(19) DANMARK

(10) **DK/EP 2043055 T3**

Patent- og
Varemærkestyrelsen

(12) Oversættelse af
europæisk patentskrift

(51) Int.Cl.: ***G 07 C 9/00 (2020.01)***

(45) Oversættelsen bekendtgjort den: **2020-09-28**

(80) Dato for Den Europæiske Patentmyndigheds
bekendtgørelse om meddelelse af patentet: **2020-08-26**

(86) Europæisk ansøgning nr.: **07117498.1**

(86) Europæisk indleveringsdag: **2007-09-28**

(87) Den europæiske ansøgnings publiceringsdag: **2009-04-01**

(84) Designerede stater: **AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC MT NL PL PT RO SE SI SK TR**

(73) Patenthaver: **iLOQ Oy, Yrttipellontie 10, 90230 Oulu, Finland**

(72) Opfinder: **Lohiniva, Seppo, Kansankatu 57 C83, 90120, Oulu, Finland**
Pukari, Mika, Tohtorintie 5, 90570, OULU, Finland

(74) Fuldmægtig i Danmark: **Plougmann Vingtoft A/S, Strandvejen 70, 2900 Hellerup, Danmark**

(54) Benævnelse: **Låseadministrationssystem**

(56) Fremdragne publikationer:
EP-A- 1 024 239
EP-A- 1 132 871
EP-A- 1 249 797
EP-A- 1 549 020
EP-A- 1 653 415
WO-A-2006/136662
US-A- 5 602 536

DESCRIPTION

Field

[0001] The invention relates to lock administration systems for electromechanical locks. Especially, the invention relates to systems for self-powered locks.

Background

[0002] Various types of electromechanical locks are replacing the traditional mechanical locks. Electromechanical locks require an external supply of electric power, a battery inside the lock, a battery inside the key, or means for generating electric power within the lock making the lock self-powered. Electromechanical locks provide many benefits over traditional locks. They provide better security and the control of keys or security tokens is easier.

[0003] In addition, most electromechanical locks and/or keys and tokens are programmable. It is possible to program the lock to accept different keys and decline others.

[0004] One problem associated with electromechanical and self-powered locks is the programming of locks and keys. In many known electromechanical locking systems the lock manufacturer delivers factory programmed locks to the end user. The lock manufacturer has performed required programming of the locks belonging to a given locking system.

[0005] US 5602536 discloses a synchronization method for an entry system, EP 1549020 discloses an entry control system and EP 1653415 discloses management of access control badges.

Brief description

[0006] According to an aspect of the present invention, there is provided a lock administration system for self-powered locks as claimed in claim 1.

[0007] According to another aspect of the present invention, there is provided a method for administrating a system for self-powered locks as claimed in claim 11.

[0008] The invention has several advantages. The proposed solution enables flexible lock and key programming. The lock manufacturer or distributor maintains an ASP server which maintains a database of locking systems. However, the lock and key programming is performed by the end user. Thus, the lock manufacturer may deliver locks in an initial state in which the locks do not belong to any particular locking system. The initial state locks do not

store any security sensitive information.

[0009] In the proposed solution, locks need not have a dedicated wired connection to the ASP server. Encrypted lock programming data may be transmitted to the lock via public networks, which may be wired or wireless connections.

List of drawings

[0010] Embodiments of the present invention are described below, by way of example only, with reference to the accompanying drawings, in which

Figure 1 illustrates an example of the structure of a lock administration system;

Figure 2 illustrates a key and a lock;

Figure 3A is a flowchart illustrating an embodiment where a locking-system-shared-secret is generated;

Figure 3B is a flowchart illustrating an embodiment where an additional system token is created into the locking system;

Figure 3C is a flowchart illustrating an embodiment where the locking-system-shared-secret is transferred into a lock;

Figure 3D is a flowchart illustrating an embodiment where a key shared secret is set to a new key;

Figure 3E is a flowchart illustrating an embodiment where a lock is about to be opened using a key;

Figure 4 is a signalling chart illustrating an embodiment of the invention; and

Figure 5 illustrates another example of a key and a lock.

Description of embodiments

[0011] The following embodiments are exemplary. Although the specification may refer to "an", "one", or "some" embodiment(s) in several places, this does not necessarily mean that each such reference is made to the same embodiment(s), or that the feature only applies to a single embodiment. Features of different embodiments may also be combined to provide other embodiments.

[0012] With reference to Figure 1, an example of the structure of a lock administration system

is explained. The system comprises an application service provider (ASP) server 100 operationally connected to the Internet 104 and configured to store lock-system-related information to a database 102. The database 102 may be realised with detachable or fixed mass storage in the server or it may be a separate computer. Other realisations are also feasible. Typically, a lock system manufacturer or a lock system distributor maintains the ASP server 100. The database maintains data on locks and keys belonging to the locking system. The data comprises information on lock and key identities, key holders, lock and key status and access rights, for example.

[0013] The system further comprises a client module 110. The client module may be client software run in a client terminal 108 at a clients premises. Typically, the client terminal 108 is a personal computer or a corresponding processing unit connected to the Internet 104 through a wired or wireless connection 106.

[0014] The implementation of the client module 110 may vary, depending on the client terminal design. The client module may consist program instructions coded by a programming language, which may be a high-level programming language, such as C, Java, etc., or a low-level programming language, such as a machine language, or an assembler.

[0015] The client module 110 may be configured to manage locking-system-related information. For example, the client module may generate shared secrets for encrypting and decrypting, and generate and encrypt lock access data packets using a security token.

[0016] The client module may be connected 112 to a first device 114 configured to be in connection with a key 118 and a system token 120. The connection 112 between the client module and the first device may be realised with a wired or a wireless connection. The connection may be realised with USB, Bluetooth, Infrared or other known wireless techniques.

[0017] The first device 114 comprises an electronic circuit 116 and holders for a key 118 and a token 120. The electronic circuit 116 may comprise a processor and a memory for storing data and software for the processor. The electronic circuit may be configured to perform calculations relating to locking data and transfer information between the client module, key and the system token. The first device 114 and the client terminal 108 offer a platform for the client module 110 and a key 118 and a system token 120 communications. The client module 110 and the ASP server 100 communicate with the system token 120 for storing shared secrets of the lock system and for encrypting and decrypting lock access data packets and for authenticating a user access in the lock system.

[0018] The lock administration system may further comprise a second client module 126. The second client module 126 may be client software run in a client terminal 124. The client terminal 124 may be a personal computer, a personal data assistant (pda) or a mobile phone connected 122 to the Internet 104. The second client module 126 may be implemented in the same manner as the client module 110.

[0019] The second client module 126 may be connected 128 to a second device 130 configured to be in connection with a key 134 and a system token 136. The connection 128 between the second client module and the second device may be realised with a wired or a wireless connection. The connection may be realised with USB, Bluetooth, Infrared or other known wireless techniques. In addition, the second device may have a connection 138 to a lock 140. The connection may be wired or wireless. For example, a wired connection may be realised with a 1-wire bus connection. A wired connection may provide electric power to the self-powered lock. A wireless connection may be realised with known wireless protocols.

[0020] The second device 130 and the client terminal 124 offer a platform for the client module 126, the key 134, the system token 136 and the lock 140 communications for storing shared secrets of the locking system and for encrypting and decrypting lock access data packets and for authenticating a user access in the lock system.

[0021] In an embodiment, the first device and the second device are identical devices.

[0022] In an embodiment, the user of the client module 110 or 126 establishes a session between the client module and the ASP server 100 by logging in to the ASP server 100. The client module may contact the ASP server and check if there is an updated version of the module available. If so, the updated version may be downloaded and installed on the client terminal. After the required locking system administration operations have been initiated or performed the session may be ended by logging out of the ASP server.

[0023] Figure 2 illustrates a key 118 and a lock 140. The lock 140 is configured to read access data from the key 118 and match the data against a predetermined criterion. The key 118 comprises an electronic circuit configured to store access data and perform calculations relating to encrypting and decrypting. The electronic circuit may be an iButton® (www.ibutton.com) of Maxim Integrated Products, for example; such an electronic circuit may be read with 1-Wire® protocol. The electronic circuit may be placed in a key or a token, for example, but it may be positioned also in another suitable device or object. The only requirement is that the lock may read the data from the electronic circuit. The data transfer from the key to the lock 140 may be performed with any suitable wired or wireless communication technique. In self-powered locks, produced energy amount may limit the techniques used. Magnetic stripe technology or smart card technology may also be used in the key. Wireless technologies may include RFID (Radio-frequency identification) technology, or mobile phone technology, for example. The key may comprise a transponder, an RF tag, or any other suitable memory type capable of storing data.

[0024] The data read from the key is used for authentication by matching the data against the predetermined criterion. The authentication may be performed with SHA-1 (Secure Hash Algorithm) function, designed by the National Security Agency (NSA). In SHA-1, a condensed digital representation (known as a message digest) is computed from a given input data sequence (known as the message). The message digest is to a high degree of probability unique for the message. SHA-1 is called "secure" because, for a given algorithm, it is

computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest. If security needs to be increased, other hash functions (SHA-224, SHA-256, SHA-384 and SHA-512) in the SHA family, each with longer digests, collectively known as SHA-2 may be used. Naturally, any suitable authentication technique may be used to authenticate the data read from the external source. The selection of the authentication technique depends on the desired security level of the lock 140 and possibly also on the permitted consumption of electricity for the authentication (especially in user-powered electromechanical locks).

[0025] Figure 3A is a flowchart illustrating an embodiment where a locking-system-shared-secret (SS) is generated and a first system token is created into the locking system. The locking system shared secret is utilised in encrypting and decrypting lock access data. A system token comprises an electronic circuit described above and it is used in the first device 114 for generating and storing the locking system shared secret. The system token is a special token as it is not used as a key but for programming keys and locks of the locking system. Typically, creating a system token is the first step in programming locks and keys for a new locking system. A locking system may have more than one system tokens but they all store the identical locking-system-shared-secret.

[0026] The client module 110 is responsible for controlling the generation of the locking system shared secret and the system token. As the client module resides in a client terminal the procedure may be performed at the client's premises provided that the client module has Internet access and the device 114 is connected to the client terminal 108. In an embodiment, the client module 110 controls the device 114 to perform some or all of the tasks which in the following are allocated to the client module. The lock manufacturer or distributor has no part in the process other than maintaining the ASP server 100.

[0027] The process starts in step 300 when the user sets an empty token 120 into the first device 114.

[0028] In step 302, the client module 110 requests the user to type in seed 1. Seed 1 can be typically an alphanumeric string having 10-20 characters. Seed 1 is not stored in the system. The user must remember it.

[0029] In step 304, the client module 110 generates seed 2 using a random number generator. Seed 2 is typically 10 to 20-byte long list of numbers. Each byte can have any value between 0 and 255.

[0030] In step 306, the client module 110 generates seed 3 using a random generator. Seed 3 is typically 10 to 20 bytes long. Each byte can have any value between 0 and 255.

[0031] In step 308, the client module 110 sends seeds 1-3 to the token 120. The token receives the seeds and generates an SHA-1 hash to be used as the locking system shared

secret. The token 120 stores the shared secret into its hidden write only memory. The shared secret is not transmitted back to the client module or revealed to the user.

[0032] The hash may be generated using some other cryptographic hash function, as one skilled in the art is well aware. SHA-1 is used in this document merely as an example.

[0033] In an embodiment, the client module 110 is configured to calculate the hash which is used as the shared secret and to send the hash to the token 120 which stores the hash.

[0034] In step 310, the client module 110 stores seed 3 in the token 120.

[0035] In step 312, the client module 110 transmits seed 2 to the locking system database 102 maintained by the ASP server. This transmission may be encrypted with SSL (Secure Sockets Layer), for example.

[0036] In step 314, the client module 110 registers the token 120 as a system token in the locking system database 102. Each token may have a unique serial number which may be stored in the database 102. This storing may be encrypted with SSL (Secure Sockets Layer), for example.

[0037] The process ends in 316.

[0038] Figure 3B is a flowchart illustrating an embodiment where an additional system token is created into the locking system. The locking system already has at least one system token which was created using the procedure described in Figure 3A. The client module 110 is responsible for controlling the generation of the additional system token. As the client module resides in a client terminal the procedure may be performed at the client's premises provided that the client module has Internet access and the device 114 is connected to the client terminal 108. In an embodiment, the client module 110 controls the device 114 to perform some or all of the tasks which in the following are allocated to the client module. The lock manufacturer or distributor has no part in the process other than maintaining the ASP server 100.

[0039] The process starts in step 320 when the user has one of the existing system tokens 120 installed in the device 114.

[0040] In step 322, the client module 110 requests the user to type in seed 1. Seed 1 must be exactly the same as the one typed when generating the first system token 120.

[0041] In step 324, the client module 110 contacts the lock system database 102 via the Internet and reads seed 2 from the database 102.

[0042] In step 326, the client module 110 reads seed 3 from the existing system token 120 installed in the device 114.

[0043] In step 328, the client module 110 uses seeds 1 to 3 and generates an SHA-1 hash.

[0044] In step 330, the client module 110 validates the hash using the existing system token 120.

[0045] In step 332, the validation result is analysed. If the validation fails, the user has probably typed an incorrect seed 1 and the process is cancelled or restarted from step 322.

[0046] Otherwise, the process continues in step 334, where the client module requests the user to remove the existing system token 120 from the device 114 and set an empty token 121 into the device 114.

[0047] In step 336, the client module 110 stores seed 3 in the new token 121.

[0048] In step 338, the client module 110 sends seeds 1 and 2 to the token 120. The token receives the seeds and generates an SHA-1 hash using seeds 1 to 3. The generated hash is the locking system shared secret, the same that is stored in the first system token 120. The token stores the hash as the shared secret in its hidden write-only memory.

[0049] In step 340, the client module 110 registers the new system token 121 into the lock system database 102. This transmission may be encrypted with SSL (Secure Sockets Layer), for example.

[0050] The process ends in 342.

[0051] Figure 3C is a flowchart illustrating an embodiment where the locking system shared secret is transferred into a lock.

[0052] The process starts in step 350 when a user has one of the existing system tokens 120 installed in the device 114. Again, the client module 110 is responsible for the initial steps. As the client module 110 resides in a client terminal 108 the procedure may be performed at the client's premises provided that the client module 110 has Internet access and the device 114 is connected to the client terminal 108. The initial steps 350 to 366 may be performed at a site other than the one where the lock is situated. The lock manufacturer or distributor has no part in the process other than maintaining the ASP server 100. In an embodiment, the client module 110 controls the device 114 to perform some or all of the tasks which in the following are allocated to the client module.

[0053] In step 352, the client module 110 requests the user to type in seed 1. Seed 1 must be exactly the same as the one typed when generating the first system token 120.

[0054] In step 354, the client module 110 contacts the lock system database 102 via the Internet and reads seed 2 from the database 102.

[0055] In step 356, the client module 110 reads seed 3 from the system token 120 installed in the device 114.

[0056] In step 358, the client module 110 uses seeds 1 to 3 and generates an SHA-1 hash. The hash corresponds to the shared secret of the locking system.

[0057] In step 360, the client module 110 validates the hash against the shared secret stored in the system token 120 installed in the device 114.

[0058] In step 362, the validation result is analysed. If the validation fails, the user has probably typed an incorrect seed 1 and the process is cancelled or restarted from step 332.

[0059] Otherwise, the process continues in step 364 where seeds 1 to 3 are encrypted and stored in the system token as a programming job to a lock.

[0060] In step 366, the system token 120 is removed from the device 114 connected to the client module 110.

[0061] The remaining steps of the procedure are performed at the site where the lock is installed. A client terminal 124 comprises a second client module 126. The client terminal may be a personal computer, a pda, a smart phone or a corresponding apparatus. A second device 130 is connected to the client terminal and to the second client module and it has a connection to a lock 140.

[0062] In step 368, a system token 120 (which is illustrated as token 132 in Figure 1) is plugged into the device 130 which is connected to the lock 140.

[0063] In step 370, the lock 140 reads a programming job from the system token 120, decrypts seeds 1 to 3 and generates an SHA-1 hash.

[0064] In step 372, the lock 140 validates the hash against the shared secret stored in the system token 120 installed in the device 130.

[0065] In step 374, validation result is analysed.

[0066] If the validation fails, the lock 140 sets an error and does not set the locking system shared secret in step 378.

[0067] If the validation succeeds, the shared secret is stored in the lock 140 in step 378.

[0068] Process ends in step 376 or 378.

[0069] Steps 368 to 378 may be repeated on several locks. It is possible to transfer the locking system shared secret to several locks with the same initial steps.

[0070] Figure 3D is a flowchart illustrating an embodiment where a key shared secret is set to a new key. The client module 110 is responsible for controlling the generation of the shared secret. As the client module resides in a client terminal, the procedure may be performed at the client's premises provided that the client module has Internet access and the device 114 is connected to the client terminal 108. The lock manufacturer or distributor has no part in the process other than maintaining the ASP server 100. In an embodiment, the client module 110 controls the device 114 to perform some or all of the tasks which in the following are allocated to the client module.

[0071] The process starts in step 380 when a new key 118 and an existing system token 120 are connected in the device 114.

[0072] In step 382, the client module 110 reads key data from the key 118 and sends it to the system token 120. The key data may comprise a key serial number.

[0073] In step 384, the system token 120 computes key shared secret using key data and the locking system shared secret.

[0074] In step 386, the client module 110 sets the key shared secret to the new key 118.

[0075] In step 387, the client module 110 registers the new key 188 into the lock system database 102. This transmission may be encrypted with SSL (Secure Sockets Layer), for example.

[0076] The process ends in 388.

[0077] In addition to the above, additional access data may be programmed into a key of the locking system. In an embodiment, the key stores a data structure comprising key identification, the key shared secret and access group data. Each key has a unique identification ID which may be used to identify the key. The access group data comprises one or more access groups the key belongs to.

[0078] In an embodiment, a key may open a lock if it belongs to an access group to which access is allowed or if the key has a key identification ID to which access is allowed.

[0079] With the access groups, the organization of keys is greatly enhanced. A key may be provided with several access groups to allow access to different locations. For example, the same key may provide access to an apartment (access group 1), a cellar (access group 2), a garage (access group 3), and a waste bin shelter (access group 4). A user may then provide a waste management company with a key comprising only the access group 4. Thus, the company may be provided an access to the waste bin shelter but the key does not authorize access to other parts of the building.

[0080] Figure 3E is a flowchart illustrating an embodiment where a lock 140 is about to be opened using a key 118.

[0081] The process starts in step 390 when a user inserts the key 118 into the lock 140. At this phase, a self-powered lock may generate electric power from the key movement as the key is inserted into the lock. Alternatively, the lock may comprise a battery.

[0082] In step 391, the lock 140 reads key data and a hash from the key 118.

[0083] In step 392, the lock 140 computes an SHA-1 hash using the key data and the locking system shared secret stored in the lock.

[0084] In step 393, the lock 140 validates the hash computed by the lock against the hash read from the key 118.

[0085] In step 394, the validation result is analysed.

[0086] In step 399, if the validation fails, the lock 140 sets an error and does not open and the process ends.

[0087] If the validation succeeds, the lock 140 validates the key access data in step 396.

[0088] In step 397, the validation result is analysed. The key access data compromises information of possible access groups the key belongs to. The lock checks if there is a match between the access groups the key belongs to and the access groups the lock is programmed to open.

[0089] If the validation fails, the lock 140 sets an error and does not open. This is done in step 399.

[0090] If the validation succeeds, the lock 140 is opened in step 398.

[0091] The process ends in steps 398 or 399.

[0092] Figure 4 illustrates an example where an access right to a lock 140 is changed by the user using the client module 110. The client module 110 is responsible for controlling the initial part of the access right change. As the client module resides in a client terminal 108 the procedure may be performed at the client's premises provided that the client module has Internet access. Before the process starts, the system token 120 is placed in the device 114 and the device 114 is connected to the client terminal 108 and the client module 110. In addition, the client module logs in to the ASP server 100.

[0093] The ASP server maintains a database 102 where information on the locking system's locks, keys and access rights are stored. However, the access rights may not be changed at

the ASP server. The changing of the access rights requires the use of a client module 110, 126 and a system token connected to the client module via the device 114, 130.

[0094] In an embodiment, the client module provides the user of the system an interface to change the access rights and to program the locks and the keys. The client module 110 is configured to receive new lock access data from the user. As such data is received, the client module 110 sends a Program Lock message 402 to the database 102 maintained by the ASP server 100.

[0095] The ASP server 100 stores the received data into the database 102 and sends modified lock access data back to the Client Module 110 as a Send Job message 404. The client module 110 receives the message and sends the data as a Crypt Job message 406 to the system token 120 connected to the device 114. The system token 120 encrypts the access data with the locking system shared secret and sends the encrypted lock access data to the client module 110 as a Send Crypted Job message 408. The client module receives the encrypted data and sends it to the ASP server 100 as a Send Crypted Job message 410. The ASP server 100 places the data into a work queue 400 which is a part of the database 102. The work queue 400 is a list of encrypted access data messages which are to be transmitted to a lock later. The client module 110 may log out of the ASP server 100.

[0096] The remaining steps of the procedure are performed at the site where the lock is installed. First, the user logs in the ASP server 100 from the client module 126. At the user's command, the client module contacts the ASP server and selects a job for a lock to be programmed from the work queue 400 with a message 412. The work queue 400 replies by sending encrypted lock access data in a message 414. The client module 126 receives the job and stores it in the memory of the client terminal 124. The lock access data contained by the job data is encrypted and it is not a security risk to store the data in the client terminal 124.

[0097] Next, the system token 136 is placed to device 130. A connection between the device 130 and the client terminal 124 and the client module 126 is established. The client module is configured to send encrypted lock access data 416 to the system token 136 when receiving a Program Lock command from the user. The user connects the device 130 to the lock 140 to be programmed. When the lock 140 detects that a connection with the device 130 has been established the lock is configured to request 418 lock access data from the system token 136. In an embodiment, the lock is configured to authenticate the system token before requesting the data.

[0098] The system token 136 replies by sending the encrypted data 420. The lock 140 decrypts the data and validates its signature using the shared secret stored in the lock. If the data is valid the lock 140 stores the data and sends an encrypted acknowledgement message 422 comprising the lock programming status to the System Token 136 indicating that the access data of the lock has been programmed. If the data is not valid the lock 140 ignores the data and sends a negative acknowledgement 422 to the system token 136 indicating that the lock programming failed. In an embodiment, the device 130 is configured to inform the user

about the success of the lock programming with a visual indication, such as a green or a red led.

[0099] The system token 136 sends the encrypted lock programming status 424 to the client module 126. The client module 126 sends the encrypted lock programming status 426 to the work queue 400.

[0100] The lock programming status remains in the work queue 400 until the client module connected to the system token 120 establishes a session with the ASP server 100. The client module may be configured to check 428 the work queue 400 when connected to the ASP server 100. As a response to the query message 428 the ASP server 100 sends 430 the encrypted lock programming status to the client module 110.

[0101] When receiving the encrypted status message 430 the client module 110 sends 432 the message to the system token 120 which decrypts the data and replies by sending the decrypted data 434 to the client module 110. The client module sends the data 436 comprising the lock 140 status to the ASP server 100 which stores the lock status in the database 102.

[0102] The procedure described in connection with Figure 3C installs the locking system shared secret to a lock. Before the locking system shared secret is installed a lock may be in an initial state. An initial-state lock does not yet belong to any locking system. It is not configured to authenticate any keys and validate access data of the keys. The locking system shared secret may also be removed from a lock in a procedure similar to the procedure of Figure 3C. In an embodiment, the client module 110 is configured to generate lock access data packets comprising a command restoring a lock to an initial state. After the shared secret has been uninstalled the lock is back again in the initial state and it can be reused in another locking system without any security risk. A lock without a locking system shared secret does not have any stored security sensitive information.

[0103] When the locking system shared secret is installed into the lock using the procedure of Figure 3C the lock is a member of the locking system. Only the keys belonging to the locking system can open the lock. However, the lock does not validate any additional access data. This state of the lock may be called a commissioned state.

[0104] The locking system shared secret is generated on the basis of a seed given by the user with the system token 120 in the device 114 or the client module 110 as described in Figure 3A. The locking system shared secret is stored in the system token in a write-only memory.

[0105] Locks belonging to a system administrated by the described lock administration system have the ability to calculate the locking system shared secret as the system tokens. Keys have unique secrets generated from the unique identification of each key and the locking system shared secret. The locks are configured to generate the key secret on the basis of the unique identification read from a key and the locking system shared secret stored in the lock.

[0106] When lock access groups are installed into a lock using the procedure described in Figure 4, the lock is able to authenticate keys and validate key access data. This state of the lock may be described as an operating state. The key access data validation is explained further in European Patent Application 07112675.

[0107] Figure 5 illustrates an example of a key 118 and a lock 140. In the example of Figure 5, the key 118 comprises an electronic circuit 500 connected to a contact arrangement 502 and a key frame. The electronic circuit 500 may comprise a memory unit. The electromechanical lock 140 of Figure 1 is a self-powered lock. The lock 140 comprises power transmission mechanics 504 which transforms mechanic energy from a user to an electric generator 506 powering the electronic circuit 508 when the key 118 is inserted into the lock 140. In this example, the electronic circuit 508 is configured to communicate with the electronic circuit 500 of the key through a contact arrangement 510 and the contact arrangement 502 of the key. The communication may be realized as a wireless connection or by physical conductivity.

[0108] The electronic circuit 508 is configured to read key data from the electronic circuit 500 of the key 118 upon the key insertion. The electronic circuit 508 is further configured to authenticate the key and validate the access data as previously described. The electronic circuit may comprise a processor and a memory unit for storing data and required software for the processor. The software may be configured to perform the previously described procedures related to generating the locking system shared secret, updating the access data and authenticating the keys.

[0109] The lock of Figure 5 further comprises an actuator 512 configured to receive the open command, and to set the lock in a mechanically openable state. The actuator may be powered by the electric power produced with the generator 506. The actuator 512 may be set to the locked state mechanically, but a detailed discussion thereon is not necessary to illuminate the present embodiments.

[0110] When the actuator 512 has set the lock in a mechanically openable state a bolt mechanism 514 can be moved by rotating the key 118, for example. The mechanical power required may also be produced by the user by turning a handle or a knob of a door (not shown in Figure 5). Other suitable turning mechanisms may be used as well.

[0111] The steps and related functions described above are in no absolute chronological order, and some of the steps may be performed simultaneously or in an order differing from the given one. Other functions can also be executed between the steps or within the steps. Some of the steps or part of the steps can also be left out or replaced by a corresponding step or part of the step.

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US5602536A **[0005]**
- EP1549020A **[0005]**
- EP1653415A **[0005]**
- EP07112675A **[0106]**

Patentkrav

1. Låseadministrationssystem til selvdrevne låse, omfattende:

en applikationsserviceudbyder(ASP)-server (100), og mindst en lås (140), og en nøgle, mindst et klientmodul (110), en første indretning (114) og et systemtoken (121) der er hver konfigureret til at lagre mindst noget information relateret til låsesystemet;

hvor systemtokenet er konfigureret til at lagre en låsningssystem-delt hemmelighed til programmering af nøgler og låse,

hvor den første indretning (114) omfatter holdere (118, 120) til en nøgle og et systemtoken, og

hvor det mindst ene klientmodul (110) er konfigureret til

at forbinde og styre den første indretning (114), når et systemtoken og en ny nøgle er forbundet med den første indretning, til at programmere den nye nøgle under anvendelse af systemtokenet ved at indstille en nøgle-delt hemmelighed til kryptering og dekryptering til den nye nøgle,

at forbinde og styre den første indretning (114), når systemtokenet er forbundet med den første indretning, til at generere og kryptere låseadgangspakker under anvendelse af systemtokenet og at generere delte hemmeligheder til kryptering og dekryptering,

at transmittere datapakkerne til ASP-serveren (100) under anvendelse af offentlige netværk,

at modtage en krypteret statuspakke fra ASP-serveren (100) under anvendelse af offentlige netværk,

at styre dekrypteringen af statuspakken under anvendelse af systemtokenet og

at sende information vedrørende den dekrypterede statuspakke til ASP-serveren (100) under anvendelse af offentlige netværk;

hvor ASP-serveren (100) er konfigureret til

at være virksomt forbundet til internet (104),

at opretholde en database (102) til lagring af låse- og nøgleadgangsdata og

at lagre låseadgangspakker og krypterede statuspakker i databasen;

og mindst en lås (140) konfigureret til

at modtage datapakker fra ASP-serveren (100) via offentlige netværk,

at dekryptere datapakkerne under anvendelse af systemtokenet og at sende en krypteret statuspakke til ASP-serveren (100) under anvendelse af offentlige netværk.

2. Låseadministrationssystemet ifølge krav 1, hvor klientmodulet er konfigureret til at styre den første indretning til at generere låseadgangsdatapakker omfattende information om låsningssystemet til hvilket en lås hører til og om adgangsrettigheder af låsen.

3. Låseadministrationssystemet ifølge krav 1, hvor klientmodulet er konfigureret til at styre den første indretning til at generere låseadgangsdatapakker omfattende en kommando til at gendanne en lås til indledende tilstand.

4. Låseadministrationssystemet ifølge krav 1, hvor den første indretning er konfigureret til at være i forbindelse med en nøgle, et klientmodul og til at kommunikere med systemtokenet.

5. Låseadministrationssystemet ifølge krav 1, omfattende en anden indretning konfigureret til at være i forbindelse med en lås og til at kommunikere med systemtokenet.

6. Låseadministrationssystemet ifølge krav 5, omfattende et andet klientmodul konfigureret til at være i forbindelse med ASP-serveren under anvendelse af offentlige netværk og med den anden indretning igennem en trådet eller trådløs forbindelse.

7. Låseadministrationssystemet ifølge krav 6, hvor det andet klientmodul er konfigureret til at modtage en låseadgangsdatapakke fra ASP-serveren og at transmittere pakken til en lås via den anden indretning.

8. Låseadministrationssystemet ifølge krav 6, hvor det andet klientmodul er konfigureret til at modtage en krypteret statuspakke fra en lås via den anden indretning og at transmittere pakken til ASP-serveren.

9. Låseadministrationssystemet ifølge krav 6, hvor forbindelsen mellem det andet klientmodul og ASP-serveren er mindst delvist trådløs.

10. Låseadministrationssystemet ifølge krav 6, hvor systemet omfatter et andet klientmodul i en mobil terminal.

11. Fremgangsmåde til administrering af et system ifølge krav 1 til selvdrevne låse, omfattende:

lagring af mindst noget information relateret til låsesystemet i en applikationsserviceudbyder(ASP)-server (100), mindst en lås (140), mindst et klientmodul (110), en første indretning (114) og et systemtoken (121);

lagring af en låsningssystem-delt hemmelighed til programmering af nøgler og låse i systemtokenet;

styring af den første indretning (114) med et klientmodul (110) i programmeringen af nye nøgler under anvendelse af systemtokenet (121), når et systemtoken og en ny nøgle forbindes med den første indretning, ved at indstille en nøgle-delt hemmelighed til kryptering og dekryptering til den nye nøgle;

styring af den første indretning (114) med et klientmodul (110) i genereringen og krypteringen af låseadgangsdatapakker, når systemtokenet forbindes med den første indretning, under anvendelse af systemtokenet ved generering af delte hemmeligheder til kryptering og dekryptering;

kryptering af de genererede låseadgangsdatapakker under anvendelse af systemtokenet (121);

transmittering af de krypterede datapakker til applikationsserviceudbyder(ASP)-serveren (100) under anvendelse af offentlige netværk;

opretholdelse af en database i ASP-serveren (100);

lagring med ASP-serveren (100) af de krypterede datapakker i databasen;

læsning af de krypterede datapakker med en lås (140) fra serveren via offentlige netværk;

dekryptering af datapakkerne i låsen (140);

generering af krypteret statuspakke i låsen (140) og transmittering af pakken til ASP-serveren (100);

læsning af en statuspakke fra ASP-serveren (100) og styring af dekrypteringen af statuspakken med et klientmodul (110);

transmittering af information vedrørende den dekrypterede statuspakke fra klientmodulet (110) til ASP-serveren (100).

12. Fremgangsmåden ifølge krav 11, yderligere omfattende:
generering af i klientmodulet låseadgangsdatapakker omfattende information om låsningssystemet til hvilket en lås tilhører og om adgangsrettigheder af låsen.

13. Fremgangsmåden ifølge krav 11, yderligere omfattende:
generering af i klientmodulet låseadgangsdatapakker omfattende en låsekommando "gendan til indledende tilstand".

DRAWINGS

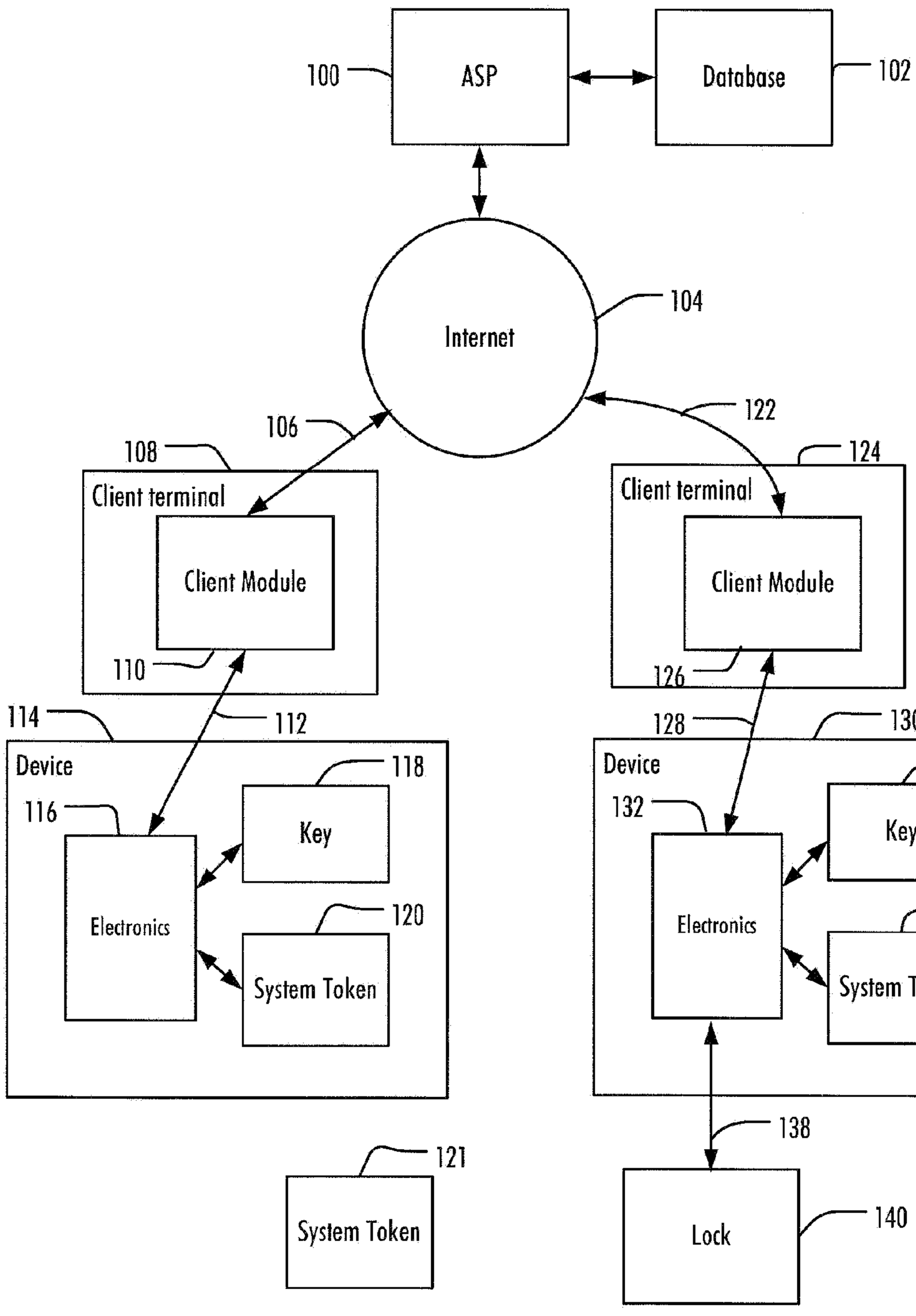


FIG. 1

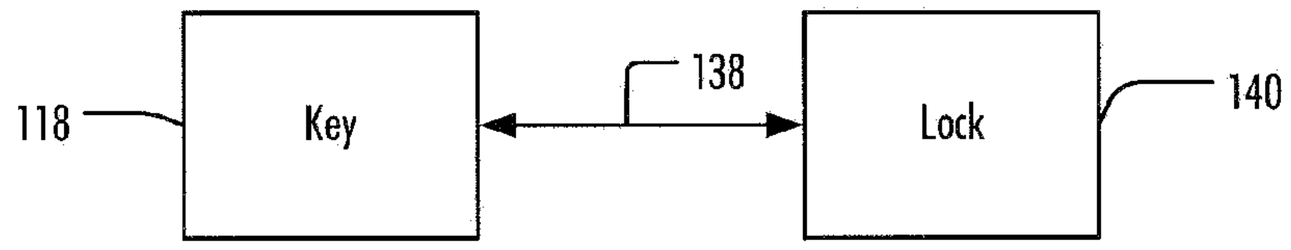

118
Key
138
Lock
140

FIG. 2

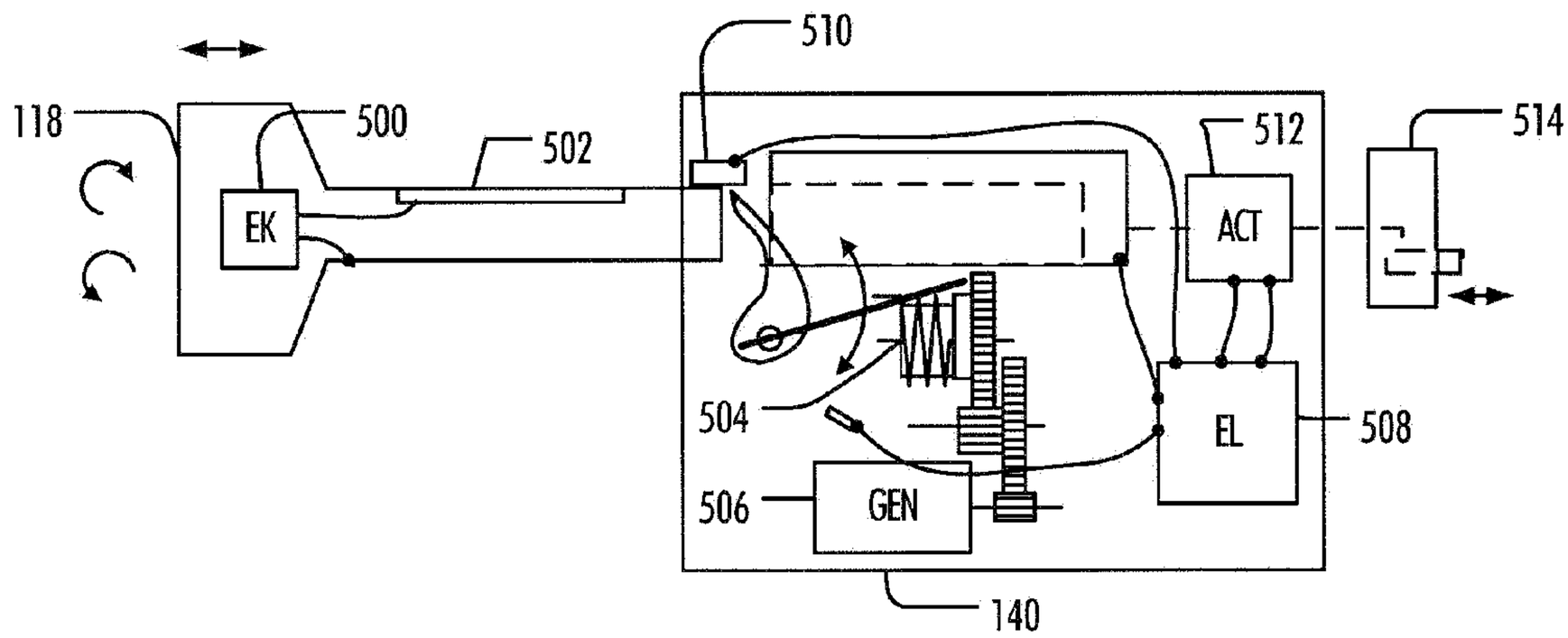

118
500
502
EK
510
512
514
ACT
504
EL
508
506
GEN
140

FIG. 5

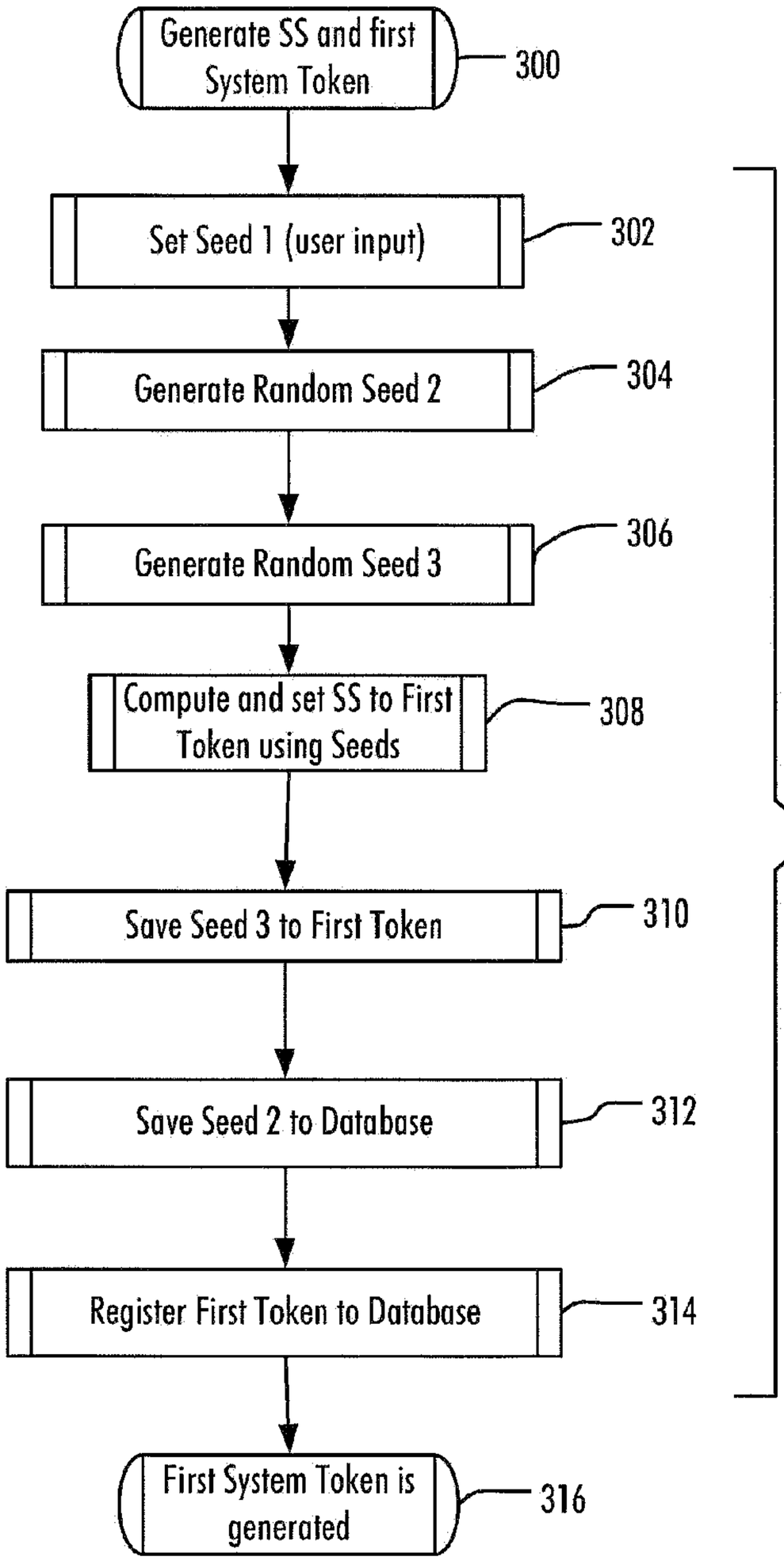

Generate SS and first System Token
300
Set Seed 1 (user input)
302
Generate Random Seed 2
304
Generate Random Seed 3
306
Compute and set SS to First Token using Seeds
308
Save Seed 3 to First Token
310
Save Seed 2 to Database
312
Register First Token to Database
314
First System Token is generated
316
Client Module

FIG. 3A

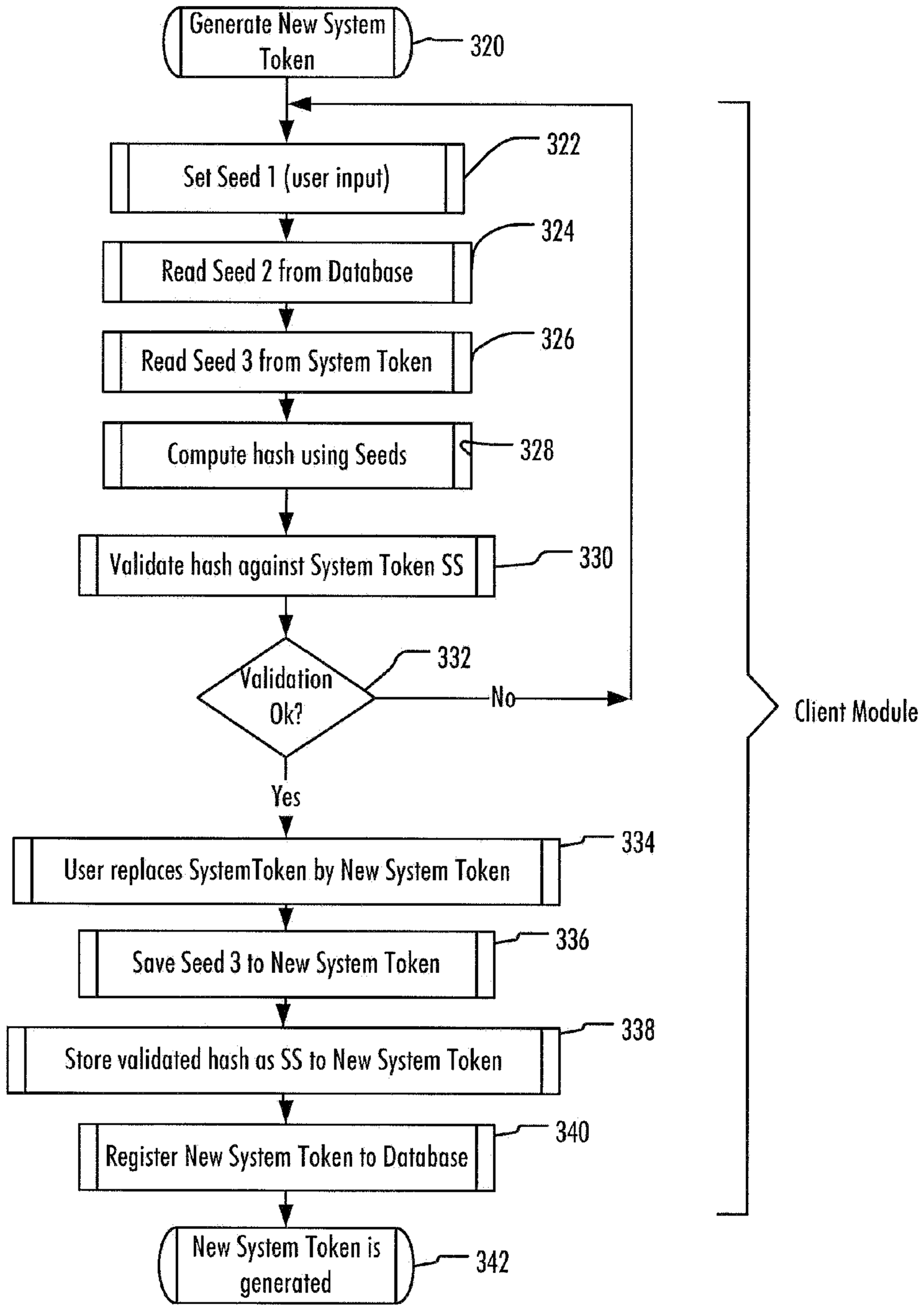

Generate New System Token
320
Set Seed 1 (user input)
322
Read Seed 2 from Database
324
Read Seed 3 from System Token
326
Compute hash using Seeds
328
Validate hash against System Token SS
330
Validation Ok?
332
No
Yes
User replaces SystemToken by New System Token
334
Save Seed 3 to New System Token
336
Store validated hash as SS to New System Token
338
Register New System Token to Database
340
New System Token is generated
342
Client Module

FIG. 3B

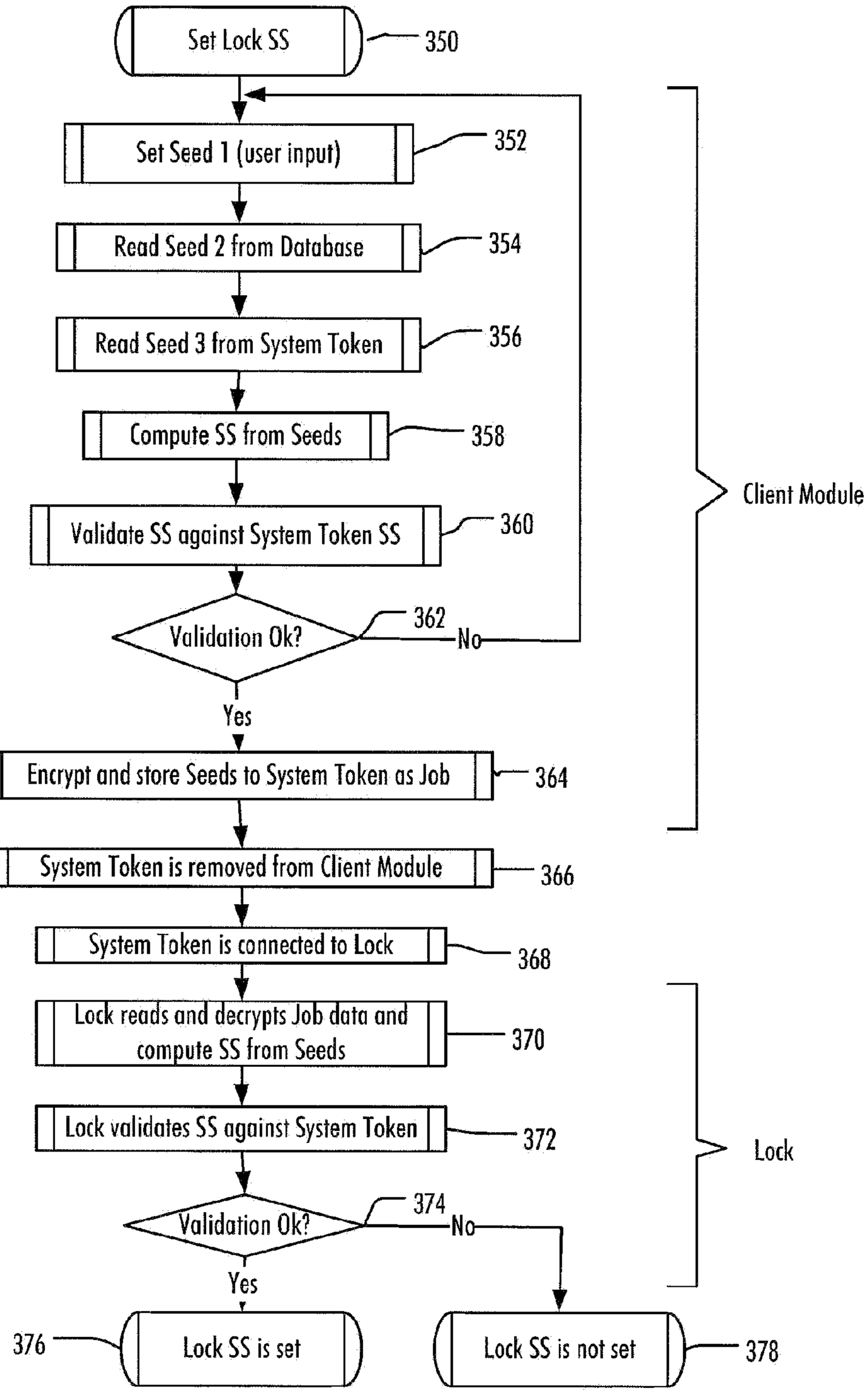

Set Lock SS
350
Set Seed 1 (user input)
352
Read Seed 2 from Database
354
Read Seed 3 from System Token
356
Compute SS from Seeds
358
Validate SS against System Token SS
360
Validation Ok?
362
No
Yes
Encrypt and store Seeds to System Token as Job
364
System Token is removed from Client Module
366
System Token is connected to Lock
368
Lock reads and decrypts Job data and compute SS from Seeds
370
Lock validates SS against System Token
372
Validation Ok?
374
No
Yes
376
Lock SS is set
Lock SS is not set
378
Client Module
Lock

FIG. 3C

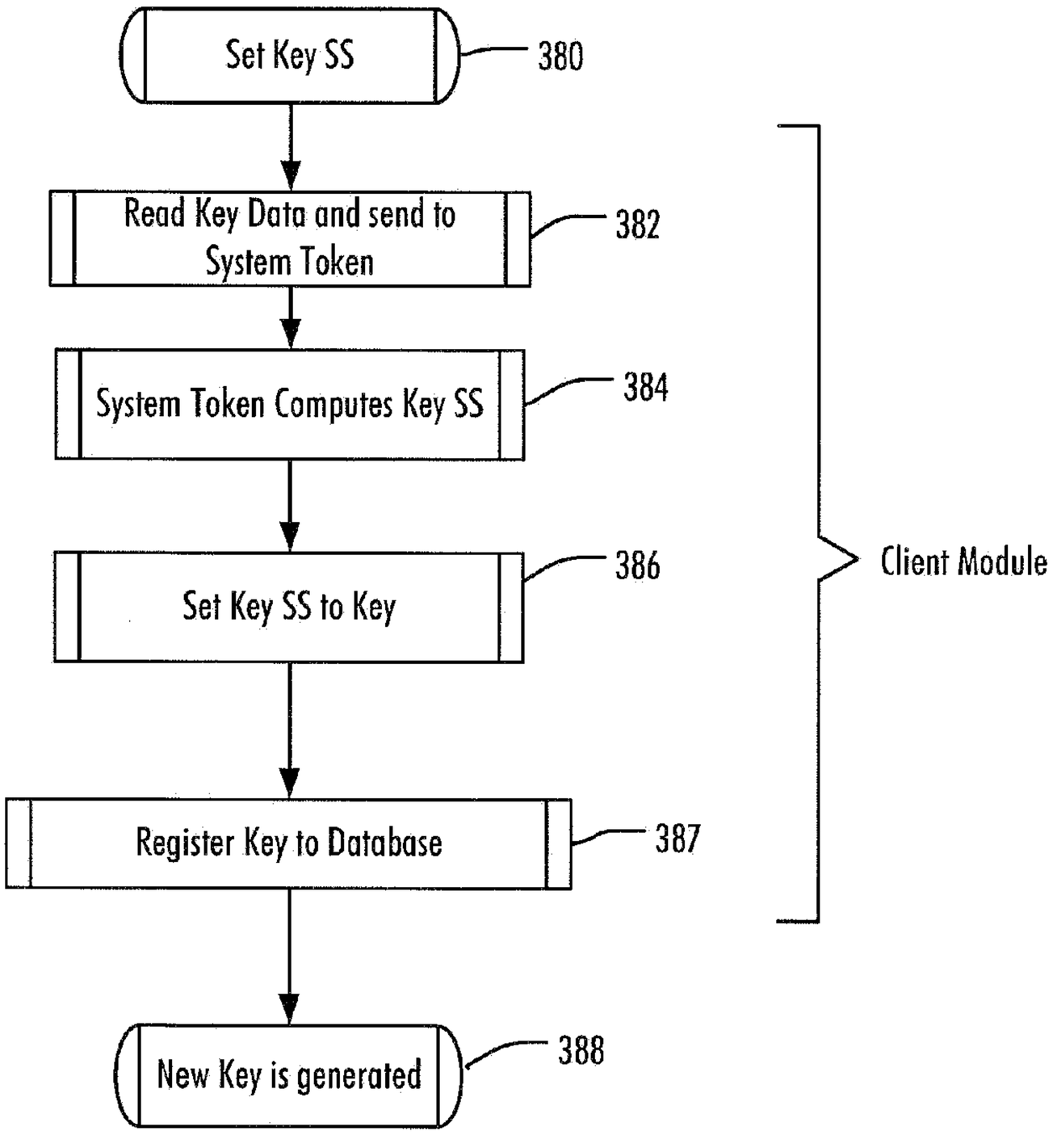

Set Key SS
380
Read Key Data and send to System Token
382
System Token Computes Key SS
384
Set Key SS to Key
386
Register Key to Database
387
New Key is generated
388
Client Module

FIG. 3D

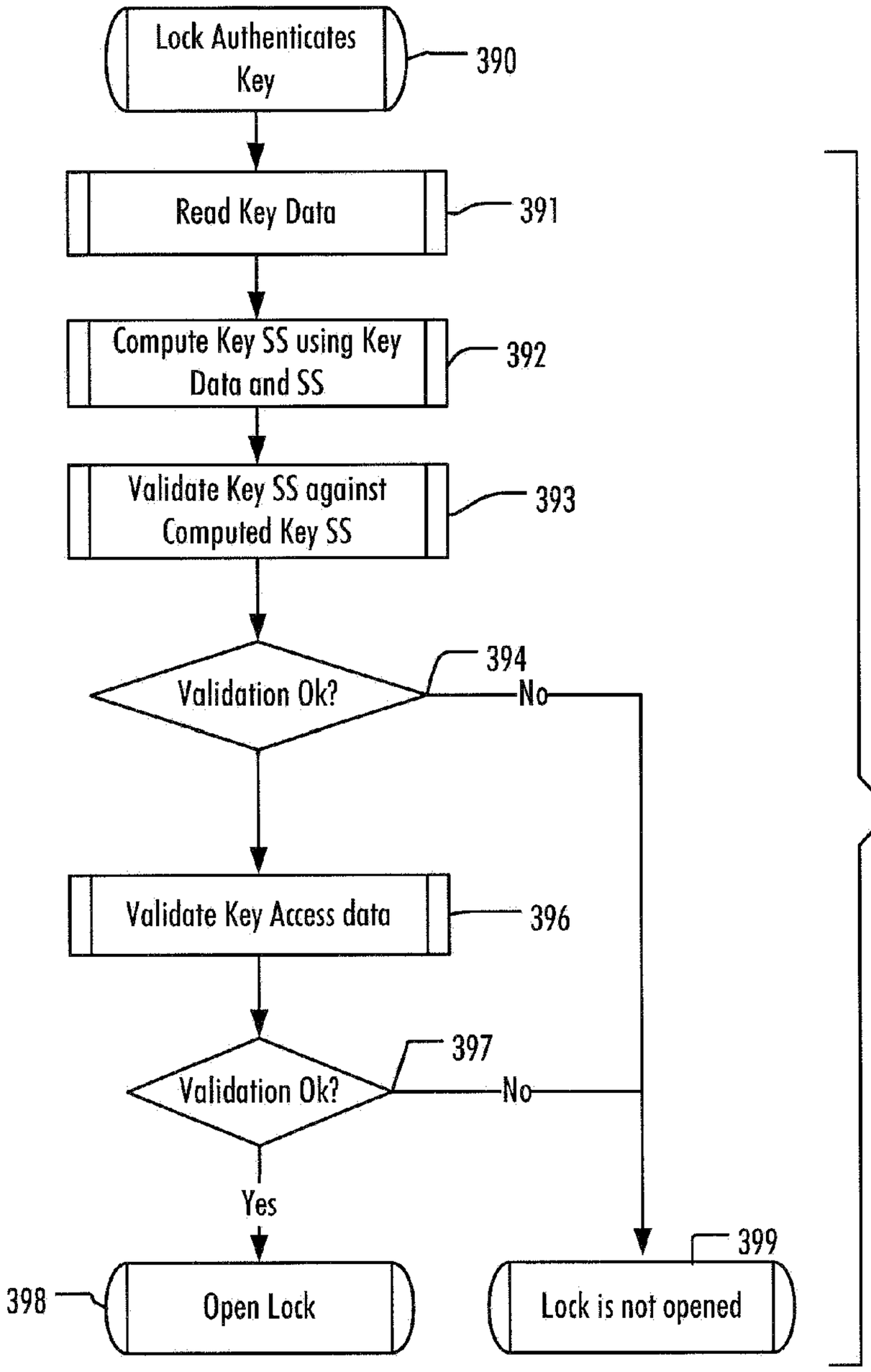

Lock Authenticates Key
390
Read Key Data
391
Compute Key SS using Key Data and SS
392
Validate Key SS against Computed Key SS
393
Validation Ok?
394
No
Validate Key Access data
396
Validation Ok?
397
No
Yes
398
Open Lock
399
Lock is not opened
Lock

FIG. 3E

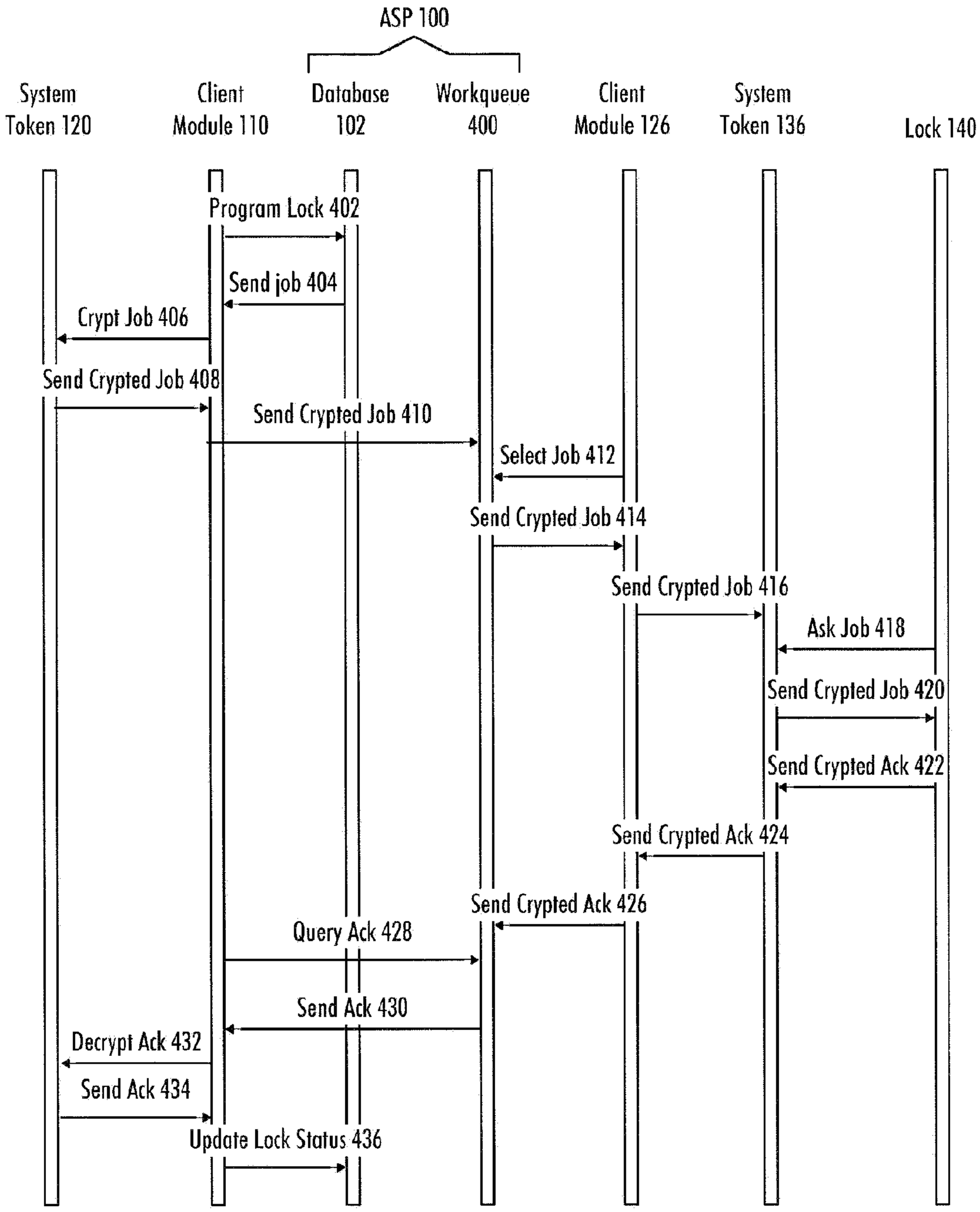

ASP 100
System Token 120
Client Module 110
Database 102
Workqueue 400
Client Module 126
System Token 136
Lock 140
Program Lock 402
Send job 404
Crypt Job 406
Send Crypted Job 408
Send Crypted Job 410
Select Job 412
Send Crypted Job 414
Send Crypted Job 416
Ask Job 418
Send Crypted Job 420
Send Crypted Ack 422
Send Crypted Ack 424
Send Crypted Ack 426
Query Ack 428
Send Ack 430
Decrypt Ack 432
Send Ack 434
Update Lock Status 436

FIG. 4