the license being secured with a private key by a server, and the private key is not stored at the client; (d) verification of license on the User's computer; (e) storing the license on the User's computer; (f) periodic tracking of activation state, (g) another action with the User's license, wherein the verifying the license comprises validating, at the client associated with the application, the license received from the server without any modifications, and saving the license to the client.

WO 2018/217224 A1

SYSTEM AND METHOD FOR SOFTWARE ACTIVATION AND LICENSE TRACKING

BACKGROUND

[01] Technical Field

[02] The field of the present invention relates to systems and methods for software activation.

[03] Related Art

[04] Currently there are various related art methods of software activation. Most of the related art methods of software activation require an Internet connection. However, the use of an Internet connection in software activation can cause various problems and disadvantages. For example, but not by way of limitation, there is a problem of unauthorized (e.g., illegal) software usage. Unless the methods of software activation become more sophisticated than those of the related art, the related art problem of security and prevention of illegal software usage will remain in the field of software activation.

[05] Some related art approaches may use verification techniques for the license that include the client that is associated with the application generating a fake license file that includes incorrect parameter information, performing the verification of the license, and once the verification of the license has been completed, replacing the incorrect parameter information with correct parameter information. However, this approach may have various problems and disadvantages. For example but not by way of limitation, the fake license approach assumes that the content of a license file is created on the client's machine (not on the server side). Thus, a keygen app can create such a file the same way as a vendor app does, without any need of breaking the vendor app.

SUMMARY

[06] The present invention is directed toward the following aspects: simplifying the activation process for the end user; securing the methods of activation at each step of the activation process to prevent unauthorized (e.g., illegal) software usage; and tracking the activation state in the future to decrease the number of customer support inquiries related to some unexpected issues such as re-activation of software at a new computer, accidently lost activation number, etc.

[07] The first step in the activation process is acquiring an application. The acquiring of the application can occur through purchase, giveaway or any other method defined by the Vendor. A User must receive a confirmation of the acquisition of the application. This confirmation of the acquisition of the application can exist in a form of (but not limited to): (a) an activation key — e.g., a unique combination of symbols, that was provided by software Developer; or (b) a record in a database of the activation system, which is connected to the specific user.

[08] This acquiring of the application can be done in various ways. For example, but not by way of limitation, software may be purchased software at online store, delivery may occur on physical media including, but not limited to, CD, DVD or USB-drive, including software associated with an OEM supply set. Each of these modes of acquisition of the application has a distinct mechanism associated with setting a conformity of acquisition confirmation and also has a set of rules by which the license should be issued.

[09] The acquisition confirmation may include an activation number. From one side, the acquisition confirmation may be human-readable, which may decrease the number of possible errata. From the other side, the acquisition confirmation is long enough to prevent brute-force search. In addition, an activation-by-URL scheme has been developed that allows the Developer to provide users with an option to activate the

software with a user action, including (but not limited to) a selection or click action on a button on a web page or in an email.

[10] A record in the database of the activation system, which is associated with a specific user, can be initiated in one or more manners. For example, but not by way of limitation, the record in the database of the activation system may be initiated (a) after the purchase of software in an online or offline store; or (b) manually, by the administrator of the activation system. The user must provide identification information to obtain a software license. The identification information may include, but is not limited to: (a) a unique pair of a login identifier and a password; (b) establishing a secure connection to an activation server, which is protected with a security key provided by the activation server; or (c) a method defined by the administrator of the activation system, which allows for definite identification of the User.

[11] After initiation of the record in the database of the activation system, the next step is to generate a request for a license. This step requires an Internet connection. According to the present example implementation, a protocol is provided. Further, mechanics of how data is processed on the activation server is also provided. The request from the User's Computer may contain the following data, including, but not limited to, one or more of: (a) application bundle ID (e.g., product name in a specific format); (b) bundle version (e.g., product version); (c) additional set of data maintained for the current activation; (d) user's account identifier (e.g., email address); (e) two values for a unique computer identifier; (f) the acquisition confirmation; (g) User's first and last name; (h) a block of random data to increase security; (i) a password to open data on the Server side; and (j) language for an error message.

[12] Once the request for license has been generated, the third step is issuing and delivering the license to the User, the license being secured with a private key by a server, and the private key is not stored at the client. After the license request is

received, the Server issues a license if the activation request is correct, or else sends an error message. The server response includes the following information: (a) an error code (e.g., 0 in case of success); (b) an error message or an encrypted license; (c) a signature of license file or an error message, which is generated using the private key of the software.

[13] After the software license is delivered to the End User's computer, the software license should pass validation by validating, at the client associated with the application, the license received from the server without any modifications, and saving the license to the client. The license file is written to a specific place on the User's computer file system. The license validity is checked after the first launch of the application, during app installation, during the first launch after the trial period of the app is over, or in other appropriate situations, as would be understood by those skilled in the art. After verification of the license origin (e.g., validation by the public key) and confirmation of completeness has been completed, a user can continue to work with the software.

[14] The state of activation is periodically sent to the activation server. The rules of reporting may be defined by the developer. The activation state can be triggered depending on various conditions defined by developer, for example, re-activating the software if reactivations are allowed.

[15] The example implementation is also directed to the possibility of disabling a license, both remotely (e.g., from the activation server side) and directly (e.g., initiated by the User).

BRIEF DESCRIPTION OF THE DRAWINGS

[16] FIG. 1 shows an example process of the protocol including the client and the server according to an example implementation.

[17] FIG. 2 shows an example for a first protocol according to the example implementation.

[18] FIG. 3 shows an example for a second protocol according to the example implementation.

[19] FIG. 4 shows an example process of the protocol including the server according to an example implementation.

[20] FIG. 5 shows an example process of the protocol including the client according to an example implementation.

[21] FIG. 6 shows an example environment suitable for some example implementations.

[22] FIG. 7 shows an example computing environment with an example computing device associated with the external host for use in some example implementations.

DETAILED DESCRIPTION

[23] The subject matter described herein is taught by way of example implementations. Various details have been omitted for the sake of clarity and to avoid obscuring the subject matter. The examples shown below are directed to structures and functions for software activation and license tracking.

[24] The terms used herein and understood to have their plain and ordinary meaning to those skilled in the art. Terms may be defined by way of example, but are not limited to the definition as provided. For example, but not by way of limitation, a “User” or “End User” may include an entity, such as a person, who has acquired a copy of a software in an authorized manner (e.g., legally), and has configured the software to properly operate on a computing device. Further, a “Developer” or a “Vendor” may be an entity, such a person or legal entity that distributes one or more copies of software, such

as a logically completed computer program. Additionally, a “software installation” is a copy of a computer program that has been placed on a User computing device associated with a user, in the way that makes the computer program operate in a manner that was intended by the Developer.

[25] Additionally, “Activation” may include a procedure or set of procedures that confirms the legality of software installation (e.g., whether the software installation was authorized), and removes any functional limitations designed by Developer (if they were present). “Trial” may include a type of software installation usage with the limited functionality made for User to become acquainted with the software. Also, “Acquisition Confirmation” is the digital entity which confirms a right of the User to use software legally (e.g., an authorization), and must be provided to the User after the acquisition process (e.g., a purchase, giveaway, or the like) has been completed.

[26] In addition, “Activation number”, or “activation key” may be a human-readable combination (e.g., alphanumeric symbols) which is used as a type of Acquisition confirmation. Further, a “License” may include a computer file that contains data used to identify whether the software installation can be used on a specific computer device. A “License Type” may include a set of rules for license generation, which are set up by the Developer to create different conditions of software usage. Those rules may include, but are not limited to, the period of software usage (e.g., 1-year, 6 month, lifetime), a number of possible re-activations, certain functionality to be provided, etc.

[27] Further, “Activation Server” or “Server” is a remote computing device with respect to the computing device of the User, and which performs necessary operations for software installation activation. According to the example implementations, the server may include hardware that is known to those skilled in the art. Also, “Bundle ID” is an application identifier in a special format that is readable by the operating system of End User’s computer.

[28] Aspects of the example implementations are directed to simplifying the activation process for an end user; securing the methods of activation at each step of the activation process to prevent unauthorized (e.g., illegal) software usage; tracking the activation state in the future to decrease the number of customer support inquiries related to some unexpected issues such as re-activation of software at a new computer, accidently lost activation number, etc.

[29] To overcome the aforementioned related art security gap, the example implementations are directed to the asymmetric scheme of a digital signature. More specifically, the example implementations use a unique pair of keys for each vendor application: a public key, which is integrated in every vendor app, and the private key, which is stored on the server side securely. To secure the private key that is located at the server, the example implementation may use various techniques and methods, including (but not limited to) isolated storage systems, available to the internal servers only, wherein the servers are not connected to the internet, using of SSL for establishing encrypted connections, and using AWS Virtual Private Cloud service for provisioning virtually isolated section of the Cloud.

[30] **Using network firewalls for protection from illegal access.**

[31] When the server receives a corresponding request, the server generates the content of a license file, and signs the license file with the unique, secured private key. Thus, the server creates a digital signature. The signed license file with the unique, secured private key is specific for every vendor app.

[32] The vendor app receives the license file with the digital signature and stores it on the machine of the user (e.g., client). If a bad actor (e.g., hacker) attempts a malicious act (e.g., decides to attempt to change the license file or attempts to generate a fake license), the signature validation of this license file fails on the vendor app side.

Thus, the content of the license file becomes invalid, and the vendor app withstands the malicious attempt (e.g., to crack the vendor app).

[33] To accomplish these and other aspects, various steps may be performed in a process 100, including, but not limited to, the activation process, as shown in FIG.1. At 101, a user acquires an application. Further, at 103, a User receives a confirmation of the acquisition of the application by one of various modes. Each of the modes of acquisition of the application has a distinct mechanism associated with setting a conformity of acquisition confirmation and a set of rules by which the license should be issued. The acquisition confirmation may include an activation number and may be human-readable, but is long enough to prevent brute-force search by an unauthorized party to determine the activation confirmation. Further, an activation-by-URL scheme according to the example implementation may permit the Developer to provide users with an option to activate the software with a user action.

[34] A record in the database of the activation system associated with a specific user can be initiated at 105. The user must provide identification information to obtain a software license at 107. After initiation of the record in the database of the activation system, a request for a license is generated by a protocol at 109. According to the protocol, the license is generated at the server, and signed with a secure private key. Once the request for license has been generated at 109, the license is provided to the User at 111, and the user receives the license at 113 with the digital signature, and stores the license file at the user device.

[35] After the license request is received as noted above, the Server determines whether the activation is correct at 115, and issues a license at 119 if the activation request is correct, or else sends an error message at 117. At 119, after the software license is delivered to the End User's computer, it should pass validation. The license file is written to a specific place on the User's computer file system. As the

application runs, the license validity is checked at 121, without modifying the license as received from the server; it is noted that if an attempt is made to change the signed license file or generate a fake license, this attempt will fail at the vendor side, the content of the license file will become invalid, such that the vendor app is not cracked. After verification of the license origin and confirming completeness, the software installation is complete, and the software is ready to use at 123. The state of activation is periodically sent to the activation server. Further, a license may be disabled remotely or directly.

[36] Further details of the various operations are explained below in greater detail.

[37] **Obtaining an Acquisition Confirmation**

[38] To use software in an authorized manner (e.g., legally), a User must receive an acquisition confirmation. Such an acquisition confirmation may exist in various forms, including, but not limited to: (a) an activation key (e.g., a unique combination of symbols, that was provided by software Developer, for example); or (b) a record in the database of the activation system, which is connected with the specific user. After completing the acquisition, the User must be provided with instructions on how to exchange the acquisition confirmation for a software license.

[39] As explained above, the user must obtain an acquisition confirmation (e.g., activation number). A software vendor may have one or more different options of distribution of the activation number:

[40] direct sales (e.g., direct via online sale on vendor website, or through direct vendor retail outlet);

[41] sales via third-party resellers and partners;

[42] distribution either via online or box version on a physical media (e.g., CD, USB-sticks or SD-cards), etc.

[43] The foregoing distribution methods may require different ways and sequences of obtaining an acquisition confirmation. In some cases, the acquisition confirmation can be provided on demand, as soon as the payment received; in other cases, it may be necessary to provide a bulk number of pre-generated activation numbers to be delivered on some physical media. In each case, the Vendor must know more information associated with the conditions (e.g., partner, marketing campaign, data, count and other conditions) under which the activation numbers were generated (e.g., for marketing and analytics purposes).

[44] The example implementation includes at least three basic mechanisms of acquisition confirmation generation:

[45] generation by URL — each request for the acquisition confirmation will be redirected onto specific URL, which is associated with the activation server. In case of a complete and correctly constructed request, the Server will respond with an acquisition confirmation;

[46] 'bulk' generation — the Server generates a finite amount of activation numbers, which are provided 'as is' to a software distributor or are spread throughout users;

[47] single generation — this method of generating an acquisition confirmation is mostly used for testing and support purposes, such as when it is necessary to generate one acquisition confirmation.

[48] Acquisition confirmations may be sent by server 'as is', with no encryption.

[49] Generation by URL

[50] This method includes a request for one or more acquisition confirmations. This method is most likely to be used during direct online purchases, but may be used during other modes of obtaining the application. FIG. 2 illustrates an example implementation 200 of the process of requesting an activation code based on generation

by URL. In FIG. 2, a user workstation 201, a payment system 203, and an activation server 205 are disclosed. However, other implementations may be substituted therefor with departing from the inventive scope, as would be understood by those skilled in the art.

[51] 1. User visits the purchase page (e.g., the developer's website, or at 3rd-party web-resource, or some either way of purchasing the product). (207)

[52] 2. User provides required information, such as his first/last name, e-mail address, phone number, quantity of products, duration of software usage, etc. (207)

[53] 3. If the purchase is successful, billing system sends collected information by a specific API to the server of software Vendor; the order information is saved at the activation server and an activation key is generated.(209, 211a, 211b)

[54] 4. User obtains an activation key from the activation server, which is sent to the email stated and/or displayed on a purchase page. (213, 215)

[55] 5. If payment for some reason cannot be completed, the user is informed about inability to complete the transaction, along with instructions for further actions (re-enter billing information, check bank account, try later, etc.). (217)

[56] Bulk generation

[57] In another example implementation, 'bulk' activations may be employed, to provide many codes for partners: affiliates, resellers or producers of physical media (CDs, USBs).

[58] In this example implementation, a software vendor can generate certain amount of activation numbers (e.g., in advance) with required marketing parameters or the conditions of contract (e.g., 500 licenses are provided to Resell LLC to be used till December, 25 2017).

[59] In this example implementation, a database having a blank record of order is created, and developer can state the specific pattern of the activation number. The

application will ask the end user for his/her personal data (name and email address). Thus, the order record will be filled after the software has been activated.

[60] **Single Generation**

[61] This option may be implemented for customer support, as well as promotional or test purposes.

[62] According to this example implementation, the software vendor generates an activation code via the special administrator's panel on the server, and then sends this code to the customer or required party. The order record is then filled by the administrator.

[63] **Creating a Record in Activation System's Database**

[64] Operations for creating a record in the activation system's database after software acquisition are described as follows. It is noted that the User should identify himself or herself on the activation system. This can be done by the one or more of the following methods (but not limited to):

[65] by providing a unique pair of login and password information;

[66] by establishing a secure connection with the activation server, protected with a security key (e.g., encrypted) as provided by the activation server;

[67] by establishing a connection with activation server and sending the parameter, which allows for definitive identification of the User (e.g., by providing session identifier from the third-party service, or any other parameter, defined by activation system administrator).

[68] After User identifies himself or herself on the activation system, the information about software acquisition will be stored as a record in the activation system database.

[69] **Issuing a License**

[70] After the User receives an acquisition confirmation, he or she must request a license to use the software without limitations.

[71] The process of issuing a license is similar for each type of acquisition confirmation. The difference is in the ways the User provides the acquisition confirmation for receiving the license:

[72] by entering the activation number inside the application;

[73] by activating the application by the URL scheme;

[74] by authenticating on the activation system:

[75] by providing a unique pair of login and password;

[76] by establishing a secure connection with activation server, protected with a security key previously provided;

[77] by establishing a connection with activation server and sending the parameter, which allows for the definitive identification of the User (e.g., by providing session identifier from the third-party service, or any other parameter, defined by activation system administrator).

[78] The process 300 of the license request is disclosed as follows, and is illustrated in FIG. 3. As shown in FIG. 3, a user computing device (e.g., workstation) 301, application-side operation space 303 (e.g., user-controlled devices) and server-side space (e.g., server having a database) 305 are provided. Operations include, but are not limited to, the following:

[79] 1. The User provides an acquisition confirmation by one of the methods described above. (307)

[80] 2. Software generates a license request and sends the license request to an activation server. (308)

[81] 3. The activation server checks if activation is allowed for this acquisition confirmation (e.g., the activation server may search for an 'active' parameter in activation number record in database). (309)

[82] 4. If yes (i.e., activation is allowed), the activation server issues a license based on the identification information (311) and transmits a response to the application containing a License file. (317)

[83] 5. The license file passes the validation by an application-side part of activation system.(319)

[84] 6. The User is informed that the activation was successful. (321)

[85] 7. If activation server determines that an error occurred during activation process (e.g., not allowed to issue a license at 309, and the license was not issued before at 313), the activation server sends back an error code and message to the application, and hence, the user. (315) On the other hand if the activation server determines that it is not allowed to issue a license at 309, but that the license was issued before at 313, then the process continues to send the license as previously issued in 317, as explained above.

[86] 8. The software informs the user about the error type with instructions on further actions to address the error (e.g., check the validity of activation code, contact customer support, etc.). (323)

[87] To decrease the number of requests to the server and to simplify the User's input requirements, the pattern recognition system for activation keys was implemented according to the present example implementation. More specifically, that activation keys are generated using a specific pattern, which will be recognized by software installation, and which will trigger the software installation for the specific actions. For example, but not by way of limitation, the activation key with a specific suffix can initiate an appearance of a form for requesting personal data from the user. The

vendor can set up the specific patterns and software installation behavior to match their own patterns and behavior.

[88] After the software is registered on Activation server, the activation server issues a pair of private and public keys for the signing of the license. The private key is stored on the server and is used for signing each license before the license is sent to the application. The public key is integrated to the application-side part of the activation system, and is used later for verifying the license signature, as explained further below.

[89] After the successful activation request has occurred, a license is generated on the activation server. As one of its parameters, the license includes hash that is generated based on the following application identifiers:

[90] bundle ID of application;

[91] one or more identifiers of the User's computing device (e.g., hardware), which can be (but is not limited to) a username that is used for authenticating the User on operating system; a MAC-address of network interface card; hard disk drive or solid-state drive; a motherboard identifier, or other identifier as may be known by those skilled in the art;

[92] and the randomly generated string (e.g., 'salt'), which will be included to the license.

[93] Before sending the license file to end user, the license file is encrypted with a symmetric encryption algorithm, and signed with the server signature based on the private key issued during software registration on the Activation server.

[94] **Reactivation**

[95] To decrease the number of support inquiries caused by the necessity to continue software usage after changing hardware (e.g., changing to a new computer), the re-activation system according to the example implementation has been implemented.

[96] When the activation number is issued, the activation number has the finite number of possible activations. For example, if the user for some reason deactivates the software installation on his/her computer device, and uses the same activation number to activate the software on that same computer device again, the old license will be issued.

[97] On the other hand, if the user changes the computing device, he/she still has a capability to activate software using the same activation number. The information about the number of possible reactivations is stored on the activation server. The activation server automatically adds one additional reactivation once a specified period of time has passed (e.g., six months by default, but not limited thereto). As a result, the user will be able to reactivate the software in case he or she changes the computing device or operating system.

[98] **Validation of License**

[99] To make the software installation fully functional, it is necessary to check that the license is available on User's computer, and verify its validity. The verification consists of the checks of the following parameters: (a) license origin (e.g., if the license was issued by the correct server); (b) if license was issued to this computer (e.g., if the license was issued to the correct computer); and (c) if license is not expired.

[100] In the related art, the verification process is a period during which a hacker may be able to intrude into the code and receive information necessary to crack an application. To address this risk, the example implementation includes several mechanisms.

[101] At the time points where a license check is necessary (these time points may be defined by the developer, for example, as a launch of application, an upgrade, or for performing some particular action), but the license check has not yet been performed, the application-side of activation system validates the license, as it was

received from the server, without modifying the license, and saves the validated license at the client side, or user, machine. The unauthorized party (e.g., hacker) is prevented from obtaining data, in the case of an inner intrusion.

[102] More specifically, and as explained above, to overcome the aforementioned related art security gap, the example implementations are directed to the asymmetric scheme of a digital signature. More specifically, the example implementations use a unique pair of keys for each vendor application: a public key, which is integrated in every vendor app, and the private key, which is stored on the server side securely.

[103] When the server receives a corresponding request, the server generates the content of a license file, and signs the license file with the unique, secured private key. Thus, the server creates a digital signature. The signed license file with the unique, secured private key is specific for every vendor app.

[104] The vendor app receives the license file with the digital signature and stores it on the machine of the user (e.g., client). If a bad actor (e.g., hacker) attempts a malicious act (e.g., decides to attempt to change the license file or attempts to generate a fake license), the signature validation of this license file fails on the vendor app side. Thus, the content of the license file becomes invalid, and the vendor app withstands the malicious attempt (e.g., to crack the vendor app).

[105] Accordingly, there is no need to prevent the hacker from obtaining the license data, because it is not possible for this to occur, as explained above. The license data is uniquely generated for each machine (e.g., using machine specific identifications like a network MAC address or serial number) and is signed by the private key stored on the server. Therefore, the license data cannot be used or accessed on another machine.

[106] As the license check is required, the application-side part of the activation system is checking the pre-defined storages to determine if the license is available.

[107] At first, the application-side part of the activation systems checks the server signature using the public key that was integrated to the application-side part of application system, as explained above. If the server signature is valid based on the use of the public key, the activation system receives a key for symmetric decryption. The second part of license is then decrypted using this decryption key. As the result of successful decryption, the license information is received.

[108] The next stage is checking the validity of license for the specific computer. In order to perform this check, the application-side of activation system generates a hash string by using the following parameters:

[109] bundle ID of application

[110] one or more identifiers of a computing device

[111] 'salt' string that is stored in license.

[112] The generated hash string should match the string that is stored in license. If there is not a match, then it is determined that the license is not valid for that particular computer and/or for that particular application.

[113] As a next step, the application-side part of activation system checks to confirm that the license is not expired by comparing the license parameters with the current system and application states. Depending on the license type, the following parameters may be compared:

[114] a) license expiration time with current system time;

[115] b) license subscription end time with current system time;

[116] c) license expiration version with current application version;

[117] d) license beta-only flag with current application version.

[118] Additionally, the application-side part of the activation system may periodically transmit a request to the activation server to validate the license. The request may include an activation identifier and information that is necessary to identify

the user's workstation. If the license is still determined to be valid, the server responds with the same license. However, if the license is determined to not still be valid, the server responds with an error message. This method prevents manipulation by using the system time of the user's workstation to extend license duration, as the expiration time is validated by the server system time.

[119] Application-side Part Security

[120] The application-side part of activation system is generated uniquely for each application that is registered at the Activation server. For security purposes, the functions, methods and variables of the programming code of the application-side part of activation system are obfuscated (e.g. are made not-human readable).

[121] The functions to be obfuscated are marked with special markers, and during compilation their names are replaced to random ones from the pre-defined alphabet (usually, it contains alphanumeric symbols: capital and small letters and digits). For each application, the obfuscated names are unique.

[122] Another mechanism for increasing security is by using functions which change the destination address in memory during runtime. More specifically, during the compilation of program code the functions that are used for license verification return intentionally wrong parameters, but during software is in live operation, those functions are redirected to the address in memory, which contains functions that will return the correct parameters. Those functions are named randomly during compilation, so that in case of reverse engineering, software hackers will not be able to determine what those functions actually do. This mechanism may also be implemented at the points when the license check is required as defined by the Developer.

[123] According to the example implementation, the developer is not provided with the name of the function during operation, and need not have this information. For example, but not by way of limitation, the developer may write the software program (e.g.,

code), and then, when the application is compiled, the functions are named pseudo-randomly (i.e., obfuscated), so that it is not possible to determine the function based on the name of the function, as a security measure.

[124] **Remote Disabling of License**

[125] If the User requests a refund and the refund request is granted, the user loses the right to use the software. Thus, it is necessary to have a mechanism of remotely revoking the license. This mechanism is also used for handling the subscription tariff plans, when the user needs to pay for the software on a monthly or yearly basis. For example, a user may simply choose to not renew the software without providing an affirmative indication of cancellation.

[126] If the license is marked on the server as not being valid due to the refund or an unpaid subscription (e.g., renewal), the license will be also marked on the application-side part of activation system, and the user will not be able to use the application until the payment has been confirmed.

[127] **Disabling License on the Application Side**

[128] Additionally, the Developer may be able to add functionality to disable the license inside the application. After the User initiates the disabling of the license, the application-side of activation system sends a request to the server side. Accordingly, the server responds with the license for that particular workstation, which is marked as disabled. This procedure is useful in case the User wants to use the software on another workstation. The user can initiate disabling of the license on the first workstation, and repeat steps necessary to activate the application on the second (e.g., another) workstation.

[129] **Using App Store Receipt as License**

[130] If the vendor sells application (e.g., both via Apple App Store and via other source), the activation server has the ability to match the App Store Receipt as the license.

[131] For example, but not by way of limitation, it user purchased application via an app store (e.g., Apple App Store), the information about the purchase is sent to the activation server via the public API. If user needs to reactivate the software on his/her computing device, the software installation will be recognized as an activated one, and the license will be issued automatically.

[132] **Example Process**

[133] FIG. 4 illustrates a process 400 as performed on the server according to the example implementation, and including the foregoing disclosures of FIGS. 1-3. At 401, after a user has acquired an application and received a confirmation of the acquisition of the application by one of various modes, a record in the database of the activation system associated with a specific user is initiated at the server side. Further, the server receives identification information and a license request from the user to obtain a software license at 403. After initiation of the record in the database of the activation system, a request for a license is generated by a protocol at 405. According to the protocol, the license is generated at the server, and signed with a secure private key. Once the request for license has been generated at 109, the license with the digital signature is provided to the User at 407, and the license file is stored at the user device.

[134] After the license request is received as noted above, the Server determines whether the activation is correct at 409, and issues a license at 413 if the activation request is correct, or else sends an error message at 411. At 413, after the software license is delivered to the End User's computer, it should pass validation at 415, without modifying the license as received from the server; it is noted that if an attempt is made to change the signed license file or generate a fake license, this attempt

will fail at the vendor side, the content of the license file will become invalid, such that the vendor app is not cracked. The license file is written to a specific place on the User's computer file system. As the application runs, the license validity is checked at 415. After verification of the license origin and confirming completeness, the software installation is complete, and the software may be used by the user. The state of activation is periodically received by the activation server. Further, a license may be disabled remotely or directly by the activation server.

[135] FIG. 5 illustrates a process 500 as performed on the server according to the example implementation, and including the foregoing disclosures of FIGS. 1-3. At 501, a user acquires an application. Further, at 503, a User receives a confirmation of the acquisition of the application by one of various modes.

[136] As explained above, each of the modes of acquisition of the application has a distinct mechanism associated with setting a conformity of acquisition confirmation and a set of rules by which the license should be issued. The acquisition confirmation may include an activation number and may be human-readable, but is long enough to prevent brute-force search. Further, an activation-by-URL scheme according to the example implementation may permit the Developer to provide users with an option to activate the software with a user action.

[137] At 505 the user provides identification information and a request to obtain a software license. Once the request for license has been generated by the server with the digital signature as explained above, the user receives the license at 507, and the license file is stored at the user device.

[138] After the license request is received at 507, the Server determines whether the activation is correct, and the user receives an issuance of the license at 513 if the activation request is correct, or else receives an error message at 511. At 513, after the software license is received by the End User's computer, it should pass

validation. The license is written on the user's device without modifying the license as received from the server; it is noted that if an attempt is made to change the signed license file or generate a fake license, this attempt will fail at the vendor side, the content of the license file will become invalid, such that the vendor app is not cracked The license file is written to a specific place on the User's computer file system. As the application runs, the license validity is checked. After verification of the license origin and confirming completeness, the software installation is complete, and the software is ready to use at 515. The state of activation is periodically sent to the activation server. Further, a license may be disabled remotely or directly by the server.

[139] In some examples, the foregoing processes illustrated in FIGS. 1-5 may be implemented with different, fewer, or more blocks. The processes may be implemented as computer executable instructions, which can be stored on a medium, loaded onto one or more processors of one or more computing devices, and executed as a computer-implemented method.

[140] **Example Environment**

[141] FIG. 6 shows an example environment suitable for some example implementations. Environment 600 includes devices 605-645, and each is communicatively connected to at least one other device via, for example, network 660 (e.g., by wired and/or wireless connections). Some devices may be communicatively connected to one or more storage devices 630 and 645.

[142] An example of one or more devices 605-645 may be computing devices 705 and/or 805 described below in FIGS. 7 and 8, respectively. Devices 605-645 may include, but are not limited to, a computer 605 (e.g., a laptop computing device), a mobile device 610 (e.g., smartphone or tablet), a television 615, a device associated with a vehicle 620, a server computer 625, computing devices 635-640, storage devices 630 and 645.

[143] In some implementations, devices 605-620 may be considered user devices 625-645 may be devices associated with a server as described above and with respect to FIGS. 3-5.

[144] For example, by not by way of limitation, a user having user device 605 or 610 on a network supported by one or more devices 625-645, may have perform the acquire an application, request a license, receive a license, and use the software, using user device 605 or 610. The server may perform the above-described operations using devices 625-645, in accordance with the processes described above with respect to FIGS. 1-5.

[145] **Example Computing Environment**

[146] FIG. 7 shows an example computing environment with an example computing device associated with the external host for use in some example implementations. Computing device 705 in computing environment 700 can include one or more processing units, cores, or processors 710, memory 715 (e.g., RAM, ROM, and/or the like), internal storage 720 (e.g., magnetic, optical, solid state storage, and/or organic), and/or I/O interface 725, any of which can be coupled on a communication mechanism or bus 730 for communicating information or embedded in the computing device 705.

[147] Computing device 705 can be communicatively coupled to input/user interface 735 and output device/interface 740. Either one or both of input/user interface 735 and output device/interface 740 can be a wired or wireless interface and can be detachable. Input/user interface 735 may include any device, component, sensor, or interface, physical or virtual, that can be used to provide input (e.g., buttons, touch-screen interface, keyboard, a pointing/cursor control, microphone, camera, braille, motion sensor, optical reader, and/or the like). Output device/interface 740 may include a display, television, monitor, printer, speaker, braille, or the like. In some example

implementations, input/user interface 735 and output device/interface 740 can be embedded with or physically coupled to the computing device 705. In other example implementations, other computing devices may function as or provide the functions of input/user interface 735 and output device/interface 740 for a computing device 705.

[148] Examples of computing device 705 may include, but are not limited to, highly mobile devices (e.g., smartphones, devices in vehicles and other machines, devices carried by humans and animals, and the like), mobile devices (e.g., tablets, notebooks, laptops, personal computers, portable televisions, radios, and the like), and devices not designed for mobility (e.g., desktop computers, other computers, information kiosks, televisions with one or more processors embedded therein and/or coupled thereto, radios, and the like).

[149] Computing device 705 can be communicatively coupled (e.g., via I/O interface 725) to external storage 745 and network 750 for communicating with any number of networked components, devices, and systems, including one or more computing devices of the same or different configuration. Computing device 705 or any connected computing device can be functioning as, providing services of, or referred to as a server, client, thin server, general machine, special-purpose machine, or another label.

[150] The I/O interface 725 may include wireless communication components (not shown) that facilitate wireless communication over a voice and/or over a data network. The wireless communication components may include an antenna system with one or more antennae, a radio system, a baseband system, or any combination thereof. Radio frequency (RF) signals may be transmitted and received over the air by the antenna system under the management of the radio system.

[151] I/O interface 725 can include, but is not limited to, wired and/or wireless interfaces using any communication or I/O protocols or standards (e.g., Ethernet,

802.11x, Universal System Bus, WiMax, modem, a cellular network protocol, and the like) for communicating information to and/or from at least all the connected components, devices, and network in computing environment 700. Network 750 can be any network or combination of networks (e.g., the Internet, local area network, wide area network, a telephonic network, a cellular network, satellite network, and the like).

[152] Computing device 705 can use and/or communicate using computer-usable or computer-readable media, including transitory media and non-transitory media. Transitory media include transmission media (e.g., metal cables, fiber optics), signals, carrier waves, and the like. Non-transitory media include magnetic media (e.g., disks and tapes), optical media (e.g., CD ROM, digital video disks, Blu-ray disks), solid state media (e.g., RAM, ROM, flash memory, solid-state storage), and other non-volatile storage or memory.

[153] Computing device 705 can be used to implement techniques, methods, applications, processes, or computer-executable instructions in some example computing environments. Computer-executable instructions can be retrieved from transitory media, and stored on and retrieved from non-transitory media. The executable instructions can originate from one or more of any programming, scripting, and machine languages (e.g., C, C++, C#, Java, Visual Basic, Python, Perl, JavaScript, and others).

[154] Processor(s) 710 can execute under any operating system (OS) (not shown), in a native or virtual environment. One or more applications can be deployed that include logic unit 760, application programming interface (API) unit 765, input unit 770, output unit 775, acquisition confirmation and user request review unit 780, license issuance determination 785, license generation unit 790, and inter-unit communication mechanism 795 for the different units to communicate with each other, with the OS, and with other applications (not shown). For example, acquisition confirmation and user request review unit 780, license issuance determination 785, and license generation

unit 790 may implement one or more processes shown in FIGs. 1-5. The described units and elements can be varied in design, function, configuration, or implementation and are not limited to the descriptions provided.

[155] In some example implementations, when information or an execution instruction is received by API unit 765, it may be communicated to one or more other units (e.g., logic unit 760, input unit 770, output unit 775, acquisition confirmation and user request review unit 780, license issuance determination 785, and license generation unit 790).

[156] For example, after input unit 770 has received input from a user, such as an instruction to acquire an application, a license request, or other such user interaction, input unit 770 may use API unit 765 to communicate the data acquisition confirmation and user request review unit 780. For example, acquisition confirmation and user request review unit 780 may make a determination regarding whether a license should be issued, as explained above.

[157] Acquisition confirmation and user request review unit 780 may, via API unit 765, interact with the license issuance determination 785 to provide an output as to whether a license should be issued. Using API unit 765, acquisition confirmation and user request review unit 780 may interact with license generation unit 790 to generate the license and provide the license to the user, as explained above.

[158] In some instances, logic unit 760 may be configured to control the information flow among the units and direct the services provided by API unit 765, input unit 770, output unit 775, acquisition confirmation and user request review unit 780, license issuance determination 785, and license generation unit 790 in some example implementations described above. For example, the flow of one or more processes or implementations may be controlled by logic unit 760 alone or in conjunction with API unit 765.

[159] The example implementations may have various benefits and advantages. For example, but not by way of limitation, increased security is provided during the license activation and verification process, to reduce the risk of a hacker obtaining sensitive information or using the application, as well as the risk of an information leakage to a developer of the software.

[160] Although a few example implementations have been shown and described, these example implementations are provided to convey the subject matter described herein to people who are familiar with this field. It should be understood that the subject matter described herein may be implemented in various forms without being limited to the described example implementations. The subject matter described herein can be practiced without those specifically defined or described matters or with other or different elements or matters not described. It will be appreciated by those familiar with this field that changes may be made in these example implementations without departing from the subject matter described herein as defined in the appended claims and their equivalents.

CLAIMS

1. A method for software activation and license tracking, comprising:

(a) obtaining an acquisition confirmation associated with an acquisition of the software, and providing the acquisition confirmation to a server associated with software activation and licensing;

(b) requesting a license for a client to be authorized to use the software, wherein the license is secured with a private key by a server, and the private key is not stored at the client;

(c) issuing and providing the license to a client;

(d) verifying the license at the client;

(e) storing the license at the client;

(f) on a periodic basis, tracking an activation state of the license; and

(g) performing an action on the license at the client, based on the activation state of the license, wherein the verifying the license at the client comprises validating, at the client associated with the application, the license received from the server without any modifications, and saving the license to the client.

2. The method of claim 1, wherein the acquisition confirmation comprises at least one of an activation key and a record in a database associated with the server, and further wherein the server provides the user with an instruction to exchange the acquisition confirmation for the software license.

3. The method of claim 1, wherein the acquisition confirmation is obtained by one of direct sales, sales via a third-party reseller, and distribution via online or box version on a physical media.

4. The method of claim 1, wherein the acquisition confirmation is generated by at least one of (a) a uniform resource locator (URL) associated with the server that generates the acquisition conformation; (b) a bulk generation of a plurality of acquisition confirmations; and (c) a single generation of the acquisition confirmation.

5. The method of claim 4, wherein the acquisition confirmation generated by the URL comprises,

providing the user with webpage access for obtaining the acquisition confirmation,

receiving required information associated with the user;

for a purchase of the application being successful, providing the required information of the user to the server, wherein order information of the application is saved at the server, and the acquisition confirmation is generated by the server, and providing the acquisition information to the user; and

for a purchase of the application not being successful, providing the user with an indication that the purchase of the application was not successful.

6. The method of claim 1, wherein the acquisition confirmation is provided to the user without encryption.

7. The method of claim 1, further comprising creating a record in the server associated with the generation of the acquisition confirmation by one or more of providing a unique pair of login and password information, establishing a secure connection with the server protected with a security key, and establishing a connection with the server and sending a parameter that identifies the user, wherein after the user provides an identification, storing the information about the acquisition of the application in the server.

8. The method of claim 1, wherein the (c) and the (d) comprise:

Receiving the acquisition confirmation from the user;

The client Generating and providing the license request to the server;

The server determining if activation is permitted for the acquisition confirmation; and

For the activation being permitted, the server issuing a license based on the identification information of the user, and providing a file of the license to the user.

9. The method of claim 1, wherein the acquisition confirmation comprises one or more activation keys generated using a pattern having a specified suffix that is recognized by installation of the application, and will trigger appearance of a form for the user to provide information to the server, wherein a vendor of the application can determine the pattern based on its own pattern.

10. The method of claim 1, wherein the license is generated including parameters associated with at least one of a bundle ID of the application, one or more identifiers of the client, and a randomly generated string.

11. The method of claim 1, the verifying further comprising performing a verification of the license generated on the server, wherein the license is signed with the private key that is unique and secure to generate a digital signature, and the digital signature is further verified by a public key of the software at the client, the software comprising a vendor application.

12. A non-transitory computer-readable medium configured to perform operations for software activation and license tracking, the operations comprising:

(a) obtaining an acquisition confirmation associated with an acquisition of the software, and providing the acquisition confirmation to a server associated with software activation and licensing;

(b) requesting a license for a client to be authorized to use the software;

(c) issuing and providing the license to a client, wherein the license is secured with a private key by a server, and the private key is not stored at the client;

(d) verifying the license at the client;

(e) storing the license at the client;

(f) on a periodic basis, tracking an activation state of the license; and

(g) performing an action on the license at the client, based on the activation state of the license, wherein the verifying the license at the client comprises validating, at the client associated with the application, the license received from the server without any modifications, and saving the license to the client.

13. The method of claim 12, wherein the acquisition confirmation is generated by the URL by,

providing the user with webpage access for obtaining the acquisition confirmation,

receiving required information associated with the user;

for a purchase of the application being successful, providing the required information of the user to the server, wherein order information of the application is saved at the server, and the acquisition confirmation is generated by the server, and providing the acquisition information to the user; and

for a purchase of the application not being successful, providing the user with an indication that the purchase of the application was not successful.

14. The non-transitory computer readable medium of claim 12, wherein the acquisition confirmation is provided to the user without encryption.

15. The non-transitory computer readable medium of claim 12, further comprising creating a record in the server associated with the generation of the acquisition confirmation by one or more of providing a unique pair of login and password information, establishing a secure connection with the server protected with a security key, and establishing a connection with the server and sending a parameter that identifies the user, wherein after the user provides an identification, storing the information about the acquisition of the application in the server.

16. The non-transitory computer readable medium of claim 12, wherein the (c) and the (d) comprise:

Receiving the acquisition confirmation from the user;

The client Generating and providing the license request to the server;

The server determining if activation is permitted for the acquisition confirmation; and

For the activation being permitted, the server issuing a license based on the identification information of the user, and providing a file of the license to the user, and wherein the acquisition confirmation comprises one or more activation keys generated using a pattern having a specified suffix that is recognized by installation of the application, and will trigger appearance of a form for the user to provide information to the server, wherein a vendor of the application can determine the pattern based on its own pattern.

17. The non-transitory computer readable medium of claim 12, the verifying further comprising performing a verification of the license generated on the server,

wherein the license is signed with the private key that is unique and secure to generate a digital signature, and the digital signature is further verified by a public key of the software at the client, the software comprising a vendor application

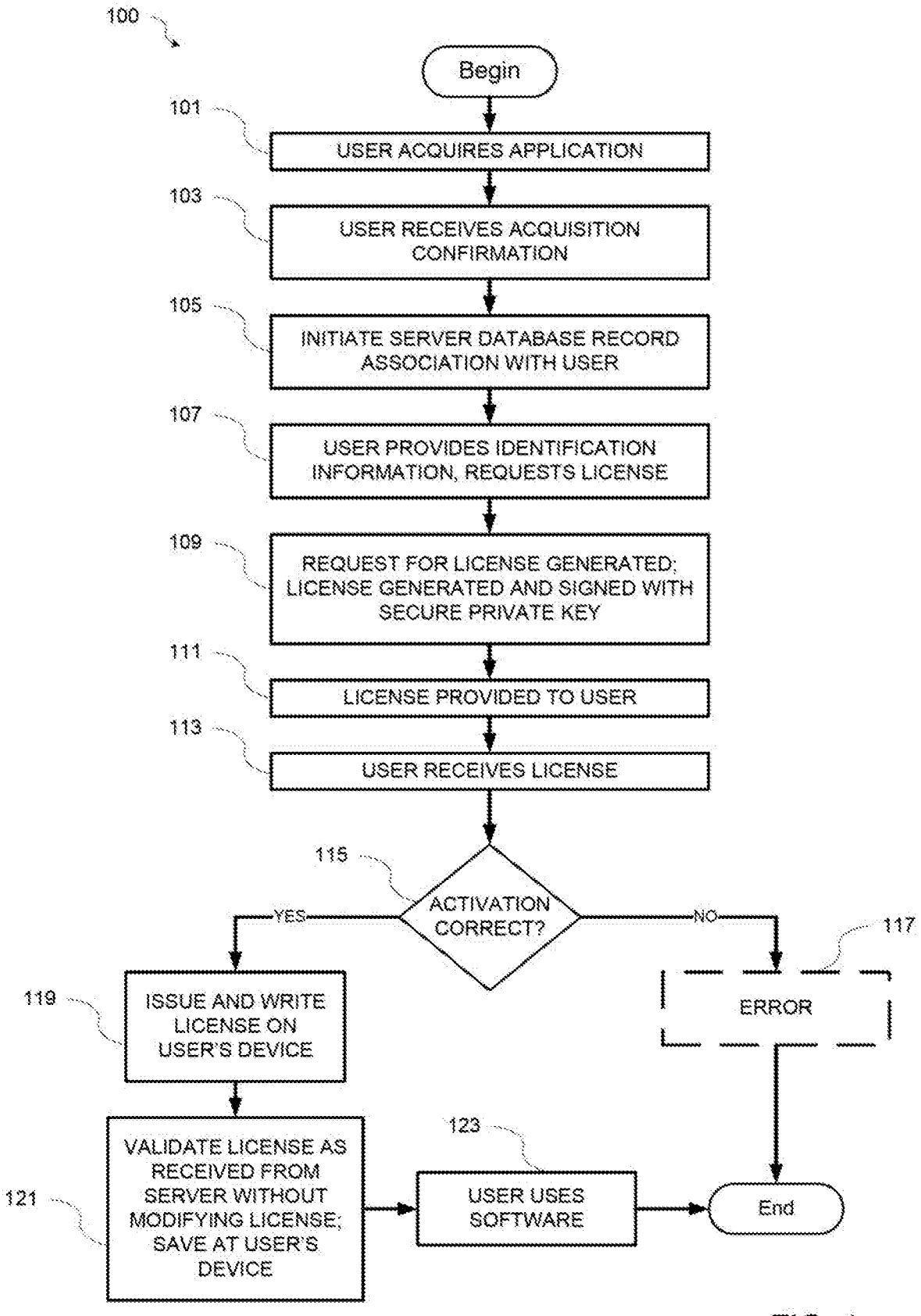
100
Begin
101
USER ACQUIRES APPLICATION
103
USER RECEIVES ACQUISITION CONFIRMATION
105
INITIATE SERVER DATABASE RECORD ASSOCIATION WITH USER
107
USER PROVIDES IDENTIFICATION INFORMATION, REQUESTS LICENSE
109
REQUEST FOR LICENSE GENERATED; LICENSE GENERATED AND SIGNED WITH SECURE PRIVATE KEY
111
LICENSE PROVIDED TO USER
113
USER RECEIVES LICENSE
115
ACTIVATION CORRECT?
YES
NO
117
ERROR
119
ISSUE AND WRITE LICENSE ON USER'S DEVICE
121
VALIDATE LICENSE AS RECEIVED FROM SERVER WITHOUT MODIFYING LICENSE; SAVE AT USER'S DEVICE
123
USER USES SOFTWARE
End

FIG. 1

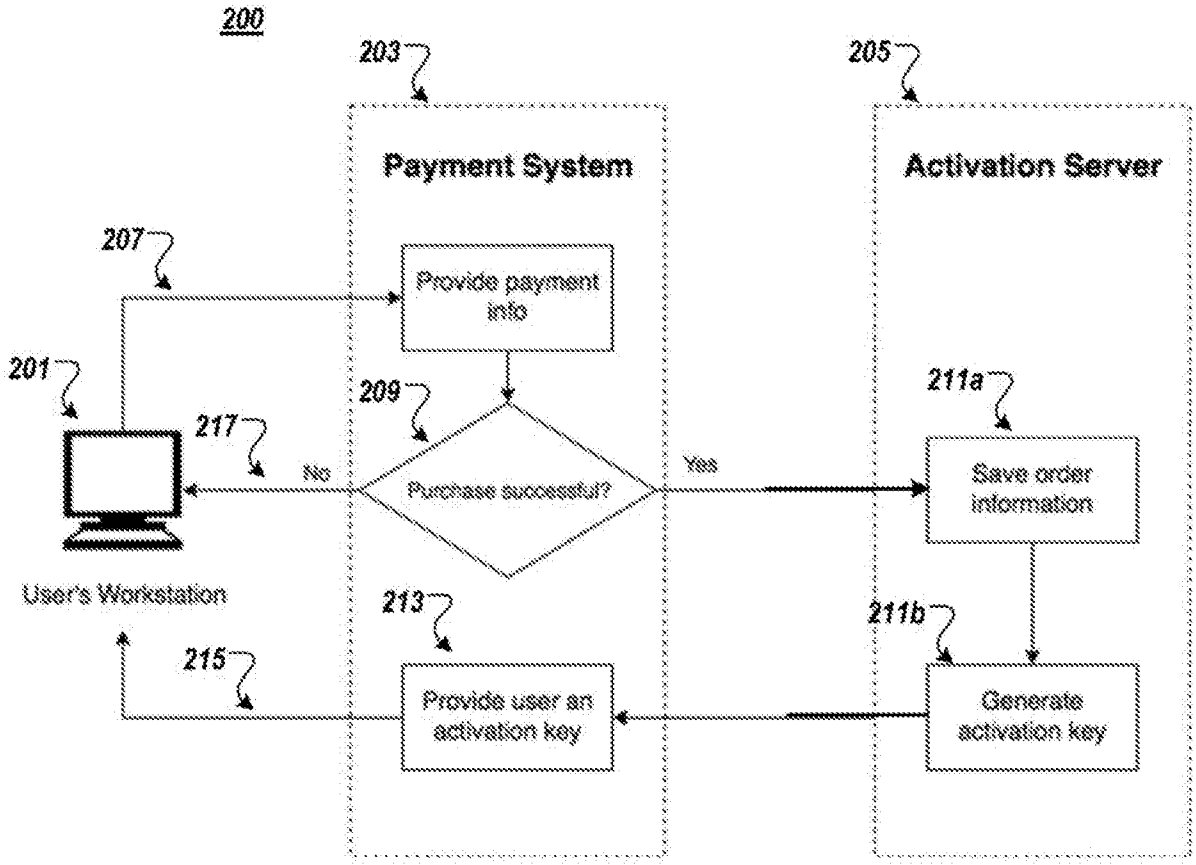
200
203
Payment System
205
Activation Server
207
Provide payment info
201
209
211a
217
No
Purchase successful?
Yes
Save order information
User's Workstation
213
211b
215
Provide user an activation key
Generate activation key

FIG. 2

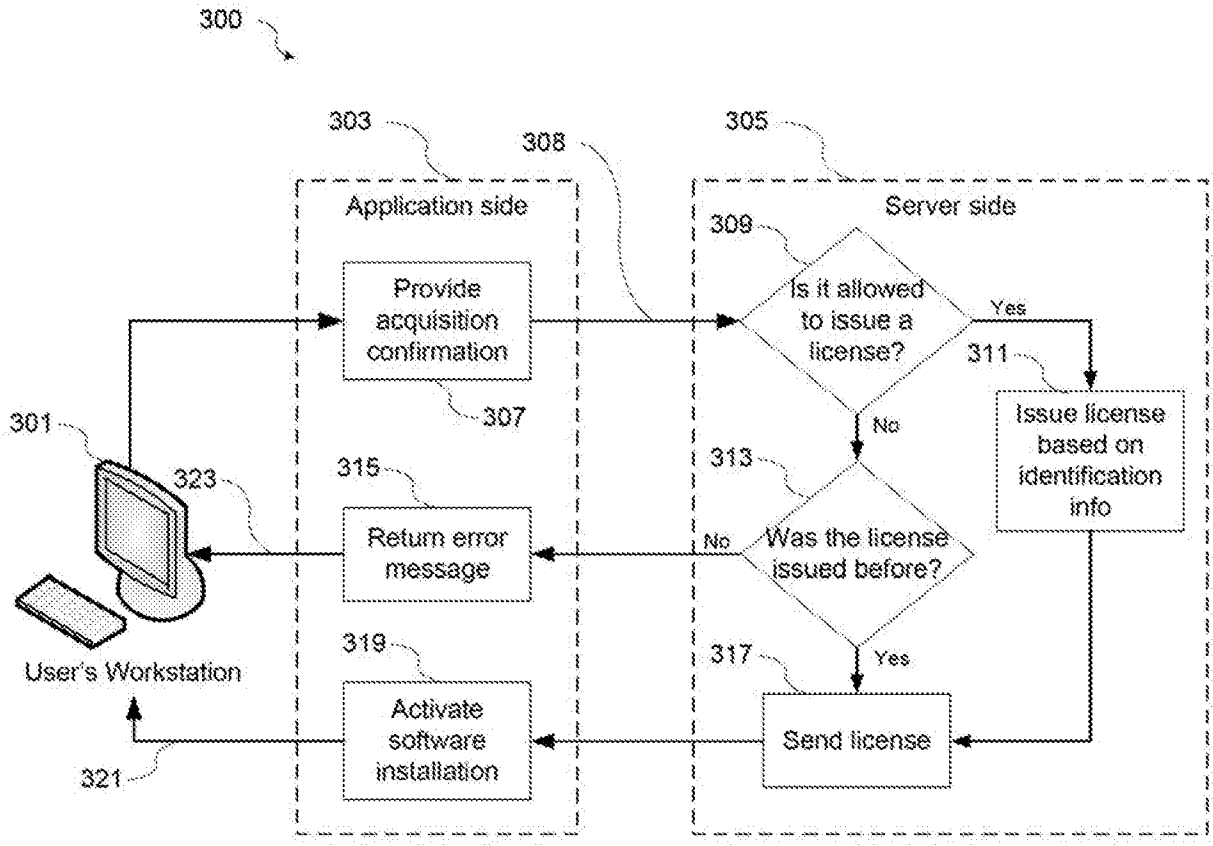
300
303
308
305
Application side
Server side
309
Provide acquisition confirmation
Is it allowed to issue a license?
Yes
311
Issue license based on identification info
307
No
313
301
323
315
Return error message
No
Was the license issued before?
User's Workstation
319
317
Activate software installation
Send license
Yes
321

FIG. 3

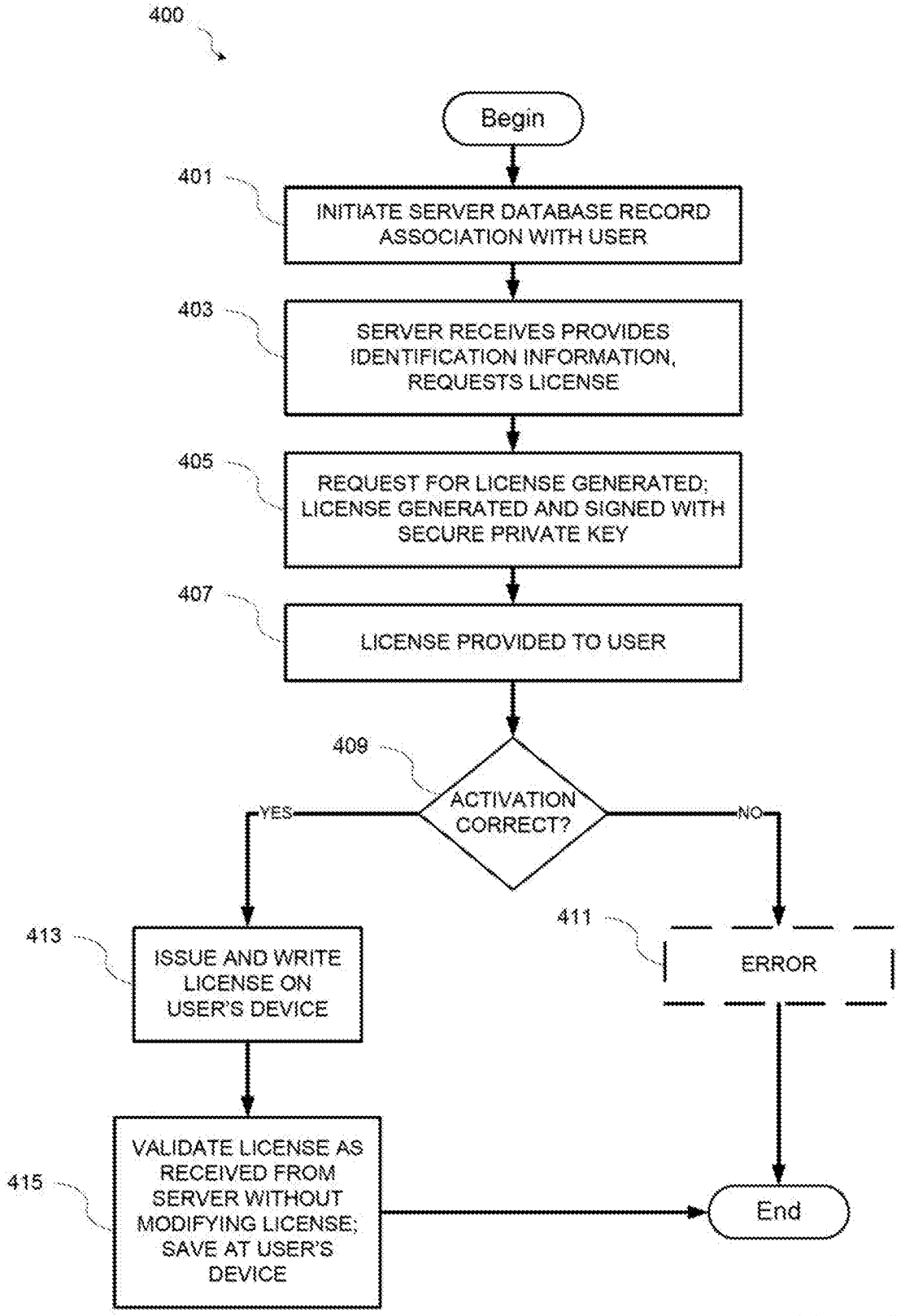
400
Begin
401
INITIATE SERVER DATABASE RECORD ASSOCIATION WITH USER
403
SERVER RECEIVES PROVIDES IDENTIFICATION INFORMATION, REQUESTS LICENSE
405
REQUEST FOR LICENSE GENERATED; LICENSE GENERATED AND SIGNED WITH SECURE PRIVATE KEY
407
LICENSE PROVIDED TO USER
409
ACTIVATION CORRECT?
YES
NO
411
ERROR
413
ISSUE AND WRITE LICENSE ON USER'S DEVICE
415
VALIDATE LICENSE AS RECEIVED FROM SERVER WITHOUT MODIFYING LICENSE; SAVE AT USER'S DEVICE
End

FIG. 4

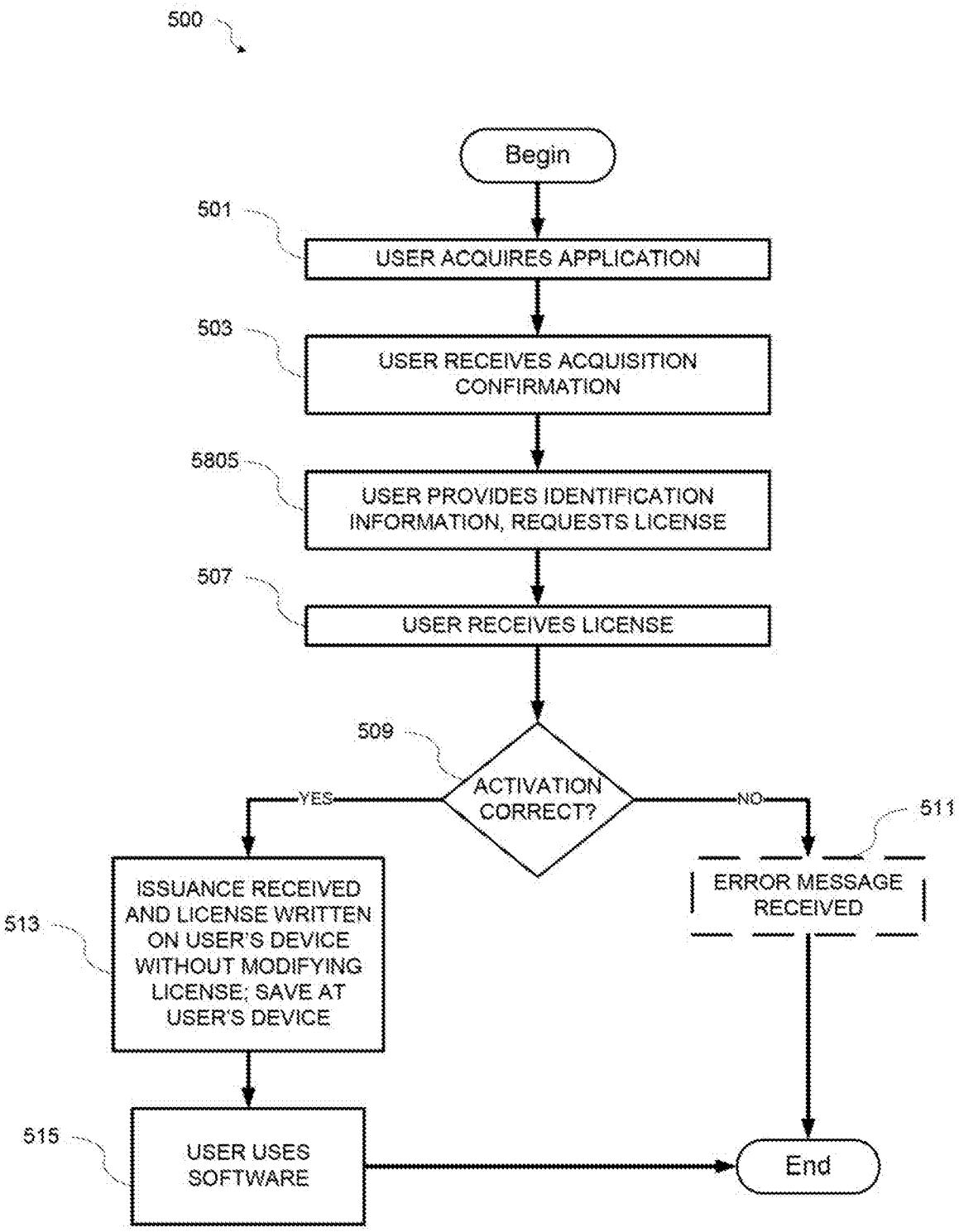
500
Begin
501
USER ACQUIRES APPLICATION
503
USER RECEIVES ACQUISITION CONFIRMATION
5805
USER PROVIDES IDENTIFICATION INFORMATION, REQUESTS LICENSE
507
USER RECEIVES LICENSE
509
ACTIVATION CORRECT?
YES
NO
511
ERROR MESSAGE RECEIVED
513
ISSUANCE RECEIVED AND LICENSE WRITTEN ON USER'S DEVICE WITHOUT MODIFYING LICENSE; SAVE AT USER'S DEVICE
515
USER USES SOFTWARE
End

FIG. 5

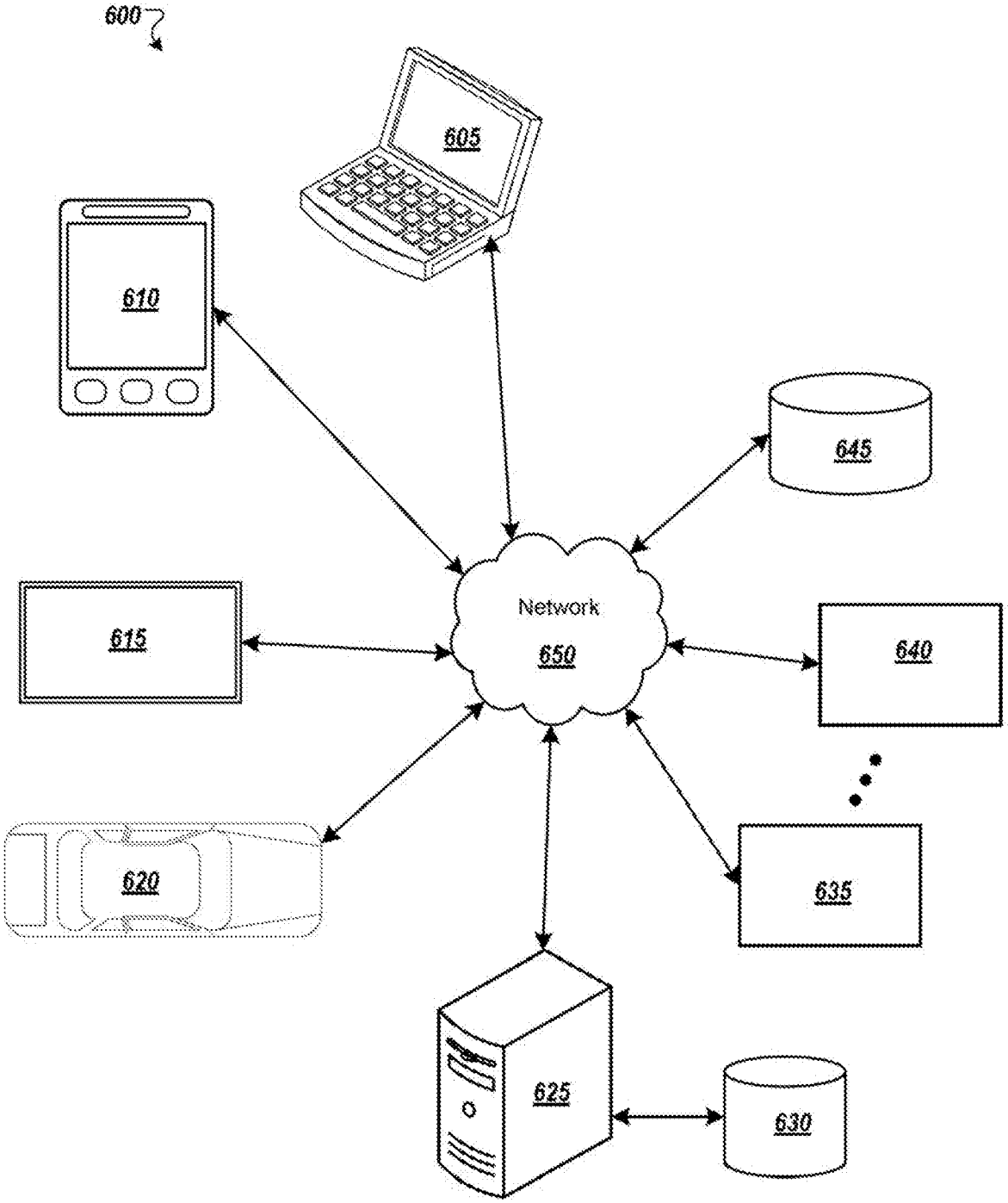

FIG. 6

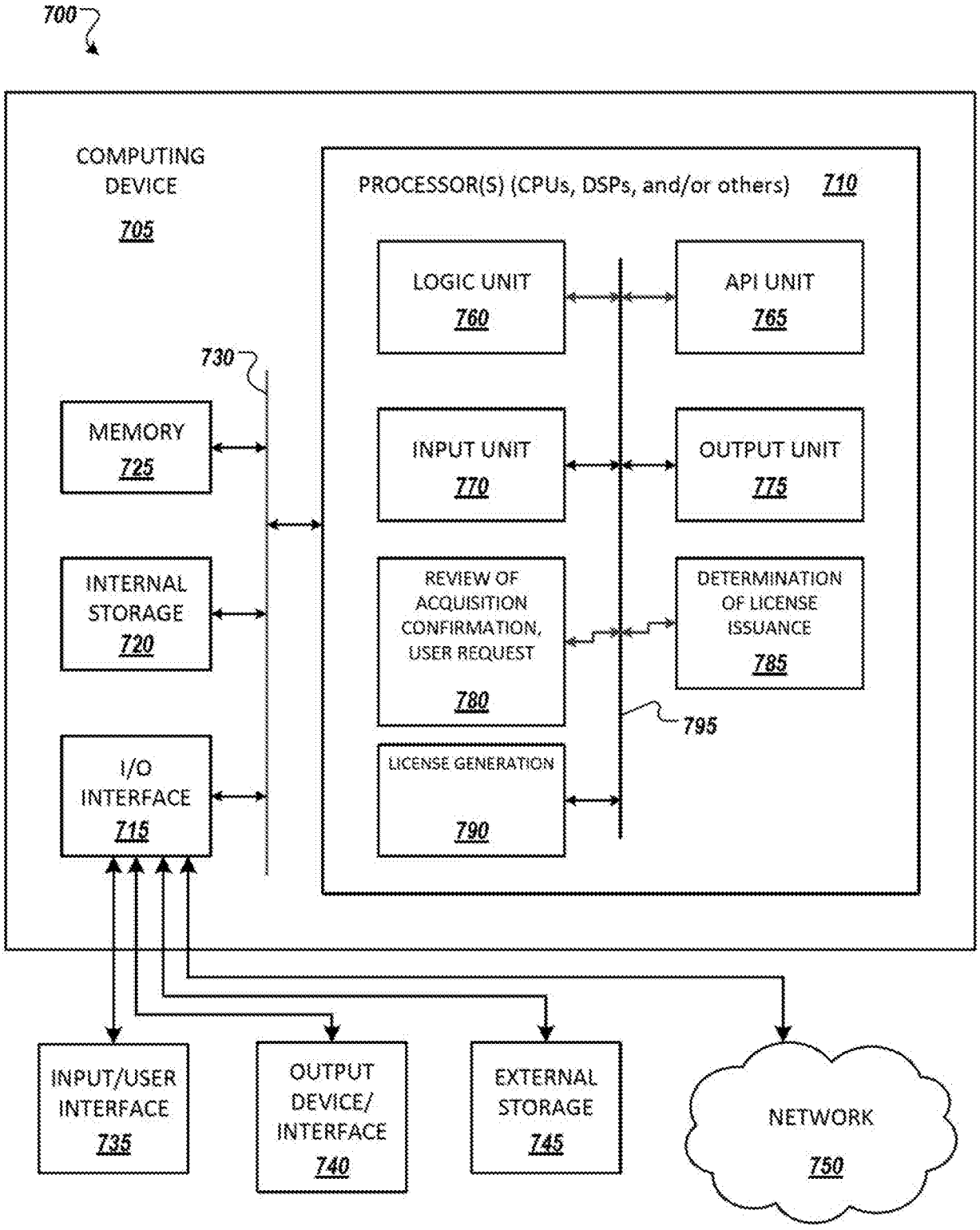
700
COMPUTING DEVICE
705
PROCESSOR(S) (CPUs, DSPs, and/or others)
710
LOGIC UNIT
760
API UNIT
765
730
MEMORY
725
INPUT UNIT
770
OUTPUT UNIT
775
INTERNAL STORAGE
720
REVIEW OF ACQUISITION CONFIRMATION, USER REQUEST
780
DETERMINATION OF LICENSE ISSUANCE
785
795
I/O INTERFACE
715
LICENSE GENERATION
790
INPUT/USER INTERFACE
735
OUTPUT DEVICE/ INTERFACE
740
EXTERNAL STORAGE
745
NETWORK
750

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2017/052567

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/10(2013.01)i, G06F 21/12(2013.01)i, H04L 9/32(2006.01)i, H04L 29/06(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 21/10; G06F 21/12; H04L 9/32; G06F 21/22; G06F 21/00; H04L 29/06

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords:software activation, licensing tracking, acquisition confirmation, private key

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2016-0232334 A1 (MACPAW INC.) 11 August 2016 See paragraphs [0078]-[0087]; and claims 1-10.	1-17
A	US 2012-0151574 A1 (NING ZHANG et al.) 14 June 2012 See paragraphs [0034]-[0042]; and figure 2.	1-17
A	US 2012-0131681 A1 (THOMAS J. LAYSON et al.) 24 May 2012 See paragraphs [0019]-[0025]; and figure 1.	1-17
A	US 2012-0131349 A1 (THOMAS J. LAYSON et al.) 24 May 2012 See paragraphs [0019]-[0025]; and figure 1.	1-17
A	US 2010-0293622 A1 (EGOR NIKITIN et al.) 18 November 2010 See paragraphs [0114]-[0119]; and figure 6.	1-17

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

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
07 February 2018 (07.02.2018)	**08 February 2018 (08.02.2018)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2017/052567

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2016-0232334 A1	11/08/2016	US 9659155 B2	23/05/2017
US 2012-0151574 A1	14/06/2012	CN 102737200 A	17/10/2012
		CN 102737200 B	29/07/2015
		TW 201224841 A	16/06/2012
		TW I550429 B	21/09/2016
		US 8683579 B2	25/03/2014
		WO 2012-082459 A1	21/06/2012
US 2012-0131681 A1	24/05/2012	CN 102419804 A	18/04/2012
		CN 102419804 B	22/04/2015
		US 8775797 B2	08/07/2014
US 2012-0131349 A1	24/05/2012	CA 2815375 A1	24/05/2012
		CN 102426640 A	25/04/2012
		CN 102426640 B	18/11/2015
		TW 201234208 A	16/08/2012
		TW I557589 B	11/11/2016
		US 8984293 B2	17/03/2015
		WO 2012-067888 A1	24/05/2012
US 2010-0293622 A1	18/11/2010	US 2016-0357949 A1	08/12/2016
		US 9424399 B2	23/08/2016