(19) SUOMI - FINLAND
(FI)

PATENTTI- JA REKISTERIHALLITUS
PATENT- OCH REGISTERSTYRELSEN
FINNISH PATENT AND REGISTRATION OFFICE

(10) **FI 132006 B1**

(12) **PATENTTIJULKAISU**
PATENTSKRIFT
PATENT SPECIFICATION

(45) Patentti myönnetty - Patent beviljats - Patent granted **09.04.2026**

(51) Kansainvälinen patenttiluokitus - Internationell patentklassificering - International patent classification
H04L 9/08 (2006 . 01)
H04L 9/16 (2006 . 01)
H04L 9/32 (2006 . 01)
H04L 9/40 (2022 . 01)
H04L 9/06 (2006 . 01)
G06K 19/06 (2006 . 01)
G06K 19/07 (2006 . 01)
H04W 4/80 (2018 . 01)

(21) Patenttihakemus - Patentansökan - Patent application 20255297

(22) Tekemispäivä - Ingivningsdag - Filing date **31.03.2025**

(23) Saapumispäivä - Ankomstdag - Reception date **31.03.2025**

(41) Tullut julkiseksi - Blivit offentlig - Available to the public **09.04.2026**

(73) Haltija - Innehavare - Holder
1• Gurulogic Microsystems Oy , Linnankatu 34 , 20100 Turku , (FI)

(72) Keksijä - Uppfinnare - Inventor
1• Kärkkäinen , Tuomas , Turku , (FI)

(74) Asiamies - Ombud - Agent
Moosedog Oy , Kurjenmäenkatu 10 B 49 , 20700 Turku

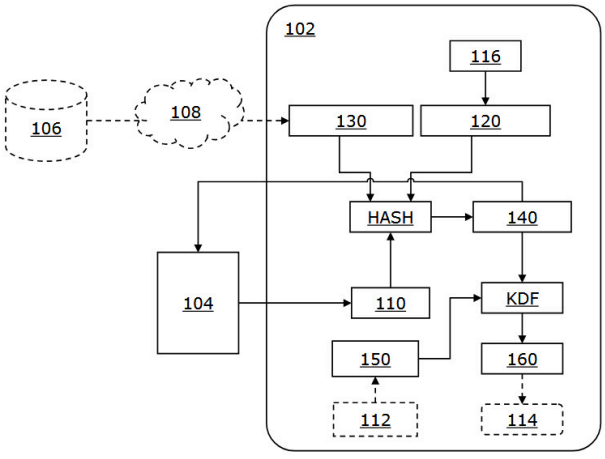
(54) Keksinnön nimitys - Uppfinningens benämning - Title of the invention
MENETELMÄ JA JÄRJESTELMÄ VÄHINTÄÄN YHDEN PÄÄSALAISUUDEN LUOMISEEN
Förfarande och system för att skapa åtminstone en huvudhemlighet
A METHOD AND SYSTEM FOR GENERATING AT LEAST ONE MASTER SECRET

(56) Viitejulkaisut - Anförda publikationer - References cited
US 2010058060 A1, WO 2021084220 A1, KRAWCZYK, H et al. HMAC-based Extract-and-Expand Key Derivation Function (HKDF). INTERNET ENGINEERING TASK FORCE. RFC 5869 [online]; May 2010; ISSN: 2070-1721 [retrieved on 2025-06-03], US 2023104633 A1

(57) Tiivistelmä - Sammandrag - Abstract

Keksinnön mukainen menetelmä vähintään yhden pääsalaisuuden luomiseksi, menetelmä käsittää seuraavat vaiheet: (i) ensimmäisen asiakaspäätelaitteen (102, 402, 506, 602) avulla haetaan ensimmäinen suola (110, 310, 610) suolalähteestä (104, 304, 502, 608); (ii) luodaan ensimmäinen nonce (120, 320, 612) ensimmäisellä ajanhetkellä (t1); (iii) sovelletaan hajautusfunktiota (HASH) ensimmäiseen suolaan, ensimmäiseen nonceen ja ensimmäiseen tunnistautumistunnisteeseen (130, 330, 606) toisen suolan (140, 340, 614) johdattamiseksi; (iv) saadaan ensimmäinen käyttäjän antama avain (150, 350, 618); (v) sovelletaan avaimen johdannaisfunktiota (KDF) toiseen suolaan ja ensimmäiseen käyttäjän antamaan avaimeen ensimmäisen pääsalaisuuden (160, 360A, 620) luomiseksi osana vähintään yhtä pääsalaisuutta.

Disclosed is a method for generating at least one master secret, the method comprising: (i) retrieving with a first client terminal (102, 402, 506, 602), from a salt source (104, 304, 502, 608), a first salt (110, 310, 610); (ii) generating a first nonce (120, 320, 612) at a first moment of time (t1); (iii) applying a hash function (HASH) to the first salt, the first nonce and a first identification token (130, 330, 606) to derive a second salt (140, 340, 614); (iv) obtaining a first user-provided key (150, 350, 618); and (v) applying a key derivation function (KDF) to the second salt and the first user-provided key to generate a first master secret (160, 360A, 620) amongst the at least one master secret.



A METHOD AND SYSTEM FOR GENERATING AT LEAST ONE MASTER SECRET

TECHNICAL FIELD

The present disclosure relates to methods for generating at least one master secret. Moreover, the present disclosure also relates to first client terminals configured to implement said methods. Furthermore, the present disclosure also relates to systems for generating at least one master secret.

BACKGROUND

Presently, security of digital authentication and encryption systems often depends on strength of user-provided secrets (for example, such as PIN codes, passwords, passphrases, or similar). These user-provided secrets are often considered as weak secrets, since they are typically prone to security vulnerabilities due to their predictability, short length, and susceptibility to brute-force attacks. In other words, the weak secrets are easily breakable. A compromised weak secret can lead to unauthorized access to systems, data breaches, and identity theft.

Traditionally, various solutions have been implemented to enhance the security of weak secrets. However, these solutions have several limitations associated therewith. As an example, one conventional approach requires users themselves to create complex secrets that meet stringent criteria, including one or more of a minimum length, a mix of uppercase letters, lowercase letters, numbers, and special characters, or similar. However, this approach is prone to usability issues since users struggle to remember the complex secrets, which ultimately results in poor secret management practices.

As another example, some conventional systems incorporate solutions such as password generators, password managers or hardware-hardened key values for handling complex secrets. Such solutions rely on a master password, which the user must remember, for enabling access to the complex secrets. A drawback of such solutions is that they involve a single point of failure, which is the master password. For example, if the master password is compromised, all stored complex secrets become vulnerable.

As a yet another example, some conventional solutions involve multi-factor authentication (MFA), which combine user-provided secrets with other forms of authentication for additional security. For example, MFA requires users to verify their identity using the user-provided secret and at least one other authentication factor, such as biometric recognition, one-time passwords (OTPs), hardware tokens, or similar. While MFA enhances security, it often introduces inconvenience and reliance on additional hardware or communication channels for provision of the at least one other authentication factor.

A commonly-used approach in cryptographic security nowadays is the use of random values (known as salts), in combination with hash functions to prevent attacks using precomputed tables (e.g., rainbow table attacks). However, the storage and management of these salts is typically performed by third-party devices and thus imposes additional security challenges since improper handling of the salts can expose systems to replay attacks or unauthorized access.

Therefore, high-security systems cannot fully rely on user-provided secrets and nowadays use secure hardware for enabling extra protection. Examples of such secure hardware may be Hardware security modules (HSMs), Secure Elements (SEs), Trusted Platform Modules (TPMs), Trusted Execution Environments (TEEs), or similar. Such secure hardware provides a high level of security but is cost-inefficient (due to

certification requirements which require significant resources). As a result, such secure hardware is deployed to a very limited extent.

Document US 2010058060 A1 discloses a method and apparatus for a system and process for sharing a secret over an unsecured channel in conjunction with an authentication system. A client computes a message authentication code based on a hashed password value and a first random string received from the server. The client sends a response to the server that includes authentication data including a second random string. Both the client and server concatenate the first random string, second random string and username. These values are processed to generate as a shared master secret to further generate shared secrets or keys to establish a secured communication channel between the client and server. The secured communication can be based on stateless messaging where the decryption key associated with the message is identified by the message authentication code, which is placed within the message.

Document WO 2021084220 A1 discloses system(s), method(s), apparatus and device(s) that are provided in relation to iterative key generation for a constrained device with at least a first execution stage and a second execution stage. Each execution stage is associated with a corresponding set of computer code or instructions stored on the device and that is configured to execute on the device during execution of said execution stage. Each execution stage sequentially executes on the device. During execution of the first execution stage, a second stage key is generated for use by the second execution stage based on a first stage key accessible by the first execution stage. The generated second stage key is stored in accessible storage on the device for access and use by the second execution stage. The first stage key is accessible for use by the first execution stage during execution of the first execution stage on the device and inaccessible thereafter.

Document KRAWCZYK, H et al. “HMAC-based Extract-and-Expand Key Derivation Function (HKDF).” INTERNET ENGINEERING TASK FORCE. RFC 5869; May 2010, discloses a simple Hashed Message Authentication Code (HMAC)-based key derivation function (HKDF), which can be used as a building block in various protocols and applications. The key derivation function (KDF) is intended to support a wide range of applications and requirements, and is conservative in its use of cryptographic hash functions.

Document US 2023104633 A1 discloses systems and methods to generate unique high entropy deterministic authentication tokens or passwords derived from user account information and a high entropy random number secured on a user device. Systems and methods can further include a counting variable such as a nonce to generate more than one unique high entropy deterministic authentication token for a single account, for instance, to facilitate updating the authentication token for that account to a new unique authentication token. From the user perspective, systems and methods can be used to provide an authentication token manager and/or password manager, where the user is able to access user accounts via a single user-provided authentication such as a master password or biometric data. Because the tokens/passwords are generated deterministically, they need not be stored by the token/password manager.

Therefore, in light of the foregoing discussion, there exists a need to overcome the aforementioned drawbacks.

SUMMARY

The aim of the present disclosure is to provide a method, a first client terminal, and a system for generating at least one master secret to enhance security of memory-based user-provided secrets without requiring a hardware-hardened security environment. The aim of the

present disclosure is achieved by a method, a first client terminal, and a system for generating at least one master secret as defined in the appended independent claims to which reference is made. Advantageous features are set out in the appended dependent claims. The embodiments of the present disclosure substantially enable the improvement of cryptographic security by deriving high-entropy master secrets from user-provided keys using cryptographic operations performed in secure memory that are resistant to brute-force attacks and replay attacks.

Throughout the description and claims of this specification, the words "*comprise*", "*include*", "*have*", and "*contain*" and variations of these words, for example "*comprising*" and "*comprises*", mean "*including but not limited to*", and do not exclude other components, items, integers or steps not explicitly disclosed also to be present. Moreover, the singular encompasses the plural unless the context otherwise requires. In particular, where the indefinite article is used, the specification is to be understood as contemplating plurality as well as singularity, unless the context requires otherwise.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating how a method for generating at least one master secret is implemented, in accordance with an embodiment of the present disclosure;

FIG. 2 is a flowchart illustrating steps of the method for generating the at least one master secret, in accordance with an embodiment of the present disclosure;

FIG. 3 is a block diagram-based timeline illustrating how the method is used to generate a plurality of master secrets, in accordance with an embodiment of the present disclosure;

FIG. 4 is a block diagram of a first client terminal for generating the at least one master secret, in accordance with an embodiment of the present disclosure;

FIG. 5 is a schematic illustration of a system for generating the at least one master secret, in accordance with an embodiment of the present disclosure; and

FIG. 6 is a sequence diagram illustrating how at least one master secret is generated, in accordance with an embodiment of the present disclosure.

DETAILED DESCRIPTION OF EMBODIMENTS

The following detailed description illustrates embodiments of the present disclosure and ways in which they can be implemented. Although some modes of carrying out the present disclosure have been disclosed, those skilled in the art would recognize that other embodiments for carrying out or practising the present disclosure are also possible.

In a first aspect, the present disclosure provides a method for generating at least one master secret, the method comprising:

(i) retrieving, with a first client terminal, from a salt source, a first salt, wherein the first salt is generated by a trust provider using a true random number generator (TRNG);

(ii) generating a first nonce at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);

(iii) applying a hash function to the first salt, the first nonce and a first identification token to derive a second salt, wherein the first identification token is generated by an identification token provider and provided to the first client terminal for authenticating a user of the first client terminal,

FI20255297 PRH 18-12-2025

wherein the identification token is a short-lived or session-bound authentication credential;

(iv) obtaining a first user-provided key; and

(v) applying a key derivation function to the second salt and the first user-provided key to generate a first master secret amongst the at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

The combination of the aforementioned steps (i), (ii), (iii), (iv), and (v) results in a cryptographic method that significantly enhances security of at least one weak secret by transforming the at least one weak secret into the at least one master secret which is stronger than the at least one weak secret. The use of the first salt and the first nonce prevents replay attacks and ensures uniqueness in key derivation. The hash function ensures that the first salt (and optionally, its subsequent salt(s)) evolve dynamically, reducing the risk of exposure even if some inputs to the hash function are compromised. The first user-provided key is securely transformed using the key derivation function, thereby preventing direct brute-force attacks. Together, these steps create a strong and secure first master secret on demand in a memory-resident execution format (i.e., cryptographic operations are performed in memory) without requiring expensive secure hardware, making the method suitable for implementation in a wide range of consumer and enterprise applications. In other words, the method ensures high security of the first master secret by leveraging true randomness, secure storage, and user-memorised key-based authentication, making the first master secret suitable for various cryptographic use cases.

It will be appreciated that the method enables computation of the at least one master secret without necessarily requiring the first client terminal (which is a personal device) of the user. For example, in case of a fault, breakage or loss of the first client terminal, the user can still generate

FI20255297 PRH 18 -12- 2025

the at least one master secret using another client terminal and the salt source. In this way, the method enables secure access to various digital services, even without using personal devices.

In a second aspect, the present disclosure provides a first client terminal comprising:

a user interface;

an access interface; and

a processor coupled to the user interface and the access interface, wherein the processor is configured to:

(I) retrieve a first salt from a salt source, via the access interface, wherein the first salt is generated by a trust provider using a true random number generator (TRNG);

(II) generate a first nonce at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);

(III) apply a hash function to the first salt, the first nonce and a first identification token to derive a second salt, wherein the first identification token is generated by an identification token provider and provided to the first client terminal for authenticating a user of the first client terminal, wherein the identification token is a short-lived or session-bound authentication credential;

(IV) obtain a first user-provided key via the user interface; and

(V) apply a key derivation function to the second salt and the first user-provided key to generate a first master secret amongst at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

FI20255297 PRH 18 -12- 2025

The present disclosure provides the aforementioned first client terminal. The user interface, the access interface, and the processor of the first client terminal collectively enable the first client terminal to implement the method of the first aspect. In particular, the user interface enables obtaining the first user-provided key from a user of the first client terminal in a user-friendly and convenient manner; the access interface enables the first client terminal to automatically retrieve data (such as the first salt) from the salt source and optionally also write data to a storage medium (which is external to the first client terminal); and the processor executes processing steps for generating the first master secret in a secure manner that is resistant to cyber-attacks and does not require expensive secure hardware. The first client terminal is simple, easy to implement, and cost-efficient.

In a third aspect, the present disclosure provides a system for generating at least one master secret, the system comprising:

a salt source;

an identification token provider; and

a first client terminal communicably coupled to the salt source and the identification token provider, wherein the first client terminal is configured to:

(A) retrieve a first salt from the salt source, wherein the first salt is generated by a trust provider using a true random number generator (TRNG);

(B) generate a first nonce at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);

(C) apply a hash function to the first salt, the first nonce and a first identification token to derive a second salt, wherein the first identification

token is generated by an identification token provider and provided to the first client terminal for authenticating a user of the first client terminal, wherein the identification token is a short-lived or session-bound authentication credential;

(D) obtain a first user-provided key; and

(E) apply a key derivation function to the second salt and the first user-provided key to generate a first master secret amongst the at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

The present disclosure provides the aforementioned system. The salt source enables secure and easy retrieval of at least one salt (comprising the first salt), and the identification token provider generates at least one cryptographic identification token (comprising the first identification token) that authenticates a user of the first client terminal or the first client terminal. The first client terminal uses the at least one salt and the at least one cryptographic identification token, when executing processing operations for transforming the first user-provided key (which is a weak secret) into the first master secret (which is a relatively stronger secret). The system efficiently strengthens weak secrets into stronger master secrets in a manner that is computationally efficient and resistant to cyber-attacks, without requiring expensive secure hardware.

The first client terminal and the salt source are communicably coupled via a communication network, thereby enabling the first client terminal to retrieve the first salt from the salt source. Optionally, the salt source comprises at least one of: a trust provider configured to generate one or more salts, a storage medium configured to store thereat one or more salts. This means that pursuant to embodiments of the present disclosure, the first salt (and optionally, subsequent salts such as a second salt, a third salt, and the like) could be received from the trust provider and/or the storage medium. Throughout the present disclosure,

FI20255297 PRH 18 -12- 2025

the term "*client terminal*" refers to any computing device that is capable of executing cryptographic operations. Examples of the first client terminal include, but are not limited to, a smartphone, a tablet computer, and a desktop computer. The communication network may be wired, wireless, or a combination thereof. For example, the communication network may be Internet. The "*first salt*" is a cryptographic random value obtained from a trusted source, which serves as an initial input for deriving the first master secret. In this regard, the first salt may also be referred to as an "*initial salt*", and its generation and storage is discussed later in more detail. Optionally, the first salt is received by the first client terminal, from the trust provider. In this regard, the first salt may not be stored at the storage medium, but may directly be sent from the trust provider to the first client terminal. The method allows generation of the first master secret at various client terminals, as the first salt can easily be retrieved on-demand from the salt source.

Optionally, the salt source comprises a storage medium, and wherein the storage medium is a near-field communication (NFC) chip or a quick response (QR) code. The term "*storage medium*" refers to an element where data can be stored. The storage medium is typically under the user's direct control and in the user's possession. The storage medium could be any portable storage element. The NFC chip is an electronic component capable of storing and transmitting data wirelessly over predefined distances. The NFC chip provides a secure, portable, and tamper-resistant storage solution. The QR code is a machine-readable optical label (i.e., an optically-readable device) capable of storing various types of data. In this case, the QR code encodes a given cryptographic salt (such as the first salt, the second salt, and the like), allowing the first client terminal to retrieve said salt via optical scanning (for example, using a camera or a QR code reader). Optionally, the QR code is used when the NFC chip is unavailable. Optionally, the QR code is securely backed up in a trusted storage. This trusted storage may be a data

repository of a trust provider. By using the NFC chip or the QR code as the storage medium, there is enabled secure, cost-effective storage of at least one salt without relying on cloud-based or server-side data repositories. This improves security by reducing a risk of remote breaches. The storage medium enables secure usage of the at least one salt even on non-personal (or borrowed) client terminals. The selection of the NFC chip or the QR code as the storage medium provides a balance between security, cost-efficiency, and user convenience, ensuring that the method can be deployed across various client terminals and usage scenarios.

Notably, the first client terminal may read the first salt using an access interface of the first client terminal. The access interface ensures that only authorized client terminals with proper permissions can access the first salt. When the storage medium is implemented as the NFC chip, an interfacing module of the access interface may comprise an NFC reader. When the storage medium is implemented as the QR code, the interfacing module of the access interface may comprise a camera or an optical scanner.

Next, the first nonce is generated by the first client terminal. The term "*nonce*" refers to a unique, one-time-use value that ensures cryptographic freshness in cryptographic operations and prevents replay attacks. The first nonce is generated at the first moment of time, meaning that the first nonce is specific to a particular operation for generating the first master secret. The first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, pseudo-random number generator (PRNG). This ensures that nonces are unique across different instances of the method for generating different master secrets. The high-resolution timestamp may be derived from an internal clock (i.e., a timer) of the first client terminal, and enables fast and distinct nonce generation since each nonce is derived at a precise moment of time. For example, the high-resolution timestamp may be a

microsecond-level timestamp, a millisecond-level timestamp, or similar, which minimizes risk of nonce duplication. The counter-based system ensures incremental uniqueness of the nonces across the different instances by generating non-repeating sequential counter values. Optionally, the high-resolution timestamp is used for generating the first nonce when the counter-based system is not available. The PRNG provides unpredictability of the nonces across the different instances, making future nonce prediction difficult. The use of the first nonce in the method prevents reuse of previous cryptographic computations when generating the first master secret, thereby improving security. Generation of unique one or more nonces enables the method to effectively provide secrecy, because it ensures that deterministically behaving cryptographic operations may not reveal used secrets (such as salts) if results of such operations are compromised.

Next, the second salt is derived using the hash function. The "*hash function*" is a mathematical transformation that maps one or more inputs to a fixed-size output in a way that is infeasible to reverse. The hash function is applied to three inputs: the first salt, the first nonce, and the first identification token, to derive the second salt as the fixed-size output. This step can be expressed as the following equation:

second salt = hash (first salt, first identification token, first nonce)

The first identification token is a cryptographic credential issued by a secure entity (such as an identification token provider). The first identification token may, for example, be a Firebase® token, a Google® ID token, an Apple® ID token, a Microsoft Entra™ ID token, an anonymous identification token, an anonymous identification token that does not require additional user action, an anonymous Firebase® authentication token that does not require user login, or similar. Prior to the step of deriving the second salt, the method comprises obtaining the first identification token from the identification token provider. Employing

the hash function ensures that the second salt inherits entropy of the three inputs of the hash function while introducing additional randomness. Therefore, even if one input (for example, such as the first identification token) amongst the three inputs is compromised, the (derived) second salt remains secure. The second salt has high entropy, and is a crucial confidential element in further strengthening the security of the first master secret.

Next, the first user-provided key is obtained from a user (via a user interface of the first client terminal, for example) of the first client terminal. Throughout the present disclosure, the term "*user-provided key*" refers to a secret that the user remembers and provides for the generation of the at least one master secret. The user-provided key is user-memorised secret. The first user-provided key may, for example, be a PIN, password, or a passphrase. The first user-provided key is obtained for enabling generation of the first master secret. By integrating the first user-provided key into a cryptographic framework with multiple layers of security (including the hash-based second salt derivation step), the method ensures that even simple user-provided key(s) can be securely transformed into strong master secret(s).

Next, the first master secret is generated by employing the key derivation function. The key derivation function (KDF) is a cryptographic algorithm that takes as input at least a weak secret (such as the first user-provided key) and transforms the weak secret into a high-entropy master secret (such as the first master secret) that is suitable for secure cryptographic applications. The second salt is also included as another input to the key derivation function in the described method, to prevent precomputed attacks (such as rainbow table attacks) by ensuring that each instance of key derivation is unique. This step can be expressed as the following equation:

first master secret = KDF (first user-provided key, second salt)

The first master secret produced by this transformation can beneficially be used for encryption, authentication, or other security mechanisms. Throughout the present disclosure, the term "*master secret*" refers to a strong cryptographic key that is dynamically derived from a corresponding user-provided key (which is a relatively weaker secret than the master secret) using a cryptographic technique. The use of the key derivation function makes it computationally expensive (and resultantly, impractical) and extremely difficult for an attacker to derive the first master secret through brute-force attempts. Furthermore, in the method, the first master secret is generated on demand (i.e., on the fly) without being securely stored. This means that the first master secret is a runtime cryptographic key, which can be deterministically computed into a memory of the first client terminal, as required. The memory is a secure memory region of the first client terminal. The memory may be a secure runtime memory (for example, such as a secure Random Access Memory, RAM, such as a guarded heap). This secure memory region prevents unauthorized access to the first master secret, even if the first client terminal is compromised. It will be appreciated that the step of generating the first master secret is a one-time step, but may be repeated if the confidentiality of the second salt is compromised.

Optionally, the method further comprises storing the second salt to a storage medium. This step ensures that a latest (i.e., most recently-derived) cryptographic salt remains available at the storage medium for facilitating subsequent key derivation operation(s) to generate subsequent master secret(s) amongst the at least one master secret. Salts are typically rotated or changed periodically to enhance security. The second salt, stored in the storage medium, prevents loss of cryptographic integrity which would occur if the second salt were not stored and had to be recomputed for the subsequent key derivation operation(s). Storing the second salt is implemented by writing the second salt to the storage medium after its computation. The second salt

may be written to the storage medium, and a subsequent salt (if any) may be subsequently written to the storage medium in addition to the second salt that is inactive at a time of generating the subsequent salt, or by overwriting the second salt. It will be appreciated that only one active salt for a current master secret is stored at a time, in the storage medium. This ensures that attackers do not have opportunities to exploit vulnerabilities and compromise master secrets. In an embodiment where the storage medium is the NFC chip, the second salt may be written to the storage medium using the interfacing module (of the access interface) embedded in the first client terminal. In another embodiment where the storage medium is the QR code, the second salt is encoded into the QR code and visually printed for scanning when required. So, storing the second salt in the QR code may involve printing a new QR code. Storing the second salt to the storage medium is a cost-efficient solution. Additionally, or alternatively, the second salt may be stored to a database of the trusted source (which generated the first salt). This ensures availability of the second salt even if the salt source is lost or damaged. The first master secret may beneficially be restored using this backup mechanism - by obtaining the second salt from the database of the trusted source, and using the second salt for re-generating the first master secret.

Optionally, the method further comprises:

(vi) retrieving, with the first client terminal, from the salt source, the second salt;

(vii) generating a second nonce at a second moment of time;

(viii) applying the hash function to the second salt, the second nonce and one of: the first identification token or a second identification token to derive a third salt;

(ix) obtaining a second user-provided key; and

FI20255297 PRH 18 -12- 2025

(x) applying the key derivation function to the third salt and the second user-provided key to generate a second master secret amongst the at least one master secret.

In this regard, the method can beneficially regenerate multiple strong cryptographic master secrets dynamically over time while preventing replay attacks and maintaining security integrity. At each instance of generation of a new master secret, use of a new nonce ensures that a corresponding salt evolves independently at said instance, and such a salt subsequently enables secure master secret generation for said instance. In this way, the method ensures that at each such instance of generation, any user-provided key does not become a security risk. Future salts cannot be derived based on past values. Together, the steps (vi), (vii), (viii), (ix), and (x) along with the steps (i), (ii), (iii), (iv), and (v) enable a self-refreshing cryptographic implementation of the method that strengthens a plurality of user-provided keys into a plurality of master secrets.

Optionally, a previous salt is maintained at the salt source, along with its next salt, until the next salt has been confirmed to be utilised successfully. Successful utilisation of the next salt means that the next salt has been used to generate its corresponding master secret, and this master secret has been successfully used for a further cryptographic operation. The confirmation of the successful utilisation of the next salt is received, for example, as a communication received from a device at which said cryptographic operation is implemented. Upon such confirmation, the previous salt can optionally be removed from a database/a memory of the salt source. This approach enables the method to effectively handle potential interruptions, without any adverse impact on usage of the at least one master secret.

As an example, a potential interruption may be caused by a network disconnection during use of the second master secret (which is generated

FI20255297 PRH 18 -12- 2025

using the third salt). In this case, if the previous salt (which is the second salt in this example) is maintained at the salt source, the previous salt can be used to regenerate its corresponding master secret (i.e., the first master secret in this example) for use during the interruption. If the previous salt were not maintained, there would be no master secret available during the interruption.

The first client terminal optionally retrieves the second salt from the salt source at the step (vi), to maintain continuity in the cryptographic process. The retrieval is implemented according to how the salt source is implemented, in a similar manner to how the first salt is retrieved at the step (i) of the method. The second nonce is optionally generated at the second moment of time, in a similar manner to the generation of the first nonce in the step (ii) of the method. The purpose of generating the second nonce is to introduce randomness into the cryptographic transformation of the (weak) second user-provided key into the (strong) second master secret, thereby preventing replay attacks and ensuring forward secrecy. The method ensures that each cryptographic computation of each master secret amongst the at least one master secret is associated with a distinct moment in time, reducing the risk of unauthorized reuse of previous nonce values. The hash function is optionally applied to process three inputs—the second salt, the second nonce, and either the first identification token or the second identification token—to produce the third salt. Typically, an "*identification token*" is a short-lived or session-bound authentication credential, for ensuring that replay attacks are prevented. Any identification token must be secret, and have sufficient entropy (meaning that its byte size must be at least as large as a given salt that is to be derived using the hash function). The first identification token may expire prior to applying the hash function at the step (viii), in which case the second identification token is used; otherwise, the first identification token may be re-used. A technical effect of re-using the first identification token is that there is no need to obtain

said token, thus saving a communication effort and avoiding security issues. The second identification token is issued later in time than the first identification token, and replaces the first identification token once the first identification token expires. The second identification token may be obtained on or before expiration of the first identification token. In this way, the method ensures cryptographic security since identification tokens are not used beyond their lifespan, and cryptographic derivations from such identification tokens always rely on a valid, latest identification token. The third salt extends a cryptographic chain of high-entropy randomness, ensuring that the second master secret derivation maintains strong entropy and resists dictionary attacks.

The second user-provided key is optionally obtained for the next round of cryptographic computations. The second user-provided key may be same as the first user-provided key or may be different from the first user-provided key. This flexibility allows the method to either reuse the user's existing credentials or require fresh credentials for enhanced security. The second user-provided key is entered by the user via the first client terminal's user interface. The use of the second user-provided key ensures that user authentication remains an integral part of the method for generating the at least one master secret.

At the step (x), the KDF operates on newly derived values (i.e., the third salt and the second user-provided key) to ensure cryptographic freshness of the second master secret. The second master secret can be used for same or different security applications as the first master secret. For example, the second master secret can be used for deriving one or more of a session-based encryption key, an authentication token, a digital signature, or similar. By leveraging a fresh derivation process for generating the second master secret, the method guarantees that past and future master secrets remain independent of each other, reducing the risk of cryptographic compromise.

Optionally, the first identification token has a first period of time to live and when applying the hash function of step (viii):

the first identification token is used if the first period of time has not passed, or

the second identification token is used if the first period of time has passed.

In this regard, the method ensures validity of cryptographic computations across sessions while accommodating identification token expiration policies. The first identification token is a time-bound token, which is valid only for the first period of time. In other words, the "*first period of time to live*" represents a validity window of the first identification token. This means that the first identification token can be used for deriving the second salt (and optionally, one or more subsequent salts of the second salt) only if the first identification token is valid according to the issuing identification token provider. The validity of a given identification token is determined by the identification token provider and may be indicated within the given identification token itself (for example, as an expiration time, a validity time period, or similar). The first period of time can vary significantly depending on the identification token provider and a type of identification token, and may range from a few minutes (such as session-based authentication tokens) to several hours (such as single sign-on tokens). Optionally, the first period of time ranges from 0.5 minutes to 24 hours. For example, the first period of time may be from 0.5 minutes, 1 minute, 2 minutes, 5 minutes, 10 minutes, 15 minutes, 30 minutes, 45 minutes, 1 hour, 2 hours, 4 hours, 6 hours or 12 hours up to 7 minutes, 12 minutes, 20 minutes, 40 minutes, 1.5 hours, 2.5 hours, 5 hours, 10 hours, 15 hours, 20 hours, or 24 hours. Optionally, the first client terminal comprises a software module which is configured to verify validity of identification tokens. Such verification may be done, for example, by checking the expiration time provided by the issuing

FI20255297 PRH 18 -12- 2025

identification token provider. This software module communicates a validity status of the first identification token to a processor of the first client terminal prior to applying the hash function of step (viii).

Optionally, if the first identification token is still valid at the time of executing step (viii), the method reuses the first identification token. This measure allows the first identification token to be securely reused within its lifespan. This prevents unnecessary requests for new identification tokens, optimizing performance and minimizing reliance on external authentication services (such as identification token providers). However, if the first identification token has expired, the method automatically switches to using the second identification token which is also optionally obtained from the identification token provider. The second identification token is generated using a fresh authentication challenge, ensuring continued security and mitigating risks associated with token reuse. This feature prevents unauthorized access by ensuring that expired tokens are not reused in cryptographic derivations, thereby eliminating the risk of replay attacks where an adversary attempts to use an old identification token to regenerate past authentication secrets. Additionally, this approach maintains secure session continuity without requiring user intervention to manually refresh identification tokens, as the method autonomously manages transitions between valid identification tokens. In other words, the method handles refreshing of identification tokens automatically, regardless of a refresh frequency, to maintain security.

Optionally, the first identification token is provided to the first client terminal from an identification token provider over a communication network. Optionally, prior to receiving the first identification token from the identification token provider, the first client terminal signs into a service of the identification token provider with user identification information associated with the user of the first client terminal. Upon successful authentication of the user identification information, the identification token provider generates the first identification token and

FI20255297 PRH 18 -12- 2025

provides the first identification token to the first client terminal. Throughout the present disclosure, the term "*identification token provider*" refers to a secure entity that is responsible for issuing cryptographic identification tokens that authenticate a user or a device (such as the first client terminal). Accordingly, examples of the identification token provider include, but are not limited to, a cloud-based authentication service, a federated identity provider, a trust provider that issues time-limited identification tokens, a local secure authentication module, or an external secure device. The local secure authentication module may be executing in the first client terminal, and may generate temporary identification tokens based on one or more authentication factors. This means that optionally, the first client terminal may comprise the identification token provider. The external secure device may, for example, be a secure device that is separate from the first client terminal and may be in possession of the user of the first client terminal. These various types of identification token providers generate short-lived cryptographic tokens that serve as credentials for deriving cryptographic salts. The communication network between the first client terminal and the identification token provider may be same as or different from the communication network between the first client terminal and the salt source. The communication network between the first client terminal and the identification token provider may, for example, be the Internet, a cellular network, or a private network. The first identification token may be obtained using an industry-standard protocol or a proprietary authentication mechanism.

By receiving the first identification token from the (trusted) identification token provider, the method incorporates a dynamic authentication factor, rather than solely relying on static and weak user-provided keys. This prevents attackers from forging credentials and introduces an additional layer of security. Furthermore, using the communication network for the first identification token retrieval allows the method to dynamically

refresh credentials, ensuring that cryptographic operations remain time-bound and resilient against replay attacks.

Optionally, the second identification token is provided to the first client terminal from an identification token provider over a communication network when the first period of time has passed. This identification token provider may be same as or different from the identification token provider which provides the first identification token. Likewise, the communication network over which the second identification token is provider may be same as or different from the communication network over which the first identification token is provided. The second identification token is provided automatically as a latest identification token for further cryptographic derivation (such as deriving the third salt), upon expiry of the first identification token (which happens once the first period of time has passed). Therefore, this mechanism enables continuous authentication. This process may involve a background refresh or a short interruption in service. Furthermore, the identification token provider is responsible for ensuring that cryptographic identification tokens remain time-sensitive and resistant to replay attacks through mechanisms such as nonces and token expiration policies.

The first salt is generated by a trust provider using a true random number generator (TRNG), and wherein the salt source comprises the trust provider. The trust provider is a secure trusted entity responsible for generating high-entropy cryptographic materials. The "*trust provider*" may be a centralized trust service, an authentication authority, a dedicated hardware security module (HSM), or the first client terminal (when the first client terminal executes the TRNG in a secure element environment). The trust provider ensures that the first salt is unpredictable (i.e., truly random) and is resistant to precomputed attacks. The TRNG is a specialized cryptographic component that produces genuinely random values (based, for example, on physical entropy sources, such as electronic noise or quantum fluctuations). The

TRNG produces non-deterministic values, making them ideal for cryptographic key generation. The first salt may also be cryptographically signed by the trust provider, to ensure authenticity and integrity of the first salt. The first salt may optionally be transferred securely from the trust provider to the storage medium. If the NFC chip is used as the storage medium, the first salt may be written directly to the NFC chip at the time of issuance. If the QR code is used as the storage medium, the first salt is encoded into a representation of the QR code, which can later be scanned and decoded for retrieval by the first client terminal. This ensures that the first salt remains accessible while minimizing security risks associated with the storage medium. By ensuring that the first salt is generated by the TRNG, the method eliminates predictability and ensures strong cryptographic entropy from its first step. This prevents brute-force and rainbow table attacks, ensuring that the subsequent cryptographic transformations built upon this first salt inherit its security guarantees.

Optionally, the identification token provider and the trust provider are a part of a trust system. This trust system serves the role of a vault, and ensures integrity, authenticity, and confidentiality of critical security elements like identification tokens and salts. Furthermore, the trust system ensures that the at least one master secret is strong, derivable, and secure.

The at least one master secret is used for performing one or more cryptographic operations. Optionally, in this regard, the one or more cryptographic operations comprise at least one of: an authentication operation, an authorization operation, an identification operation, an encryption operation, a digital signing operation, a secure digital transaction operation. Performing other types of cryptographic operations using the at least one master secret may also be feasible. The encryption operation may, for example, be performed for encrypting a communication session, or for encrypting data. In this regard, the

communication session may involve one or more of: an audio call, a video call, a chat, an email conversation, a workspace communication, a broadcast communication, an exchange of documents, a transaction, or similar exchanges, between two or more client terminals. It will be appreciated that the at least one master secret can be used for various use cases. Since the at least one master secret is a strong cryptographic key having high entropy, it can beneficially be utilized for example for enabling the one or more cryptographic operations without requiring expensive secure hardware. The cryptographic strength of the at least one master secret ensures that data involved in the one or more cryptographic operations remains confidential, and is resistant to attacks and unauthorized interception. The method beneficially enables runtime key generation, since the at least one master secret, which is to be used in the one or more cryptographic operations, may be directly computed into the memory of the first client terminal and used therefrom. Therefore, the first client terminal does not require a certified key vault or key generator for key management, allowing secure usage of the at least one master secret across various types of client terminals. Furthermore, when performing the one or more cryptographic operations using the at least one master secret, there is no need to utilize the trust provider, since the second salt or its subsequent salt(s) having high security and entropy is/are used in the generation of the at least one master secret. Upon using the at least one master secret for performing the one or more cryptographic operations, the at least one master secret may be discarded.

As an example, the method may be utilized in an execution mode of a wireless secure cryptographic device when the NFC chip is utilized as a cost-effective active storage medium. An active salt stored at the NFC chip may be automatically refreshed. This enables compliance with requirements of a separate secure storage medium within a software-based Wireless Secure Cryptographic Application (WSCA) solution. The

FI20255297 PRH 18 -12- 2025

software-based WSCA solution may function in conjunction with the trust system and can operate on consumer-grade devices used as the first client terminal.

Optionally, the at least one master secret is usable to encrypt at least one of: real-time data, stored data, in the communication session. Optionally, a cryptographic protocol is utilized for encrypting the communication session using the at least one master secret. The cryptographic protocol may be as AES (Advanced Encryption Standard) protocol, a TLS (Transport Layer Security) protocol, an end-to-end encryption scheme, or similar.

In an example, the at least one master secret may be used as a seed for deriving one or more crypto-products. The one or more crypto-products optionally comprise at least one of: a symmetric key, an asymmetric key pair, one or more keys for encrypting and/or warping other key(s), a nonce value or initialization vector, a secure key from an identification credential, a secure password, a derivative of the secure password, a session key, a one-time key, a pre-shared key (PSK) of a Transport Layer Security (TLS)-PSK model.

For example, the at least one master secret may be used to calculate a key pair for encrypting a communication session. The key pair may comprise a public key and a private key, wherein the public key may be shared between parties of the communication session. The private key may be used, at the first client device, to encrypt and/or decrypt messages and/or data exchanged in the communication session. Optionally, a software bot associated with the user of the first client device may be authorized to use the private key.

The present disclosure also relates to the first client terminal as described above. Various embodiments and variants disclosed above, with respect to the aforementioned method, apply *mutatis mutandis* to the first client terminal.

The first client terminal is configured to execute the steps of the method. The first client terminal comprises the user interface, the access interface, and the processor. By integrating these components therein, the first client terminal autonomously generates the at least one master secret with strong cryptographic strength and high entropy, without requiring expensive secure hardware. The first client terminal enables users to generate strong cryptographic master secret(s) on their personal devices while maintaining security against brute-force and replay attacks.

The "*user interface*" is a component through which the user interacts with the first client terminal. The user interface may be a touchscreen, a keyboard, a mouse, a speaker, a biometric scanner, or a similar input device. The user interface enables secure provision of the first user-provided key to the first client terminal.

Optionally, the access interface of the first client terminal comprises at least one of: a communication module, an interfacing module. The term "*access interface*" refers to a set of one or more modules which enable the first client terminal to receive data from and/or write (or store) data to other devices. These one or more modules may be implemented as a single integrated module, or as separate modules. Optionally, when the salt source comprises the trust provider, the communication module of the access interface is employed to receive one or more salts from the trust provider. Additionally or alternatively, optionally, when the salt source comprises the storage medium, the interfacing module of the access interface is employed to receive one or more salts from the trust provider.

Throughout the present disclosure, the term "*communication module*" refers to a component that is configured for establishing, managing, and securing communication between the first client terminal and one or more external entities over the communication network. The communication module is implemented to be compatible with the communication

network. The communication module may, for example, be a Wi-Fi module, a 5G module, or similar.

The "*interfacing module*" is a component that enables the first client terminal to interface with at least the storage medium. The interfacing module allows the first client terminal to obtain data from the storage medium and optionally also allows the first client terminal to write data to the storage medium. The implementation of the interfacing module may depend on how the storage medium is implemented, to ensure that the first client terminal can at least fetch required cryptographic data seamlessly from the storage medium. The interfacing module may be the NFC reader, a NFC reader and writer, the camera, the optical scanner, or a similar element.

The processor may be implemented as hardware, software, firmware or a combination of these. In particular, the processor is configured to execute the steps of the method described hereinabove.

Optionally, the processor is further configured to store the second salt at a storage medium, via an interfacing module of the access interface.

Optionally, the processor is further configured to:

(VI) retrieve the second salt from the salt source, via the access interface;

(VII) generate a second nonce at a second moment of time;

(VIII) apply the hash function to the second salt, the second nonce and one of: the first identification token or a second identification token to derive a third salt;

(IX) obtain a second user-provided key via the user interface; and

(X) apply the key derivation function to the third salt and the second user-provided key to generate a second master secret amongst the at least one master secret.

Optionally, the first identification token has a first period of time to live and when applying the hash function of the step VII, the processor is configured to

use the first identification token if the first period of time has not passed, or

use the second identification token if the first period of time has passed.

Optionally, in the first client terminal, a communication module of the access interface is configured to receive the first identification token from an identification token provider over a communication network.

Optionally, in the first client terminal, a communication module of the access interface is configured to receive the second identification token from an identification token provider over a communication network, when the first period of time has passed.

Optionally, the salt source comprises a storage medium, and wherein the storage medium is a near field communication chip or a quick response (QR) code.

Optionally, in the first client terminal, a communication module of the access interface is configured to receive the first salt generated by a trust provider using a true random number generator, wherein the salt source comprises the trust provider.

The processor is further configured to perform one or more cryptographic operations using the at least one master secret.

Optionally, the first client terminal further comprises a memory. The memory may be a secure memory which is designed to protect sensitive data from unauthorized access, modification, or extraction. The at least one master secret is generated directly into the memory. The memory may, for example, be a volatile memory (such as RAM), a software-based secure memory, or similar. Optionally, the first identification token and/or the second identification token is stored into the memory.

FI20255297 PRH 18 -12- 2025

The present disclosure also relates to the system as described above. Various embodiments and variants disclosed above, with respect to the aforementioned method and the aforementioned first client terminal, apply *mutatis mutandis* to the system. The system comprises the salt source, the identification token provider and the first client terminal, which collectively enable execution of the method.

Optionally, in the system, the first client terminal is further configured to store the second salt at a storage medium.

Optionally, the first client terminal is further configured to:

(F) retrieve the second salt from the salt source;

(G) generate a second nonce at a second moment of time;

(H) apply the hash function to the second salt, the second nonce and one of: the first identification token or a second identification token to derive a third salt;

(J) obtain a second user-provided key; and

(K) apply the key derivation function to the third salt and the second user-provided key to generate a second master secret amongst the at least one master secret.

Optionally, the first identification token has a first period of time to live and when applying the hash function of the step G, the first client terminal is configured to

use the first identification token if the first period of time has not passed, or

use the second identification token if the first period of time has passed.

Optionally, in the system, the identification token provider is configured to provide the first identification token to the first client terminal over a communication network.

Optionally, in the system, the identification token provider is configured to provide the second identification token to the first client terminal over a communication network, when the first period of time has passed.

Optionally, in the system, the salt source comprises a storage medium, and wherein the storage medium is a near field communication chip or a quick response (QR) code.

Optionally, the salt source comprises a trust provider configured to generate the first salt using a true random number generator.

In the system, the first client terminal is further configured to perform one or more cryptographic operations using the at least one master secret.

EXPERIMENTAL PART

A test was performed using an automatic cryptographic protocol verifier to check whether private parameters used in the method remain secret. The parameters were as follows:

first salt= URT_salt
first nonce= new IdTokenNonceNew:bitstring
first identification token= new IdTokenSecretNew:bitstring;
second salt= URT_saltNew= HASH (URT_salt, IdTokenSecretNew, IdTokenNonceNew)
first master secret= URT_pvNew=KDF(URT_pinNew,URT_saltNew) in
and subsequent master secrets= URT_pkNew= exp(g,URT_pvNew)

The queries used in the automatic cryptographic protocol verifier for such checking were:

query attacker(URT_pin), and
query attacker(URT_pinNew)

The automatic cryptographic protocol verifier returned the following results, confirming that the attacker cannot deduce such private parameters from interacting with the protocol defined in the method. The results are:

Query not attacker(K_ClientRegServ[]) is true.
Query not attacker(Vsk[]) is true.
Query not attacker(URT_pin[]) is true.
Query not attacker(URT_pinNew[]) is true.

The above proof assumes that the private parameters are not subject to brute-forcing. In the test performed, it was supposed that the used keys are long enough, However, that is not the case for “short weak secrets” like PINs and short passwords. To cover this case for the short weak secrets in the protocol, the possibility of offline attacks was checked using a function denoted by 'weaksecret'. More precisely, the following check was added:

weaksecret URT_pin
weaksecret URT_pinNew

The above lines check if the attacker cannot distinguish the actual PIN from a randomly chosen PIN, i.e., the attacker cannot carry out a dictionary attack. It was observed that:

Weak secret URT_pin is true.
Weak secret URT_pinNew is true.

This output confirms that it is true that the attacker cannot distinguish if a given (offline) PIN is the right PIN, that is using the trace of a protocol execution.

DETAILED DESCRIPTION OF THE DRAWINGS

Referring to FIG. 1, illustrated is a block diagram of how a method for generating at least one master secret is implemented, in accordance with an embodiment of the present disclosure. A first client terminal **102** retrieves a first salt **110** from a salt source **104**. A first nonce **120** is generated at a first moment of time. A hash function **HASH** is applied to the first salt **110**, the first nonce **120** and a first identification token **130**, to derive a second salt **140**. A first user-provided key **150** is obtained. A key derivation function **KDF** is applied to the second salt **140** and the first user-provided key **150** to generate a first master secret **160** amongst the at least one master secret. The first identification token **130** is optionally provided to the first client terminal **102** from an identification token provider **106** over a communication network **108**. The first user-provided key **150** is optionally provided via a user interface **112** of the first client terminal **102**. The first master secret **160** is optionally used perform one or more cryptographic operations **114**. Optionally, the first nonce **120** is derived in the first client terminal **102** from a high-resolution timestamp, which may be derived from an internal clock **116** of the first client terminal **102**.

Referring to FIG. 2, illustrated is a flowchart illustrating steps of the method for generating the at least one master secret, in accordance with an embodiment of the present disclosure. At step **202**, there is retrieved with a first client terminal, from a salt source, a first salt. At step **204**, a first nonce is generated at a first moment of time. At step **206**, a hash function is applied to the first salt, the first nonce and a first identification token to derive a second salt. At step **208**, a first user-provided key is obtained. At step **210**, a key derivation function is applied to the second salt and the first user-provided key to generate a first master secret amongst the at least one master secret.

The aforementioned steps are only illustrative and other alternatives can also be provided where one or more steps are added, one or more steps are removed, or one or more steps are provided in a different sequence without departing from the scope of the claims herein.

Referring to FIG. 3, illustrated is a block diagram-based timeline of how the method is used to generate a plurality of master secrets, in accordance with an embodiment of the present disclosure. At time **t0**, a first salt **310** is retrieved from a salt source **304**. In this example, the salt source **304** is optionally shown to comprise a storage medium **304A** and a trust provider **304B**. Let us consider, for example, that the first salt **310** is retrieved from the trust provider **304B**. A first nonce **320** is generated at a first moment of time **t1**. A hash function **HASH** is applied to the first salt **310**, the first nonce **320** and a first identification token **330**, to derive a second salt **340**. The first identification token **330** is obtained at time **t2**, for example. A first user-provided key **350** is obtained, for example at time **t3**. A key derivation function **KDF** is applied to the second salt **340** and the first user-provided key **350** to generate a first master secret **360A** amongst the plurality of master secrets. Optionally, the second salt **340** is stored at the storage medium **304A**, at time **t4**. Then, optionally, the second salt **340** is retrieved from the salt source **304** (and in particular, from the storage medium **304A**). A second nonce **370** is generated at a second moment of time **t5**. A hash function **HASH** is applied to the second salt **340**, the second nonce **370** and the first identification token **330**, to derive a third salt **380**. A second user-provided key **390** is obtained. A key derivation function **KDF** is applied to the third salt **380** and the second user-provided key **390** to generate a second master secret **360B** amongst the plurality of master secrets. Optionally, the first identification token **330** has a first period of time **tP1** to live and when applying the hash function **HASH** for deriving the third salt **380**, the first identification token **330** is used if the first period of time **tP1** has not passed, otherwise a second identification

token **392** is used if the first period of time **tP1** has passed. For example, as shown, at time **t6**, the first period of time **tP1** has not passed, so the first identification token **330** is used. It will be appreciated that there could be one or more intermediate nonces generated between the generation of the first nonce **320** and the second nonce **370**. Furthermore, optionally, the second salt **340** is maintained at the salt source **304**, along with the third salt **380**, until the third salt **380** has been confirmed to be utilised successfully. The third salt **380** is utilised successfully when the third salt **380** has been used to generate the second master secret **360B** successfully and when the second master secret **360B** has been successfully used for a further cryptographic operation.

Referring to FIG. 4, illustrated is a block diagram of a first client terminal **402** for generating the at least one master secret, in accordance with an embodiment of the present disclosure. The first client terminal **402** comprises a user interface **404**, an access interface **406**, and a processor **408** coupled to the user interface **404** and the access interface **406**. The processor **408** is configured to: (I) retrieve a first salt from a salt source, via the access interface **406**; (II) generate a first nonce at a first moment of time; (III) apply a hash function to the first salt, the first nonce and a first identification token to derive a second salt; (IV) obtain a first user-provided key via the user interface **404**; and (V) apply a key derivation function to the second salt and the first user-provided key to generate a first master secret amongst at least one master secret. Optionally, the access interface **406** comprises at least one of: a communication module **410**, an interfacing module **412**. The processor **408** may optionally be further configured to store the second salt at a storage medium, via the interfacing module **412** of the access interface **406**. Optionally, the communication module **410** of the access interface **406** is configured to receive at least the first identification token. Optionally, the first client

terminal **402** further comprises a memory **414**. The first master secret is computed into the memory **414**.

Referring to FIG. 5, illustrated is a schematic illustration of a system **500** for generating the at least one master secret, in accordance with an embodiment of the present disclosure. The system **500** comprises a salt source **502**, an identification token provider **504**, and a first client terminal **506** communicably coupled to the salt source **502** and the identification token provider **504**. The first client terminal **506** is configured to: (A) retrieve a first salt from the salt source **502**; (B) generate a first nonce at a first moment of time; (C) apply a hash function to the first salt, the first nonce and a first identification token to derive a second salt; (D) obtain a first user-provided key; and (E) apply a key derivation function to the second salt and the first user-provided key to generate a first master secret amongst the at least one master secret. Optionally, the identification token provider **504** is configured to provide the first identification token to the first client terminal **506** over a communication network **508**. The identification token provider **504** may also be configured to provide a second identification token to the first client terminal **506** over a communication network **508**. Optionally, the salt source **502** comprises at least one of: a storage medium **509**, a trust provider **510**. Optionally, the trust provider **510** is configured to generate the first salt. The first client terminal **506** is shown to comprise a user interface **512**, an access interface **514**, a processor **516**, and a memory **520**. The user interface **512** enables in obtaining the first user-provided key, from a user of the first client terminal **506**. The access interface **514** optionally comprises at least one of: an interfacing module **517**, a communication module **518**. The communication module **518** is configured to retrieve at least the first salt from the trust provider **510**, for example. The interfacing module **517** may be configured to store one or more salts at the storage medium **509**. The processor **516** is configured to execute processing steps for which the first client terminal

506 is configured. The communication module **518** is configured to receive the first identification token, and optionally, a second identification token, from the identification token provider **504** over the communication network **508**. The communication module **518** is optionally configured to receive the first salt generated by the trust provider **510** using a true random number generator **522**. The first master secret is computed into the memory **520**. Optionally, the identification token provider **504** and the trust provider **510** are a part of a trust system **524**.

Referring to FIG. 6, illustrated is a sequence diagram illustrating how at least one master secret is generated, in accordance with an embodiment of the present disclosure. In step **S6.1**, a first client terminal **602** may sign into a service of an identification token provider **604** with user identification information associated with a user of the first client terminal **602**. In step **S6.2**, the identification token provider **604** generates a first identification token **606**. The first identification token **606** is provided from the identification token provider **604** to the first client terminal **602** in step **S6.3**. Next, in steps **S6.4** and **S6.5**, the first client terminal **602** retrieves, from a salt source **608**, a first salt **610**. In step **S6.6** the first identification token **606**, the first salt **610** and a first nonce **612** are used, by applying a hash function thereon, to generate a second salt **614**. The first nonce **612** can, for example, be derived from an internal clock **616** of the first client terminal **602**. In step **S6.7**, the second salt **614** is used together with a first user provided key **618**, by applying a key derivation function thereon, to generate a first master secret **620**.

FIGs. 1, 2, 3, 4, 5, and 6 are merely examples, which should not unduly limit the scope of the claims herein. A person skilled in the art will recognize many variations, alternatives, and modifications of embodiments of the present disclosure.

CLAIMS

1. A method for generating at least one master secret, the method comprising:

(i) retrieving, with a first client terminal (102, 402, 506, 602), from a salt source (104, 304, 502, 608), a first salt (110, 310, 610), wherein the first salt is generated by a trust provider (304B, 510) using a true random number generator (TRNG) (522);

(ii) generating a first nonce (120, 320, 612) at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);

(iii) applying a hash function (HASH) to the first salt, the first nonce and a first identification token (130, 330, 606) to derive a second salt (140, 340, 614), wherein the first identification token is generated by an identification token provider (106, 504, 604) and provided to the first client terminal for authenticating a user of the first client terminal, wherein the identification token is a short-lived or session-bound authentication credential;

(iv) obtaining a first user-provided key (150, 350, 618); and

(v) applying a key derivation function (KDF) to the second salt and the first user-provided key to generate a first master secret (160, 360A, 620) amongst the at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

2. A method according to the claim 1, wherein the method further comprises storing the second salt (140, 340, 614) to a storage medium (304A, 509).

3. A method according to any of the claims 1 or 2, wherein the method further comprises;

(vi) retrieving, with the first client terminal (102, 402, 506, 602), from the salt source (104, 304, 502, 608), the second salt (140, 340, 614);

FI20255297 PRH 18 -12- 2025

(vii) generating a second nonce (370) at a second moment of time (t5);
(viii) applying the hash function (HASH) to the second salt, the second nonce and one of: the first identification token (130, 330, 606) or a second identification token (392) to derive a third salt (380);
(ix) obtaining a second user-provided key (390); and
(x) applying the key derivation function (KDF) to the third salt and the second user-provided key to generate a second master secret (360B) amongst the at least one master secret.

4. A method according to claim 3, wherein the first identification token (130, 330, 606) has a first period of time (tP1) to live and when applying the hash function (HASH) of the step viii
the first identification token is used if the first period of time has not passed, or
the second identification token (392) is used if the first period of time has passed.

5. A method according to any of the preceding claims, wherein the first identification token (130, 330, 606) is provided to the first client terminal (102, 402, 506, 602) from the identification token provider (106, 504, 604) over a communication network (108, 508).

6. A method according to claim 4, wherein the second identification token (392) is provided to the first client terminal (102, 402, 506, 602), from the identification token provider (106, 504, 604) over a communication network (108, 508), when the first period of time (tP1) has passed.

7. A method according to any of the preceding claims wherein the salt source (104, 304, 502, 608) comprises a storage medium (304A, 509), and wherein the storage medium is a near field communication chip or a quick response (QR) code.

8. A method according to any of the preceding claims wherein the salt source (104, 304, 502, 608) comprises the trust provider.

9. A first client terminal (102, 402, 506, 602) comprising:
a user interface (112, 404, 512);
an access interface (406, 514); and
a processor (408, 516) coupled to the user interface and the access interface, wherein the processor is configured to:
(I) retrieve a first salt (110, 310, 610) from a salt source (104, 304, 502, 608), via the access interface, wherein the first salt is generated by a trust provider (304B, 510) using a true random number generator (TRNG) (522);
(II) generate a first nonce (120, 320, 612) at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);
(III) apply a hash function (HASH) to the first salt, the first nonce and a first identification token (130, 330, 606) to derive a second salt (140, 340, 614), wherein the first identification token is generated by an identification token provider (106, 504, 604) and provided to the first client terminal for authenticating a user of the first client terminal, wherein the identification token is a short-lived or session-bound authentication credential;
(IV) obtain a first user-provided key (150, 350, 618) via the user interface; and
(V) apply a key derivation function (KDF) to the second salt and the first user-provided key to generate a first master secret (160, 360A, 620) amongst at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

10. A first client terminal (102, 402, 506, 602) according to claim 9, wherein the processor (408, 516) is further configured to store the

second salt (140, 340, 614) at a storage medium (304A, 509), via an interfacing module (412, 517) of the access interface (406, 514).

11. A first client terminal (102, 402, 506, 602) according to any of the claims 9 or 10, wherein the processor (408, 516) is further configured to:
(VI) retrieve the second salt (140, 340, 614) from the salt source (104, 304, 502, 608), via the access interface (406, 514);
(VII) generate a second nonce (370) at a second moment of time (t5);
(VIII) apply the hash function (HASH) to the second salt, the second nonce and one of: the first identification token (130, 330, 606) or a second identification token (392) to derive a third salt (380);
(IX) obtain a second user-provided key (390) via the user interface (112, 404, 512); and
(X) apply the key derivation function (KDF) to the third salt and the second user-provided key to generate a second master secret (360B) amongst the at least one master secret.

12. A first client terminal (102, 402, 506, 602) according to claim 11, wherein the first identification token (130, 330, 606) has a first period of time (tP1) to live and when applying the hash function (HASH) of the step VII, the processor (408, 516) is configured to
use the first identification token if the first period of time has not passed, or
use the second identification token (392) if the first period of time has passed.

13. A first client terminal (102, 402, 506, 602) according to any of the claims 9-12, wherein a communication module (410, 518) of the access interface (406, 514) is configured to receive the first identification token (130, 330, 606) from the identification token provider (106, 504, 604) over a communication network (108, 508).

14. A first client terminal (102, 402, 506, 602) according to claim 12, wherein a communication module (410, 518) of the access interface (406, 514) is configured to receive the second identification token (392) from the identification token provider (106, 504, 604) over a communication network (108, 508), when the first period of time (tP1) has passed.

15. A first client terminal (102, 402, 506, 602) according to any of the claims 9-14, wherein the salt source (104, 304, 502, 608) comprises a storage medium (304A, 509), and wherein the storage medium is a near field communication chip or a quick response (QR) code.

16. A first client terminal (102, 402, 506, 602) according to any of the claims 9-15, wherein a communication module (410, 518) of the access interface (406, 514) is configured to receive the first salt (110, 310, 610), wherein the salt source (104, 304, 502, 608) comprises the trust provider.

17. A system (500) for generating at least one master secret, the system comprising:
a salt source (104, 304, 502, 608);
an identification token provider (106, 504, 604); and
a first client terminal (102, 402, 506, 602) communicably coupled to the salt source and the identification token provider, wherein the first client terminal is configured to:
(A) retrieve a first salt (110, 310, 610) from the salt source, wherein the first salt is generated by a trust provider (304B, 510) using a true random number generator (TRNG) (522);
(B) generate a first nonce (120, 320, 612) at a first moment of time (t1), wherein the first nonce is derived in the first client terminal from one of: a high-resolution timestamp, a counter-based system, or a pseudo-random number generator (PRNG);

(C) apply a hash function (HASH) to the first salt, the first nonce and a first identification token (130, 330, 606) to derive a second salt (140, 340, 614), wherein the first identification token is generated by an identification token provider and provided to the first client terminal for authenticating a user of the first client terminal, wherein the identification token is a short-lived or session-bound authentication credential;

(D) obtain a first user-provided key (150, 350, 618); and

(E) apply a key derivation function (KDF) to the second salt and the first user-provided key to generate a first master secret (160, 360A, 620) amongst the at least one master secret, wherein the at least one master secret is used for performing one or more cryptographic operations.

18. A system (500) according to claim 17, wherein the first client terminal (102, 402, 506, 602) is further configured to store the second salt (140, 340, 614) at a storage medium (304A, 509).

19. A system (500) according to any of the claims 17 or 18, wherein the first client terminal (102, 402, 506, 602) is further configured to:

(F) retrieve the second salt (140, 340, 614) from the salt source (104, 304, 502, 608);

(G) generate a second nonce (370) at a second moment of time (t5);

(H) apply the hash function (HASH) to the second salt, the second nonce and one of: the first identification token (130, 330, 606) or a second identification token (392) to derive a third salt (380);

(J) obtain a second user-provided key (390); and

(K) apply the key derivation function (KDF) to the third salt and the second user-provided key to generate a second master secret (360B) amongst the at least one master secret.

20. A system (500) according to claim 19, wherein the first identification token (130, 330, 606) has a first period of time (tP1) to live and when applying the hash function (HASH) of the step G, the first client terminal (102, 402, 506, 602) is configured to

use the first identification token if the first period of time has not passed, or

use the second identification token (392) if the first period of time has passed.

21. A system (500) according to any of the claims 17-20, wherein the identification token provider (106, 504, 604) is configured to provide the first identification token (130, 330, 606) to the first client terminal (102, 402, 506, 602) over a communication network (108, 508).

22. A system (500) according to claim 20, wherein the identification token provider (106, 504, 604) is configured to provide the second identification token (392) to the first client terminal (102, 402, 506, 602) over a communication network (108, 508), when the first period of time (tP1) has passed.

23. A system (500) according to any of the claims 17-22, wherein the salt source (104, 304, 502, 608) comprises a storage medium (304A, 509), and wherein the storage medium is a near field communication chip or a quick response (QR) code.

24. A system (500) according to any of the claims 17-23, wherein the salt source (104, 304, 502, 608) comprises a trust provider (304B, 510).

PATENTTIVAATIMUKSET

1. Menetelmä ainakin yhden pääsalaisuuden luomiseksi, jolloin menetelmä käsittää seuraavat:

(i) noudetaan ensimmäisellä asiakaspäätteellä (102, 402, 506, 602) suolalähteestä (104, 304, 502, 608) ensimmäinen suola (110, 310, 610), jolloin ensimmäisen suolan on luonut luottamuspalveluntarjoaja (304B, 510) käyttämällä tosisatunnaislukugeneraattoria (TRNG) (522);

(ii) luodaan ensimmäinen nonce (120, 320, 612) ensimmäisellä ajanhetkellä (t1), jolloin ensimmäinen nonce johdetaan ensimmäisessä asiakaspäätteessä jostakin seuraavista: korkean resoluution aikaleima, laskuripohjainen järjestelmä tai pseudosatunnaislukugeneraattori (PRNG);

(iii) sovelletaan tiivistefunktiota (HASH) ensimmäiseen suolaan, ensimmäiseen nonceen ja ensimmäiseen tunnistustunnisteeseen (130, 330, 606) toisen suolan (140, 340, 614) johtamiseksi, jolloin tunnistustunnisteen tarjoaja (106, 504, 604) luo ensimmäisen tunnistustunnisteen ja se toimitetaan ensimmäiselle asiakaspäätteelle ensimmäisen asiakaspäätteen käyttäjän todentamiseksi, jolloin tunnistustunniste on lyhytaikainen tai istuntokohtainen todennusvarmenne;

(iv) hankitaan ensimmäinen käyttäjän toimittama avain (150, 350, 618); ja

(v) sovelletaan avainjohdosfunktiota (KDF) toiseen suolaan ja ensimmäiseen käyttäjän toimittamaan avaimeen ensimmäisen pääsalaisuuden (160, 360A, 620) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta, jolloin kyseistä ainakin yhtä pääsalaisuutta käytetään yhden tai useamman kryptografisen toiminnon suorittamiseen.

2. Patenttivaatimuksen 1 mukainen menetelmä, jolloin menetelmä käsittää lisäksi toisen suolan (140, 340, 614) tallentamisen tallennusvälineelle (304A, 509).

3. Patenttivaatimuksen 1 tai 2 mukainen menetelmä, jolloin menetelmä käsittää lisäksi seuraavat:

(vi) noudetaan ensimmäisellä asiakaspäätteellä (102, 402, 506, 602) suolalähteestä (104, 304, 502, 608) toinen suola (140, 340, 614);

(vii) muodostetaan toinen nonce (370) toisella ajanhetkellä (t5);

(viii) sovelletaan tiivistefunktiota (HASH) toiseen suolaan, toiseen nonceen ja yhteen jostakin seuraavista: ensimmäisestä tunnistustunnisteesta (130, 330, 606) tai toisesta tunnistustunnisteesta (392) kolmannen suolan (380) johtamiseksi;

(ix) hankitaan toinen käyttäjän toimittama avain (390); ja

(x) sovelletaan avainjohdosfunktiota (KDF) kolmanteen suolaan ja toiseen käyttäjän toimittamaan avaimeen toisen pääsalaisuuden (360B) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta.

4. Patenttivaatimuksen 3 mukainen menetelmä, jossa ensimmäisellä tunnistustunnisteella (130, 330, 606) on ensimmäinen käyttöajanjakso (tP1) ja vaiheen viii tiivistefunktiota (HASH) sovellettaessa

ensimmäistä tunnistustunnistetta käytetään, jos ensimmäinen ajanjakso ei ole kulunut loppuun, tai

toista tunnistustunnistetta (392) käytetään, jos ensimmäinen ajanjakso on kulunut loppuun.

5. Jonkin edellisen patenttivaatimuksen mukainen menetelmä, jossa ensimmäinen tunnistustunniste (130, 330, 606) toimitetaan ensimmäiselle

asiakaspäätteelle (102, 402, 506, 602) tunnistustunnisteen tarjoajalta (106, 504, 604) tietoliikenneverkon (108, 508) kautta.

6. Patenttivaatimuksen 4 mukainen menetelmä, jossa toinen tunnistustunniste (392) toimitetaan ensimmäiselle asiakaspäätteelle (102, 402, 506, 602) tunnistustunnisteen tarjoajalta (106, 504, 604) tietoliikenneverkon (108, 508) kautta, kun ensimmäinen ajanjakso (tP1) on kulunut loppuun.

7. Jonkin edellisen patenttivaatimuksen mukainen menetelmä, jossa suolalähde (104, 304, 502, 608) käsittää tallennusvälineen (304A, 509) ja jossa tallennusväline on lähikenttäviestintä (NFC) -siru tai pikavaste (QR) -koodi.

8. Jonkin edellisen patenttivaatimuksen mukainen menetelmä, jossa suolalähde (104, 304, 502, 608) käsittää luottamuspalveluntarjoajan.

9. Ensimmäinen asiakaspääte (102, 402, 506, 602), joka käsittää

käyttöliittymän (112, 404, 512);

pääsyliittymän (406, 514); ja

prosessorin (408, 516), joka on kytketty käyttöliittymään ja pääsyliittymään, jolloin prosessori on konfiguroitu:

(I) noutamaan ensimmäinen suola (110, 310, 610) suolalähteestä (104, 304, 502, 608) pääsyliittymän kautta, jolloin luottamuspalveluntarjoaja (304B, 510) luo ensimmäisen suolan käyttämällä tosisatunnaislukugeneraattoria (TRNG) (522);

(II) luomaan ensimmäinen nonce (120, 320, 612) ensimmäisellä ajanhetkellä (t1), jolloin ensimmäinen nonce johdetaan ensimmäisessä asiakaspäätteessä jostakin seuraavista: korkean resoluution aikaleima, laskuripohjainen järjestelmä tai pseudosatunnaislukugeneraattori (PRNG);

(III) soveltamaan tiivistefunktiota (HASH) ensimmäiseen suolaan, ensimmäiseen nonceen ja ensimmäiseen tunnistustunnisteeseen (130, 330, 606) toisen suolan (140, 340, 614) johtamiseksi, jolloin tunnistustunnisteen tarjoaja (106, 504, 604) luo ensimmäisen tunnistustunnisteen ja se toimitetaan ensimmäiselle asiakaspäätteelle ensimmäisen asiakaspäätteen käyttäjän todentamiseksi, jolloin tunnistustunniste on lyhytaikainen tai istuntokohtainen todennusvarmenne;

(IV) hankkimaan ensimmäinen käyttäjän toimittama avain (150, 350, 618) käyttöliittymän kautta; ja

(V) soveltamaan avainjohdosfunktiota (KDF) toiseen suolaan ja ensimmäiseen käyttäjän toimittamaan avaimeen ensimmäisen pääsalaisuuden (160, 360A, 620) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta, jolloin kyseistä ainakin yhtä pääsalaisuutta käytetään yhden tai useamman kryptografisen toiminnon suorittamiseen.

10. Patenttivaatimuksen 9 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa prosessori (408, 516) on lisäksi konfiguroitu tallentamaan toinen suola (140, 340, 614) tallennusvälineelle (304A, 509) pääsyliittymän (406, 514) liittymämoduulin (412, 517) kautta.

11. Patenttivaatimuksen 9 tai 10 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa prosessori (408, 516) on lisäksi konfiguroitu

(VI) noutamaan toinen suola (140, 340, 614) suolalähteestä (104, 304, 502, 608) pääsyliittymän (406, 514) kautta;

(VII) muodostamaan toinen nonce (370) toisella ajanhetkellä (t5);

(VIII) soveltamaan tiivistefunktiota (HASH) toiseen suolaan, toiseen nonceen ja yhteen jostakin seuraavista: ensimmäisestä tunnistustunnisteesta (130, 330, 606) tai toisesta tunnistustunnisteesta (392) kolmannen suolan (380) johtamiseksi;

(IX) hankkimaan toinen käyttäjän toimittaman avaimen (390) käyttöliittymän (112, 404, 512) kautta; ja

(X) soveltamaan avainjohdosfunktiota (KDF) kolmanteen suolaan ja toiseen käyttäjän toimittamaan avaimeen toisen pääsalaisuuden (360B) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta.

12. Patenttivaatimuksen 11 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa ensimmäisellä tunnistustunnisteella (130, 330, 606) on ensimmäinen käyttöajanjakso (tP1) ja sovellettaessa vaiheen VII tiivistefunktiota (HASH) prosessori (408, 516) on konfiguroitu

käyttämään ensimmäistä tunnistustunnistetta, jos ensimmäinen ajanjakso ei ole kulunut loppuun, tai

käyttämään toista tunnistustunnistetta (392), jos ensimmäinen ajanjakso on kulunut loppuun.

13. Jonkin patenttivaatimuksen 9–12 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa pääsyliittymän (406, 514) tietoliikennemoduuli (410, 518) on konfiguroitu vastaanottamaan ensimmäinen tunnistustunniste (130, 330, 606) tunnistustunnisteen tarjoajalta (106, 504, 604) tietoliikenneverkon (108, 508) kautta.

14. Patenttivaatimuksen 12 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa pääsyliittymän (406, 514) tietoliikennemoduuli (410, 518) on konfiguroitu vastaanottamaan toinen tunnistustunniste (392) tunnistustunnisteen tarjoajalta (106, 504, 604) tietoliikenneverkon (108, 508) kautta, kun ensimmäinen ajanjakso (tP1) on kulunut loppuun.

15. Jonkin patenttivaatimuksen 9–14 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa suolalähde (104, 304, 502, 608) käsittää tallennusvälineen (304A, 509) ja jossa tallennusväline on lähikenttäviestintä (NFC) -siru tai pikavaste (QR) -koodi.

16. Jonkin patenttivaatimuksen 9–15 mukainen ensimmäinen asiakaspääte (102, 402, 506, 602), jossa pääsyliittymän (406, 514) tietoliikennemoduuli (410, 518) on konfiguroitu vastaanottamaan ensimmäinen suola (110, 310, 610), jolloin suolan lähde (104, 304, 502, 608) käsittää luottamuspalveluntarjoajan.

17. Järjestelmä (500) ainakin yhden pääsalaisuuden luomiseksi, jolloin järjestelmä käsittää:

suolalähteen (104, 304, 502, 608);

tunnistustunnisteen tarjoajan (106, 504, 604); ja

ensimmäisen asiakaspäätteen (102, 402, 506, 602), joka on yhdistetty tiedonsiirtoyhteydellä suolalähteeseen ja tunnistustunnisteen tarjoajaan, jolloin ensimmäinen asiakaspääte on konfiguroitu:

(A) noutamaan ensimmäinen suola (110, 310, 610) suolalähteestä, jolloin ensimmäisen suolan luo luottamuspalveluntarjoaja (304B, 510) käyttämällä tosisatunnaislukugeneraattoria (TRNG) (522);

(B) luomaan ensimmäinen nonce (120, 320, 612) ensimmäisellä ajanhetkellä (t1), jolloin ensimmäinen nonce johdetaan ensimmäisessä asiakaspäätteessä jostakin seuraavista: korkean resoluution aikaleima, laskuripohjainen järjestelmä tai pseudosatunnaislukugeneraattori (PRNG);

(C) soveltamaan tiivistefunktiota (HASH) ensimmäiseen suolaan, ensimmäiseen nonceen ja ensimmäiseen tunnistustunnisteeseen (130, 330, 606) toisen suolan (140, 340, 614) johtamiseksi, jolloin tunnistustunnisteen tarjoaja luo ensimmäisen tunnistustunnisteen ja se toimitetaan ensimmäiselle asiakaspäätteelle ensimmäisen asiakaspäätteen käyttäjän todentamiseksi, jolloin tunnistustunniste on lyhytaikainen tai istuntokohtainen todennusvarmenne;

(D) hankkimaan käyttäjän toimittama ensimmäinen avain (150, 350, 618); ja

(E) soveltamaan avainjohdosfunktiota (KDF) toiseen suolaan ja ensimmäiseen käyttäjän toimittamaan avaimeen ensimmäisen pääsalaisuuden (160, 360A, 620) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta, jolloin kyseistä ainakin yhtä pääsalaisuutta käytetään yhden tai useamman kryptografisen toiminnon suorittamiseen.

18. Patenttivaatimuksen 17 mukainen järjestelmä (500), jossa ensimmäinen asiakaspääte (102, 402, 506, 602) on lisäksi konfiguroitu tallentamaan toinen suola (140, 340, 614) tallennusvälineelle (304A, 509).

19. Patenttivaatimuksen 17 tai 18 mukainen järjestelmä (500), jossa ensimmäinen asiakaspääte (102, 402, 506, 602) on lisäksi konfiguroitu

(F) noutamaan toinen suola (140, 340, 614) suolalähteestä (104, 304, 502, 608)

(G) muodostamaan toinen nonce (370) toisella ajanhetkellä (t5)

(H) soveltamaan tiivistefunktiota (HASH) toiseen suolaan, toiseen nonceen ja yhteen jostakin seuraavista: ensimmäisestä tunnistustunnisteesta (130, 330, 606) tai toisesta tunnistustunnisteesta (392) kolmannen suolan (380) johtamiseksi

(J) hankkimaan toinen käyttäjän toimittama avain (390) ja

(K) soveltamaan avainjohdosfunktiota (KDF) kolmanteen suolaan ja toiseen käyttäjän toimittamaan avaimeen toisen pääsalaisuuden (360B) luomiseksi kyseisen ainakin yhden pääsalaisuuden joukosta.

20. Patenttivaatimuksen 19 mukainen järjestelmä (500), jossa ensimmäisellä tunnistustunnisteella (130, 330, 606) on ensimmäinen

käyttöajanjakso (tP1) ja sovellettaessa vaiheen G tiivistefunktiota (HASH) ensimmäinen asiakaspääte (102, 402, 506, 602) on konfiguroitu

käyttämään ensimmäistä tunnistustunnistetta, jos ensimmäinen ajanjakso ei ole kulunut loppuun, tai

käyttämään toista tunnistustunnistetta (392), jos ensimmäinen ajanjakso on kulunut loppuun.

21. Jonkin patenttivaatimuksen 17–20 mukainen järjestelmä (500), jossa tunnistustunnisteen tarjoaja (106, 504, 604) on konfiguroitu toimittamaan ensimmäinen tunnistustunniste (130, 330, 606) ensimmäiselle asiakaspäätteelle (102, 402, 506, 602) tietoliikenneverkon (108, 508) kautta.

22. Patenttivaatimuksen 20 mukainen järjestelmä (500), jossa tunnistustunnisteen tarjoaja (106, 504, 604) on konfiguroitu toimittamaan toinen tunnistustunniste (392) ensimmäiselle asiakaspäätteelle (102, 402, 506, 602) tietoliikenneverkon (108, 508) kautta, kun ensimmäinen ajanjakso (tP1) on kulunut loppuun.

23. Jonkin patenttivaatimuksen 17–22 mukainen järjestelmä (500), jossa suolalähde (104, 304, 502, 608) käsittää tallennusvälineen (304A, 509) ja jossa tallennusväline on lähikenttäviestintä (NFC) -siru tai pikavaste (QR) -koodi.

24. Jonkin patenttivaatimuksen 17–23 mukainen järjestelmä (500), jossa suolalähde (104, 304, 502, 608) käsittää luottamuspalveluntarjoajan (304B, 510).

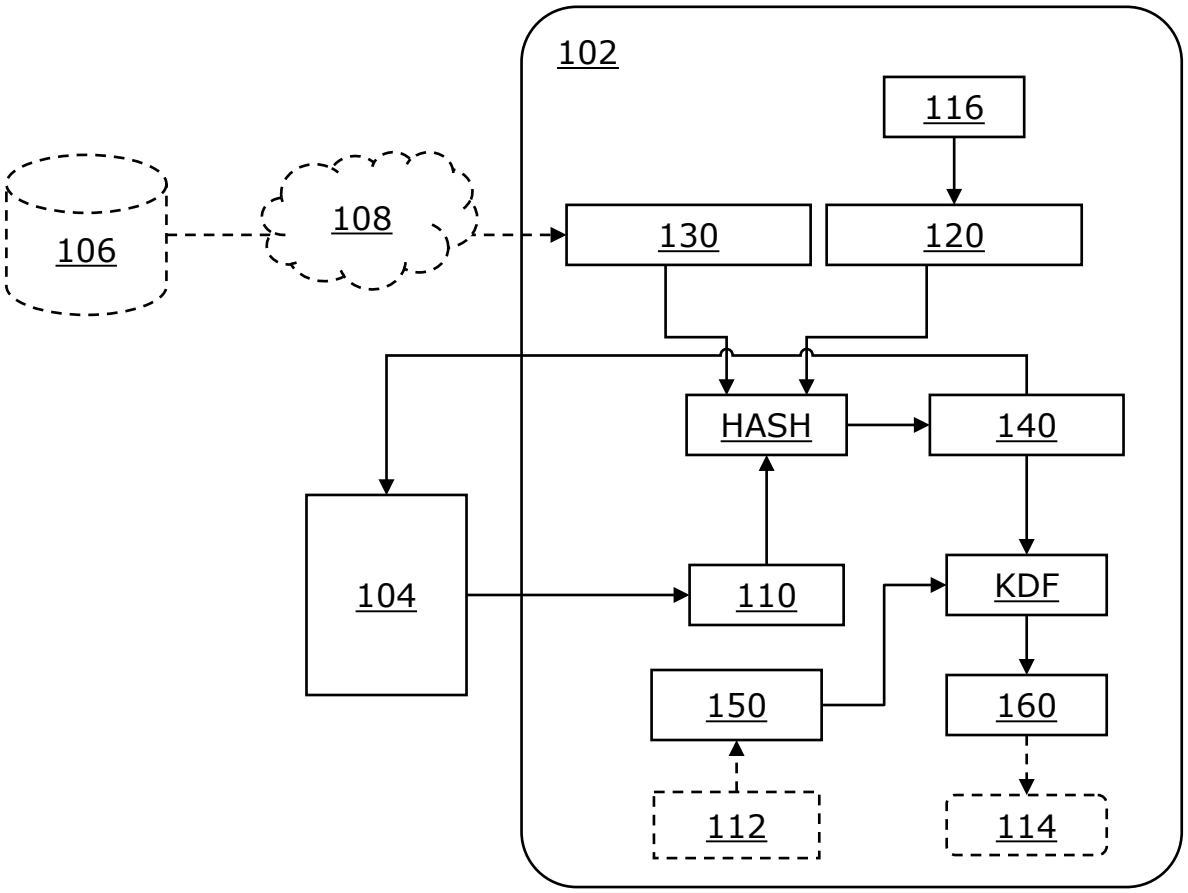


FIG. 1

20255297 PRH 31 -03- 2025

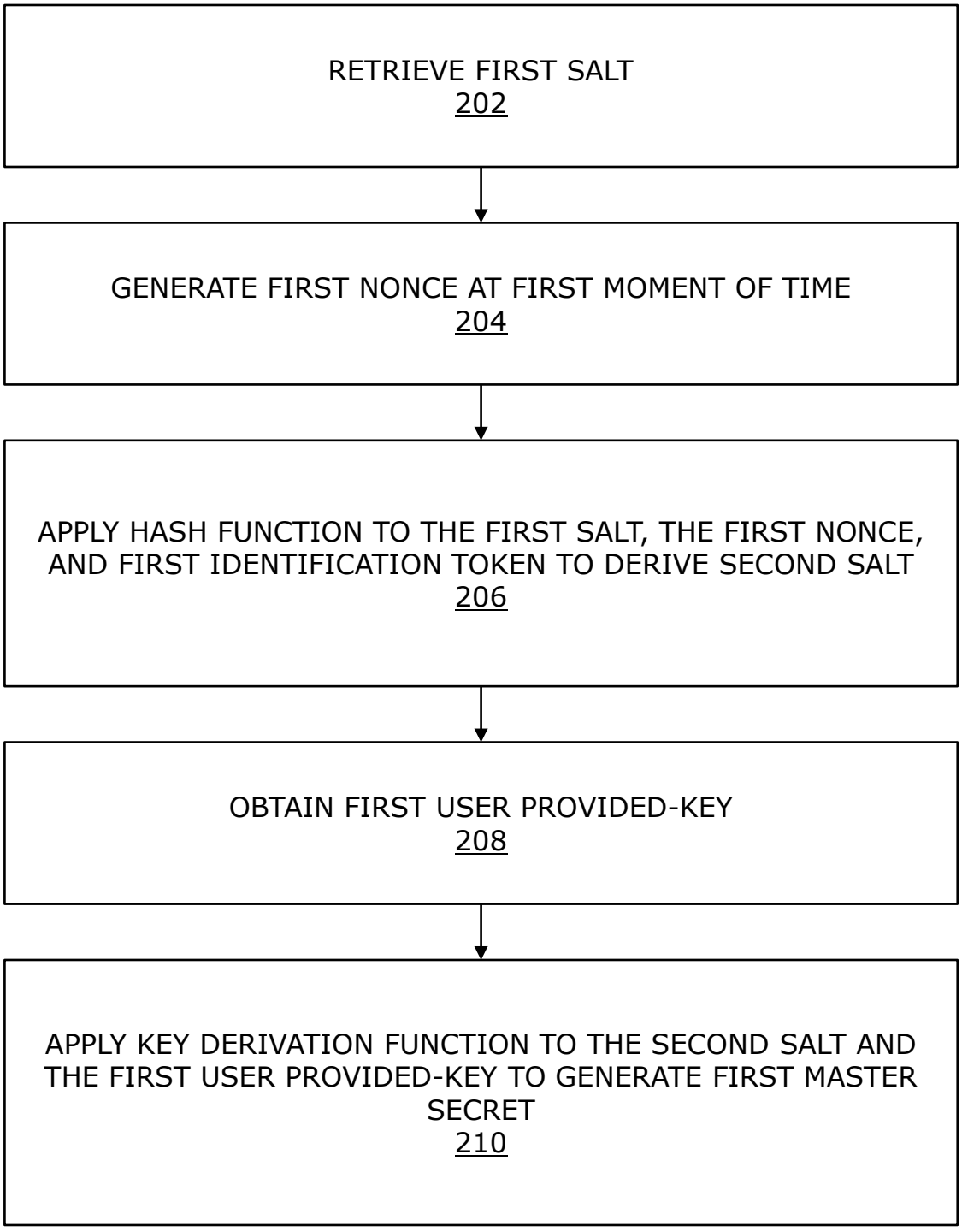


FIG. 2

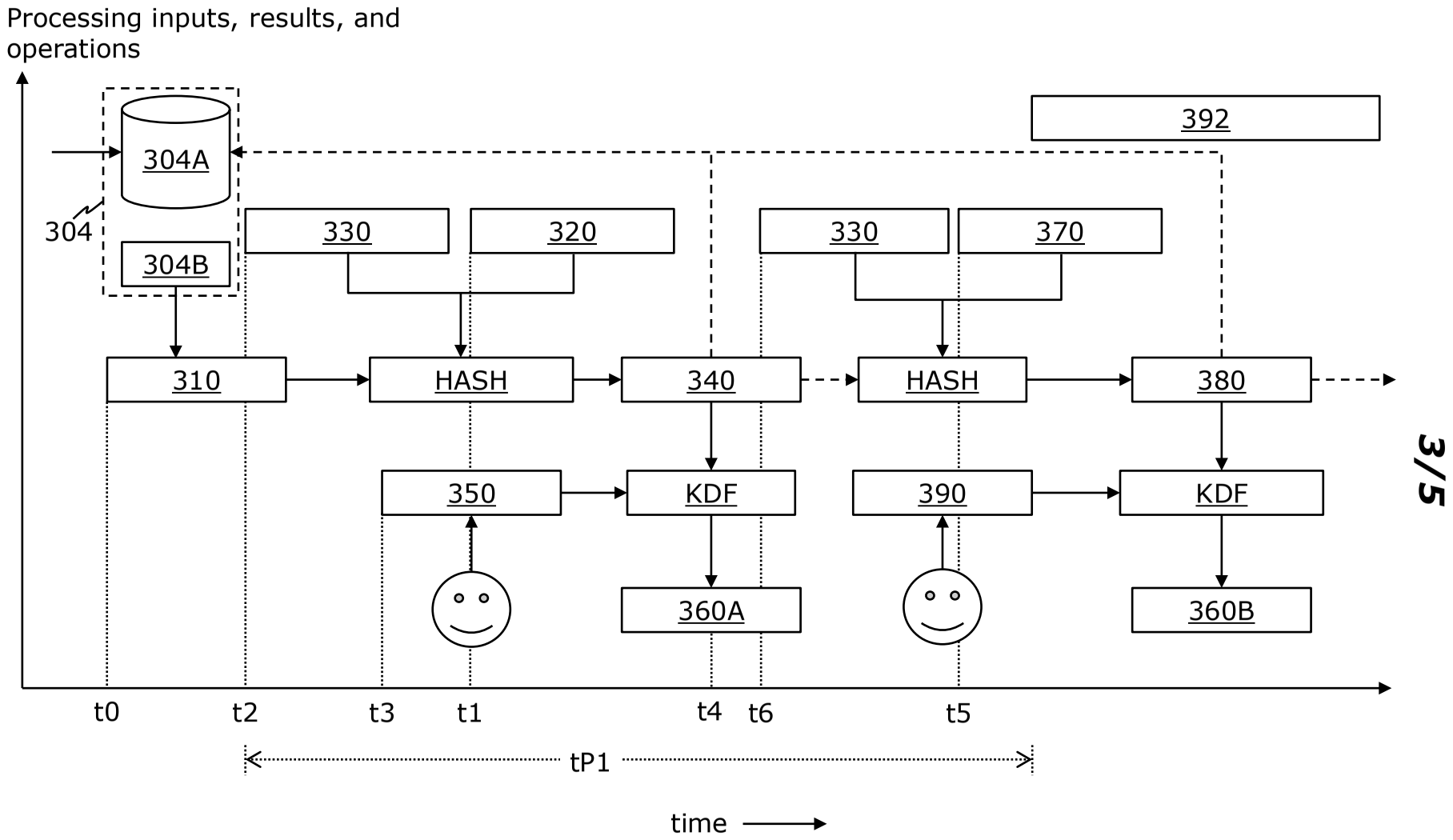

Processing inputs, results, and operations
304A
304
304B
392
330
320
330
370
310
HASH
340
HASH
380
350
KDF
390
KDF
360A
360B
t0
t2
t3
t1
t4
t6
t5
tP1
time

FIG. 3

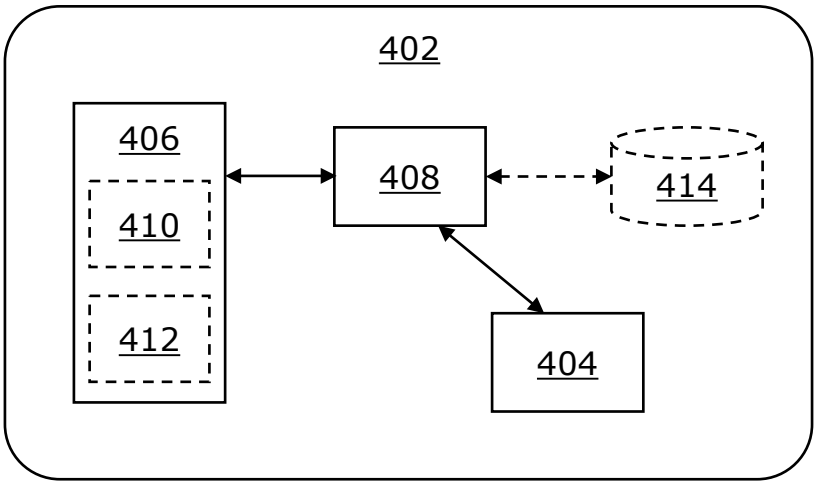


FIG. 4

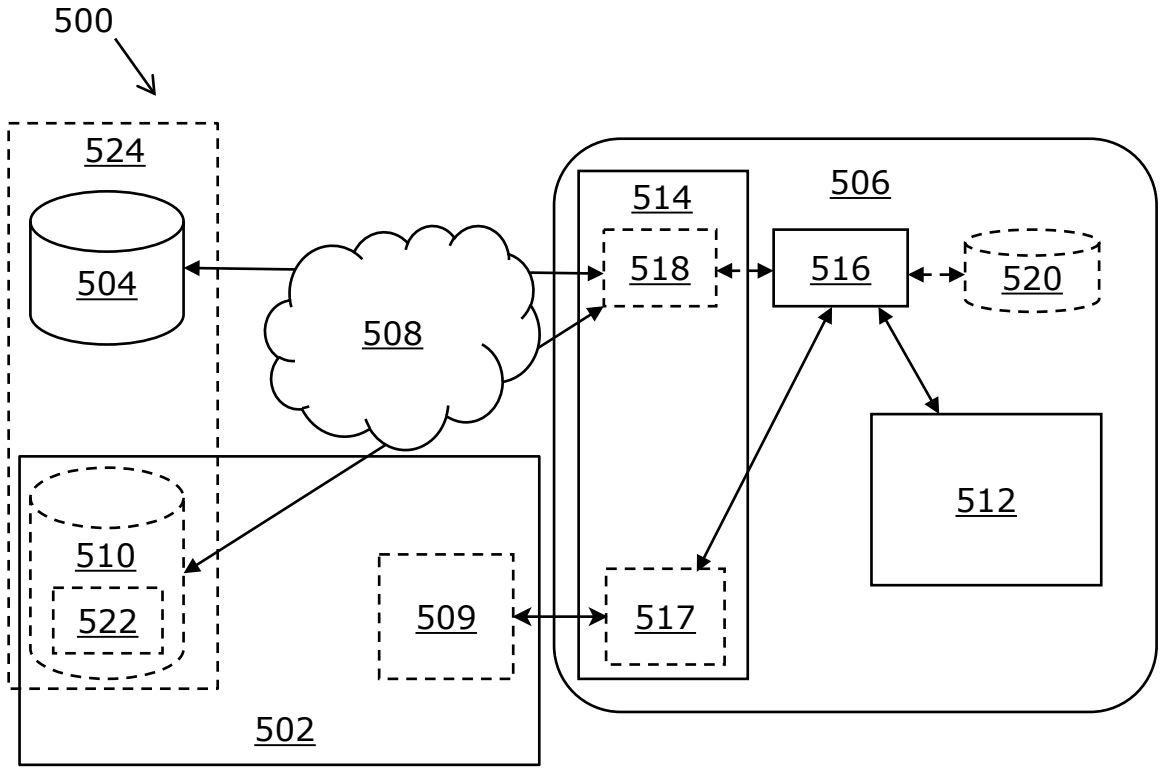


FIG. 5

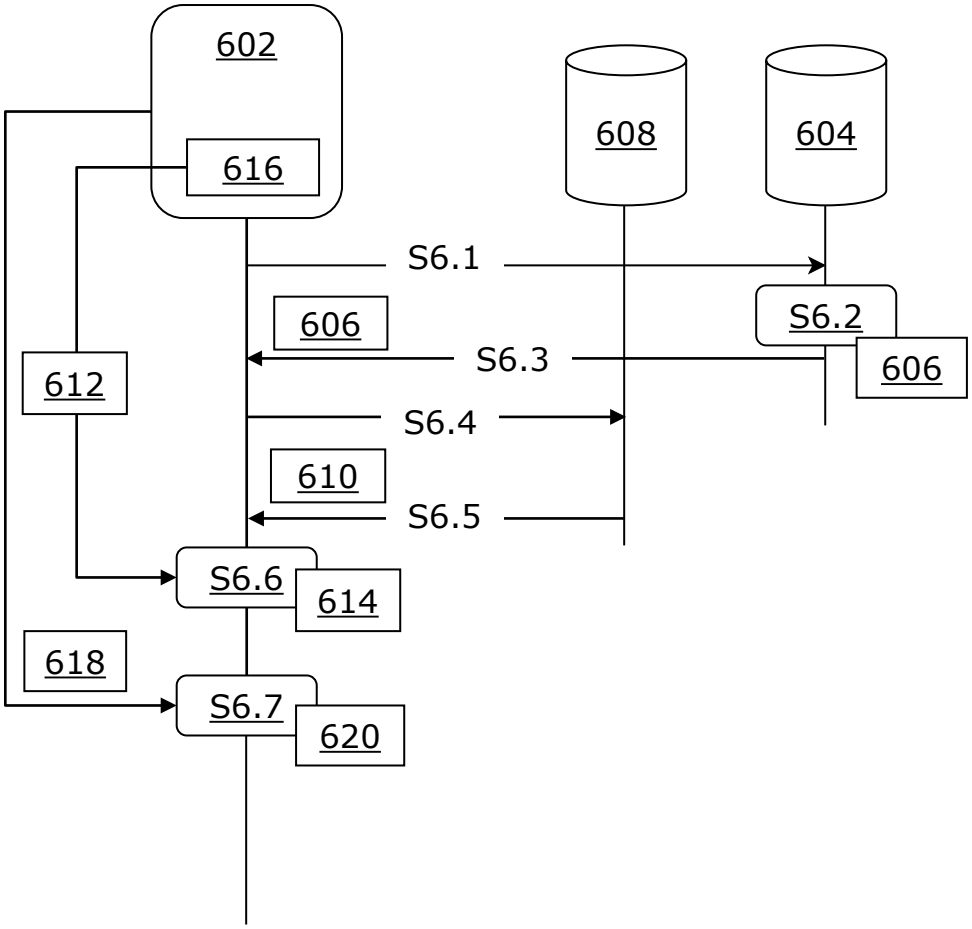


FIG. 6