



(51) International Patent Classification:
H04L 29/06 (2006.01)

(21) International Application Number:
PCT/CN2017/080574

(22) International Filing Date:
14 April 2017 (14.04.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **NOKIA TECHNOLOGIES OY** [FI/FI];
Karaportti 3, 02610 Espoo (FI).

(71) Applicant (for LC only): **NOKIA TECHNOLOGIES
(BEIJING) CO., LTD.** [CN/CN]; Unit 18, Room 1001,
Level 10, East Tower, World Financial Center, No. 1
East Third Ring Middle Road, Chaoyang District, Beijing
100020 (CN).

(72) Inventors: **DING, Wenxiu**; POX 091, No. 2 South Taibai
Road, XiDian University, Xi'an, Shaanxi 710071 (CN).
YAN, Zheng; No. 403, Building 69, No. 2 South Taibai
Road, XiDian University, Xi'an, Shaanxi 710071 (CN).

(74) Agent: **ZHONGZI LAW OFFICE**; 7F, New Era Build-
ing, 26 Pinganli, Xidajie, Xicheng District, Beijing 100034
(CN).

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

(54) Title: SECURE ENCRYPTED DATA DEDUPLICATION WITH EFFICIENT OWNERSHIP PROOF AND USER REVO-
CATION

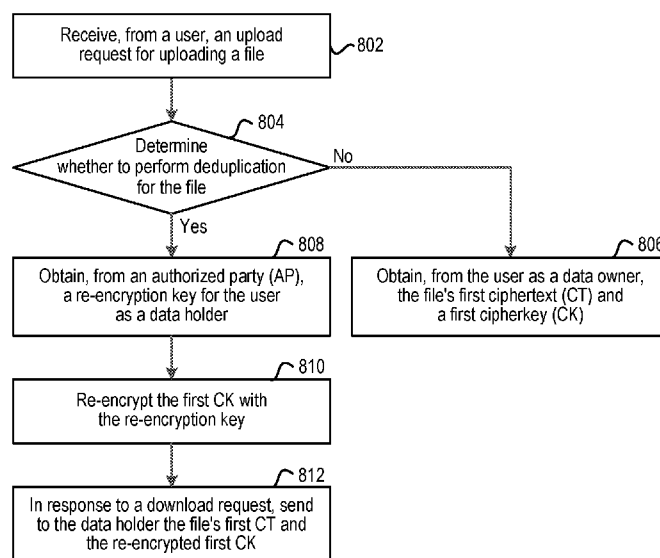


FIG. 8

(57) Abstract: Method and apparatus are disclosed for encrypted data deduplication in a system comprising a cloud service provider, an authorized party and users. According to an embodiment, a method implemented at a cloud service provider comprises: receiving, from a user, an upload request for uploading a file; determining whether to perform deduplication for the file; in response to a negative determination result, obtaining, from the user as a data owner, the file's first ciphertext and a first cipher key, the file being encrypted with a first symmetric key to generate its first ciphertext, the first symmetric key being homomorphically encrypted to generate the first cipher key; in response to a positive determination result, obtaining, from an authorized party, a re-encryption key for the user as a data holder; re-encrypting the first cipher key with the re-encryption key, such that the re-encrypted first cipher key can be decrypted with the data holder's secret key; and in response to a download request from the data holder, sending to the data holder the file's first ciphertext and the re-encrypted first cipher key.

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

Published:

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

SECURE ENCRYPTED DATA DEDUPLICATION WITH EFFICIENT OWNERSHIP PROOF AND USER REVOCATION

Field of the Invention

[0001] Embodiments of the disclosure generally relate to data processing, and, more particularly, to encrypted data deduplication in cloud service environment.

Background

[0002] Cloud computing provides seemingly unlimited resources as services to cloud users by rearranging various resources. Cloud storage as one of the most popular cloud services enables cloud users to store a tremendous amount of data in the cloud, which may exceed their own storage resources.

[0003] In order to improve the storage services and enhance the scalability, data deduplication has become an important technique in cloud storage. Data deduplication can help eliminate multiple copies of same files in the cloud and improve the utilization of cloud storage. It has proved to be effective in achieving significant cost savings. For example, up to 90-95% storage needs can be reduced for backup applications and up to 68% storage needs can be reduced for standard file systems. The savings can be fed back to cloud users in many ways, such as reducing cloud storage cost. Therefore, it would be desirable for both cloud service providers and cloud users to provide an efficient data deduplication solution.

Summary

[0004] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0005] According to one aspect of the disclosure, it is provided a method implemented at a cloud service provider (CSP), comprising: receiving, from a user, an upload request for uploading a file; determining whether to perform deduplication for

the file; in response to a negative determination result, obtaining, from the user as a data owner, the file's first ciphertext and a first cipher key (CK), wherein the file is encrypted with a first symmetric key to generate its first ciphertext, and the first symmetric key is homomorphically encrypted to generate the first CK; in response to a positive determination result, obtaining, from an authorized party (AP), a re-encryption key for the user as a data holder; re-encrypting the first CK with the re-encryption key, such that the re-encrypted first CK can be decrypted with the data holder's secret key (SK); and in response to a download request from the data holder, sending to the data holder the file's first ciphertext and the re-encrypted first CK.

[0006] In an embodiment of the disclosure, the upload request comprises a proof of ownership that is generated based on the file's hash value and the user's SK. The step of determining comprises: generating a file tag based on the proof of ownership and the user's public key (PK); and determining whether the file tag has been previously stored.

[0007] In an embodiment of the disclosure, the method further comprises: in response to the positive determination result, acquiring, from the user and another data holder, their hash chains of the file; checking whether the two hash chains are the same; in response to a positive check result, recording the user as a data holder; and in response to a negative check result, refusing the user to be a data holder.

[0008] In an embodiment of the disclosure, the step of acquiring is performed only when the user is a revoked user.

[0009] In an embodiment of the disclosure, the step of acquiring comprises: sending to the another data holder the file's first ciphertext and a re-encrypted first CK corresponding to the another data holder; and receiving a hash chain from the another data holder.

[0010] In an embodiment of the disclosure, the first symmetric key is homomorphically encrypted with the AP's PK. The re-encryption key is generated

based on the data holder's PK and the AP's SK. The proof of ownership is generated based further on the CSP's PK.

[0011] In an embodiment of the disclosure, the PK and SK pairs of the CSP, the AP and the user are generated based on bilinear mapping. The step of re-encrypting is performed based on bilinear mapping. The file tag is generated based on bilinear mapping.

[0012] In an embodiment of the disclosure, the method further comprises: encrypting the file's first ciphertext with a second symmetric key to update the file's first ciphertext to its second ciphertext; and cooperating with the AP to update the first CK to a second CK based on additive homomorphism, wherein the second CK is a ciphertext of a weighted sum of the first and second symmetric keys.

[0013] In an embodiment of the disclosure, the step of cooperating comprises: obtaining a first updating key from the AP; homomorphically encrypting the second symmetric key with the first updating key; and calculating a product between the first CK and the encrypted second symmetric key to generate the second CK.

[0014] In an embodiment of the disclosure, the method further comprises: decrypting the file's second ciphertext with the second symmetric key to generate the file's first ciphertext; encrypting the file's first ciphertext with a third symmetric key to update the file's second ciphertext to its third ciphertext; and updating the second CK to a third CK based on additive homomorphism, wherein the third CK is a ciphertext of a weighted sum of the first and third symmetric keys.

[0015] In an embodiment of the disclosure, the step of updating comprises: homomorphically encrypting, with the first updating key, a difference value between the third and second symmetric keys; and calculating a product between the second CK and the encrypted difference value to generate the third CK.

[0016] In an embodiment of the disclosure, the method further comprises: in response to a user revocation event, encrypting the file's first ciphertext with a fourth

symmetric key to update the file's first ciphertext to its fourth ciphertext; cooperating with the AP to update the first CK based on additive homomorphism, wherein the updated first CK is a ciphertext of a weighted sum of the first symmetric key and a third random number; and cooperating with the AP to generate a fourth CK from the updated first CK based on additive homomorphism, wherein the fourth CK is a ciphertext of a weighted sum of the first and fourth symmetric keys.

[0017] In an embodiment of the disclosure, cooperating with the AP to update the first CK comprises: obtaining a first updating key from the AP; homomorphically encrypting the third random number with the first updating key; and calculating a product between the first CK and the encrypted third random number to generate the updated first CK. Cooperating with the AP to generate the fourth CK comprises: sending the updated first CK to the AP, wherein the AP is able to decrypt the updated first CK to its plaintext, and homomorphically encrypt the plaintext to a further updated first CK; receiving the further updated first CK and a second updating key from the AP; homomorphically encrypting, with the second updating key, a difference value between the fourth symmetric key and the third random number; and calculating a product between the further updated first CK and the encrypted difference value to generate the fourth CK.

[0018] In an embodiment of the disclosure, the method further comprises: in response to a user revocation event, decrypting the file's second ciphertext with the second symmetric key to generate the file's first ciphertext; encrypting the file's first ciphertext with a fifth symmetric key to update the file's second ciphertext to its fifth ciphertext; updating the second CK based on additive homomorphism, wherein the updated second CK is a ciphertext of a weighted sum of the first and second symmetric keys and a fourth random number; and cooperating with the AP to generate a fifth CK from the updated second CK based on additive homomorphism, wherein the fifth CK is a ciphertext of a weighted sum of the first and fifth symmetric keys.

[0019] In an embodiment of the disclosure, updating the second CK comprises: homomorphically encrypting the fourth random number with the first updating key; and calculating a product between the second CK and the encrypted fourth random number to generate the updated second CK. Cooperating with the AP to generate the fifth CK comprises: sending the updated second CK to the AP, wherein the AP is able to decrypt the updated second CK to its plaintext, and homomorphically encrypt the plaintext to a further updated second CK; receiving the further updated second CK and a second updating key from the AP; homomorphically encrypting, with the second updating key, a difference value between the fifth symmetric key and a weighted sum of the second symmetric key and the fourth random number; and calculating a product between the further updated second CK and the encrypted difference value to generate the fifth CK.

[0020] According to another aspect of the disclosure, it is provided a method implemented an authorized party (AP), comprising: receiving, from a cloud service provider (CSP), a request for generating a re-encryption key for a user as a data holder; generating the re-encryption key for the data holder; and sending the re-encryption key to the CSP. The CSP is able to re-encrypt a first cipher key (CK) with the re-encryption key, such that the re-encrypted first CK can be decrypted with the data holder's secret key (SK) to a first symmetric key that can be used to decrypt a file's first ciphertext.

[0021] In an embodiment of the disclosure, the re-encryption key is generated based on the data holder's public key (PK) and the AP's SK.

[0022] In an embodiment of the disclosure, the method further comprises: receiving, from the CSP, a request for generating an updating key; generating a first updating key; and sending the first updating key to the CSP. The CSP is able to use the first updating key to update a CK of the file.

[0023] In an embodiment of the disclosure, the method further comprises: receiving the updated CK from the CSP; decrypting the updated CK to its plaintext;

homomorphically encrypting the plaintext to a further updated CK; generating a second updating key; sending the further updated CK and the second updating key to the CSP. The CSP is able to use the further updated CK and the second updating key to further update the CK of the file.

[0024] In an embodiment of the disclosure, the first and second updating keys are generated based on homomorphic encryption and bilinear mapping.

[0025] According to another aspect of the disclosure, it is provided a method implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider (CSP), and a notification indicating the user to be a data owner is received from the CSP. The method comprises: encrypting the file with a symmetric key to generate its ciphertext; homomorphically encrypting the symmetric key to generate a cipher key (CK); and sending the ciphertext and the CK to the CSP.

[0026] In an embodiment of the disclosure, the method further comprises: generating a proof of ownership based on the file's hash value and the user's secret key (SK). The upload request comprises the proof of ownership.

[0027] In an embodiment of the disclosure, the method further comprises: in response to a request from the CSP for verifying the user's ownership of the file, generating a hash chain from the file; and sending the hash chain to the CSP.

[0028] In an embodiment of the disclosure, the symmetric key is homomorphically encrypted with an authorized party (AP)'s public key (PK). The proof of ownership is generated based further on the CSP's PK.

[0029] In an embodiment of the disclosure, it is provided a method implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider (CSP), and a notification indicating the user to be a data holder is received from the CSP. The method comprises: sending to the CSP a download request for downloading the file; receiving, from the CSP, the file's ciphertext and a

re-encrypted cipher key (CK); decrypting, with the user's secret key (SK), the re-encrypted CK to a symmetric key; and decrypting, with the symmetric key, the file's ciphertext to its plaintext.

[0030] In an embodiment of the disclosure, the method further comprises: generating a proof of ownership based on the file's hash value and the user's SK. The upload request comprises the proof of ownership.

[0031] In an embodiment of the disclosure, the method further comprises: in response to a request from the CSP for verifying the user's ownership of the file, generating a hash chain from the file; and sending the hash chain to the CSP.

[0032] In an embodiment of the disclosure, the method further comprises: receiving, from the CSP, a request for verifying another user's ownership, wherein the request comprises the file's ciphertext and a re-encrypted CK; decrypting, with the user's SK, the re-encrypted CK to a symmetric key; decrypting, with the symmetric key, the file's ciphertext to its plaintext; generating a hash chain from the file; and sending the hash chain to the CSP.

[0033] According to another aspect of the disclosure, it is provided an apparatus comprising: at least one processor; and at least one memory including computer-executable code. The at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform all steps of any one of the above described methods.

[0034] According to another aspect of the disclosure, it is provided an apparatus comprising means configured to perform all steps of any one of the above described methods.

[0035] According to another aspect of the disclosure, it is provided a computer program product comprising at least one non-transitory computer-readable storage medium having computer-executable program code stored therein. The computer-

executable code is configured to, when being executed, cause an apparatus to operate according to any one of the above described methods.

[0036] These and other objects, features and advantages of the disclosure will become apparent from the following detailed description of illustrative embodiments thereof, which are to be read in connection with the accompanying drawings.

Brief Description of the Drawings

[0037] FIG. 1 depicts a flowchart of an overall system process for ownership check according to the first scheme of the disclosure;

[0038] FIG. 2 depicts a flowchart of an overall system process for data deduplication management according to the second scheme of the disclosure;

[0039] FIG. 3 depicts a flowchart of an overall system process for ownership check according to the third scheme of the disclosure;

[0040] FIG. 4 is a schematic diagram showing a hash chain according to an embodiment of the disclosure.

[0041] FIG. 5 depicts a flowchart of an overall system process for encrypted data updating according to the fourth scheme of the disclosure;

[0042] FIG. 6 depicts a flowchart of an overall system process for user revocation management according to the fifth scheme of the disclosure;

[0043] FIG. 7 depicts a flowchart of an overall system process for user revocation management according to the sixth scheme of the disclosure;

[0044] FIG. 8 shows a flowchart of a process implemented at a cloud service provider (CSP) according to an embodiment of the disclosure;

[0045] FIG. 9 shows a flowchart of a process implemented at a CSP according to another embodiment of the disclosure;

[0046] FIGs. 10A-10B show flowcharts of processes implemented at a CSP according to another embodiment of the disclosure;

[0047] FIGs. 11A-11B show flowcharts of processes implemented at a CSP according to another embodiment of the disclosure;

[0048] FIG. 12 shows a flowchart of a process implemented at an authorized party (AP) according to an embodiment of the disclosure;

[0049] FIGs. 13A-13B show flowcharts of processes implemented at an AP according to another embodiment of the disclosure;

[0050] FIG. 14 shows a flowchart of a process implemented at a user according to an embodiment of the disclosure;

[0051] FIG. 15 shows a flowchart of a process implemented at a user according to another embodiment of the disclosure;

[0052] FIGs. 16A-16B show flowcharts of processes implemented at a user according to another embodiment of the disclosure;

[0053] FIG. 17 shows an exemplary system into which at least one embodiment of the disclosure is applicable; and

[0054] FIG. 18 is a simplified block diagram showing an apparatus that is suitable for use in practicing some embodiments of the disclosure.

Detailed Description

[0055] For the purpose of explanation, details are set forth in the following description in order to provide a thorough understanding of the embodiments disclosed. It is apparent, however, to those skilled in the art that the embodiments may be implemented without these specific details or with an equivalent arrangement.

[0056] Although deduplication brings many benefits as mentioned above, it also faces some challenges. Firstly, cloud storage and deduplication management incurs the concern about data privacy and user privacy. Cloud users would lose the full control over their out-sourced personal data, which even may be disclosed by dishonest cloud service providers. Thus, encryption as a popular method is introduced to solve this

problem. But a same file encrypted with different encryption schemes would result in different ciphertexts, which makes it difficult for duplication check. Secondly, data deletion made by users complicates the deduplication management. If some data holders delete their stored data, their access to the data should be completely prevented even when they still hold the previous key for obtaining the data. Thus, it should support user revocation. Thirdly, how to guarantee the ownership proof of data holders and previous revoked data holders is also an issue. Sometimes revoked users may re-obtain the same file from a data source and re-store it in the cloud. But current work ignores the check of its ownership in this case.

[0057] Currently, some methods about deduplication have been proposed and designed for eliminating the storage of duplicated data. In order to preserve the user privacy and data security, encryption is applied on the outsourced data. However, it complicates the data management and incurs some challenges.

[0058] Firstly, a flexible deduplication scheme with efficient access control is expected but missed in the literature. In order to support deduplication, convergent encryption is widely applied for the key sharing among data holders. But it is vulnerable to brute-force attack. Further, a data holder needs to keep one secret key for each uploaded file for subsequent access. It complicates the key management of data holders. Most encryption algorithms generate different ciphertexts over the same original file, which can only be decrypted by corresponding secret keys. Thus, it is difficult to achieve efficient duplication check and flexible access control.

[0059] Secondly, little work considered the ciphertext update and user revocation. If some data holder deletes its stored data in the cloud, it should guarantee that this data holder cannot access the data any more. Some methods need the intervention of other authorized data holders to perform data update and user revocation. But data holders should download the stored data first and then update the ciphertext, which causes high communication cost and computation cost to data holders. In addition, one authorized data holder must be online to execute the encrypted data updating operation.

[0060] Thirdly, how to efficiently check the ownership of data holders is a serious problem. Generally, in order to save the overhead, only the first uploader is required to encrypt and upload a file. It brings challenges to cloud service providers to guarantee the ownership of all data holders. In some methods, the cloud service provider challenges data holders, which requires complicated interactions between the data holders and the cloud service provider. Moreover, if some revoked user keeps essential file information, how to judge it really has the file becomes a problem when it wants to re-upload the file again.

[0061] The present disclosure proposes a data deduplication solution in cloud service environment. It may overcome at least one of the problems mentioned above, or it may not overcome any one of the problems mentioned above. Hereinafter, the solution will be described in detail in five sections entitled “the basic principle of additive homomorphic encryption and proxy re-encryption”, “the additive homomorphic re-encryption algorithm”, “the overall system process”, “the processes at individual parties” and “the system structure”.

[0062] I. The basic principle of additive homomorphic encryption and proxy re-encryption

In this section, Paillier cryptosystem is used as an exemplary example for explaining the basic principle of additive homomorphic encryption. It should be noted that any other additive homomorphic cryptosystem can also be used. The Paillier cryptosystem is based on the cyclic group $\mathbb{G}$ of quadratic residues modulo n^2 , where $n = p * q$ (p and q are two large primes). In the key generation process, the public parameters are n , g and $h = g^x \bmod n^2$, where g is an element of maximal order in $\mathbb{G}$ and $x \in_R [1, \lambda(n^2)]$ ($\lambda(*)$ is the Euler function). Since x is coprime with $\text{ord}(\mathbb{G})$ with high probability, h is also an element of maximal order in $\mathbb{G}$. It can be generated by randomly choosing a secret value $x \in [1, \text{ord}(\mathbb{G})]$.

[0063] In the encryption process, given a message $m \in \mathbb{Z}_n$, a random number r is chosen in $\mathbb{Z}_n^*$. The ciphertext is computed as:

$$[m]_h = (T, T') = \{h^r(1 + m * n), g^r\} \pmod{n^2}.$$

In the decryption process, knowing x , m can be obtained as follows:

$$m = L(T/(T')^x \pmod{n^2}), \text{ where } L(u) = (u - 1)/n.$$

[0064] A proxy re-encryption (PRE) scheme can be represented as a tuple of (possibly probabilistic) polynomial time algorithms $(KG; RG; E; R; D)$. The algorithms $(KG; E; D)$ are the standard key generation, encryption and decryption algorithms for an underlying public key encryption scheme. On input the security parameter 1^k , KG outputs a public and private key pair $(pk_A; sk_A)$ for entity A. On input pk_A and data m , E outputs a ciphertext $C_A = E(pk_A; m)$. On input sk_A and ciphertext C_A , D outputs the plain data $m = D(sk_A; C_A)$.

[0065] On input $(pk_A; sk_A; pk_B)$, the re-encryption key generation algorithm RG outputs a re-encryption key $rk_{A \rightarrow B}$ for a proxy. On input $rk_{A \rightarrow B}$ and ciphertext C_A , the re-encryption function R outputs $R(rk_{A \rightarrow B}; C_A) = E(pk_B; m) = C_B$, which can be decrypted with entity B's private key sk_B .

[0066] II. the additive homomorphic re-encryption algorithm

In this section, an additive homomorphic re-encryption (AHRE) algorithm is designed. It can be used in the first to sixth schemes of the disclosure described hereinafter. In these schemes, there are mainly four types of entities: a cloud service provider (CSP), an authorized party (AP), a data owner and data holders.

[0067] The CSP is in charge of data storage. It is curious about the user's data. But it will follow protocols strictly in order to gain commercial benefits from providing storage service to its consumers or users. The AP would never collude with the CSP and is fully trusted. The AP cannot access the raw data at the CSP. The data owner is a cloud service consumer and is the first data uploader for a certain file. The data holders are also cloud service consumers and are the subsequent data uploaders for the file. Apparently, one file's data owner may be another file's data holder.

[0068] For ease of description, the notations used in the schemes are summarized in the following Table 1.

Symbols	Description
g	The system generator that is public
n	The system parameter
m	The raw data
$[m]$	The uploaded ciphertext of m
r	The random number
$\mathcal{L}(\cdot)$	The bit length of input data
$H(\cdot)$	The hash function
$e(\cdot, \cdot)$	The bilinear pairing: $G_1 \times G_1 \rightarrow G_2$
v	The generator in G_1
$(sk_{AP}, pk_{AP}) = (y, Y = v^y)$	The key pair of AP
$(sk_{CSP}, pk_{CSP}) = (a, v^a)$	The key pair of CSP
$(sk_j, pk_j) = (u_j, v^{u_j})$	The key pair of user j
$uKey$	The updating key for ciphertext refresh
$rk_{i \rightarrow j}$	The re-encryption key from user i to user j
k	The symmetric key for data encryption
$E_k(\cdot)$	The symmetric encryption under key k
$KGen$	The key generation algorithm in AHRE
Enc	The encryption algorithm in AHRE
$RKGen$	The re-encryption key generation algorithm in AHRE
$ReEnc$	The re-encryption algorithm in AHRE
Dec	The decryption algorithm in AHRE
UKI	The updating key issue algorithm in AHRE
$CipR$	The ciphertext refresh algorithm in AHRE
AH	The additive homomorphism in AHRE
ℓ	The length of each block in hash chain
$HC(m)$	The hash chain code of file m

Table 1: System notations

[0069] The AHRE algorithm combines modular exponentiations and bilinear pairings to achieve additive homomorphism and re-encryption over encrypted data. It lays the basis to integrate the deduplication with access control. The AHRE algorithm mainly comprises the following algorithms: System Setup, Key Generation ($KGen$), Encryption (Enc), Re-Encryption Key Generation ($RKGen$), Re-Encryption ($ReEnc$), Decryption (Dec), Updating Key Issue (UKI), Ciphertext Refresh ($CipR$) and Additive Homomorphism (AH).

[0070] In the System Setup algorithm, at the first step, a security parameter k is given according to the security requirement of application scenarios. At the second step, two large primes p and q are chosen to satisfy $\mathcal{L}(p) = \mathcal{L}(q) = k$, where $\mathcal{L}(\cdot)$ denotes the bit length of input data. Due to the property of safe primes, there exist two primes p' and q' that satisfy $p = 2p' + 1$ and $q = 2q' + 1$. At the third step, a generator g with order $\lambda = 2p'q'$ is generated by computing $n = p * q$, selecting a random number $z \in \mathbb{Z}_{n^2}^*$ and computing $g = -z^{2n}$. The value λ can be used to decrypt the encrypted data, but it is concealed and protected from all involved parties.

[0071] At the fourth step, two groups G_1 and G_2 of prime order u are chosen to satisfy a bilinear mapping $e: G_1 \times G_1 \rightarrow G_2$. At the fifth step, the system chooses a generator v of G_1 and computes $Z = e(v, v) \in G_2$. At the sixth step, a cryptographic hash function H is chosen to satisfy $H: \{0,1\}^* \rightarrow Z_n$. That is, on input a binary object with a variable length, the hash function H outputs a hash value Z_n .

[0072] With the Key Generation ($KGen$) algorithm, the AP generates its key pair based on bilinear pairing (or mapping) parameters: $(sk_{AP}, pk_{AP}) = (y, Y = v^y)$. The CSP generates its key pair $(sk_{CSP}, pk_{CSP}) = (a, v^a)$. User j generates its key pair (u_j, v^{u_j}) .

[0073] The Encryption (Enc) algorithm can be used when any users upload their data to a CSP for storage and deduplication management. For user i , at the first step, two random values r_1 and r_2 are chosen. At the second step, the raw data m is encrypted with the AP's public key pk_{AP} . This can be done by computing a hash value $H(Z^{r_1})$ and encrypting m as:

$$[m] = \{C_1, C_2, C_3\} = \{(1 + m * n)g^{H(Z^{r_1}) * r_2} \bmod n^2, g^{r_2} \bmod n^2, Y^{r_1}\},$$

where Y^{r_1} is a component of the ciphertext generated with the AP's public key pk_{AP} .

[0074] With the Re-Encryption Key Generation ($RKGen$) algorithm, the AP generates a re-encryption key for a user based on the user's public key and the AP's secret key.

For example, the AP can delegate user j by publishing a re-encryption key $rk_{AP \rightarrow j} = v^{u_j/y}$.

[0075] With the Re-Encryption (*ReEnc*) algorithm, the CSP re-encrypts the ciphertext $[m]$ with the re-encryption key. This can be done by computing $C'_3 = e(Y^{r_1}, rk_{AP \rightarrow j}) = Z^{r_1 * u_j}$ and setting $C'_2 = C_2$ and $C'_1 = C_1$. The re-encrypted ciphertext $\{C'_1, C'_2, C'_3\}$ can be sent from the CSP to user j , as described later.

[0076] With the Decryption (*Dec*) algorithm, the re-encrypted ciphertext $\{C'_1, C'_2, C'_3\}$ is decrypted with user j 's secret key u_j . This can be done by computing $C''_3 = H((C'_3)^{1/u_j}) = H(Z^{r_1})$ and decrypting $\{C'_1, C'_2, C'_3\}$ to obtain the raw data as follows:

$$m = L(C_1 / (C'_2)^{C''_3} \bmod n^2), \text{ where } L(u) = (u - 1)/n.$$

[0077] The Updating Key Issue (*UKI*) algorithm can be used by the AP to generate an auxiliary parameter $uKey$ for the CSP when the CSP wants to update the ciphertext $[m]$. The updating key $uKey$ can be generated by decrypting the third component Y^{r_1} of the ciphertext with the AP's secret key sk_{AP} . For example, $uKey$ can be calculated as:

$$uKey = g^{H(Z^{r_1})} = g^{H(e(v^{1/y}, Y^{r_1}))} \bmod n^2$$

[0078] With the Ciphertext Refresh (*CipR*) algorithm, the CSP can update the ciphertext with the updating key $uKey$. For example, the ciphertext can be updated by randomly choose a value r_3 , computing $\overline{C}_1 = C_1 * uKey^{r_3} \bmod n^2$ and $\overline{C}_2 = C'_2 * g^{r_3} \bmod n^2$, and setting $\overline{C}_3 = C_3$.

[0079] With the Additive Homomorphism (*AH*) algorithm, the CSP can achieve an additive homomorphic operation over the ciphertext $[m]$. Specifically, at the first step, the CSP chooses another data m' and computes $\widetilde{C}_1 = (1 + m' * n) * C_1 \bmod n^2$. At the second step, the CSP directly calls the Ciphertext Refresh algorithm *CipR* to update $\{\widetilde{C}_1, C_2, C_3\}$. As a result, the ciphertext of $(m' + m)$ can be obtained as follows:

$$\begin{aligned}
[m' + m] &= \{\overline{C_1}, \overline{C_2}, \overline{C_3}\} \\
&= \{(1 + m' * n) * C_1 * uKey^{r_3} \bmod n^2, C_2 * g^{r_3} \bmod n^2, Y^{r_1}\} \\
&= \{(1 + m' * n) * (1 + m * n) * g^{H(Z^{r_1}) * (r_2 + r_3)} \bmod n^2, g^{r_2 + r_3} \bmod n^2, Y^{r_1}\}.
\end{aligned}$$

[0080] Thus, except the third component Y^{r_1} , the first two components of the ciphertexts satisfy the equation $[m' + m] = [m'] * [m]$. That is, based on the additive homomorphic encryption function's characteristic $E(m_1) \times E(m_2) = E(m_1 + m_2)$, the ciphertext of $(m' + m)$ can be generated by calculating a product between the ciphertext of m' and the ciphertext of m .

[0081] III. The overall system process

FIGs. 1-3 and 5-7 depict flowcharts of overall system processes of the first to sixth schemes of the present disclosure. As shown in these figures, the system comprises a cloud service provider (CSP), an authorized party (AP), and users (U1, U2). In some of these figures, some entity such as the AP is omitted for clarity. The general description of these entities has been provided in section II and thus is omitted here.

[0082] FIG. 1 depicts a flowchart of an overall system process for ownership check according to the first scheme of the disclosure. In this scheme, the user U1 wants to upload a file m . It may be a data owner or a data holder depending on the actual situation. In the system initialization process, the AP generates system parameters v and Z by calling the System Setup algorithm. Then, the system parameters v and Z are shared with the other entities of the system. Then, by calling the Key Generation algorithm $KGen$, the CSP, the AP and the user U1 generate their key pairs respectively.

[0083] After the system initialization process has been finished, the user U1 generates a proof of ownership based on the file's hash value and its secret key at step 102. For example, the proof of ownership can be calculated as $v^{H(m)/u_1}$. Alternatively, the proof of ownership may be generated based further on the CSP's public key. For example, the proof of ownership can be calculated as $pk_{CSP}^{H(m)/u_1}$.

[0084] At step 104, the user U1 sends to the CSP an upload request for uploading the file. The upload request comprises the generated proof of ownership. In response to the upload request, the CSP generates a file tag based on the proof of ownership and the user U1's public key at step 106. This step may be performed through bilinear mapping. For example, if the proof of ownership is calculated as $v^{H(m)/u_1}$, the file tag can be calculated as $e(v^{H(m)/u_1}, pk_{u_1}) = Z^{H(m)}$. If the proof of ownership is calculated as $pk_{CSP}^{H(m)/u_1}$, the file tag can be calculated as $e(pk_{CSP}^{H(m)/u_1}, pk_{u_1}) = Z^{a*H(m)}$.

[0085] At step 108, the CSP checks whether the file tag has been previously stored. If the file tag has been previously stored, it means that the corresponding file has been uploaded before by another user. Thus, the CSP records the user U1 as a data holder without requiring it to upload the file again. On the other hand, if the file tag has not been previously stored, the CSP informs the user U1 to upload the file.

[0086] In this way, the ownership can be proved and the duplication check can be performed without any complicated interactions between the user and the CSP. The scheme can also guarantee that only one copy of the same data is kept at the CSP.

[0087] FIG. 2 depicts a flowchart of an overall system process for data deduplication management according to the second scheme of the disclosure. In this scheme, the user U1 is assumed to be a data owner of a file m , while the user U2 is assumed to be a data holder of the file m . In the system initialization process, compared with the first scheme of FIG. 1, the AP further generates system parameters n and g by calling the System Setup algorithm. Then, the system parameters n and g are shared with the other entities of the system.

[0088] After the system initialization process has been finished, the user U1 generates a proof of ownership based on the file's hash value and its secret key at step 202. Then, the user U1 sends to the CSP an upload request for uploading the file at step 204. The upload request comprises the generated proof of ownership. In response to the upload request, the CSP generates a file tag based on the proof of ownership and

the user U1's public key at step 206. Steps 202-206 are similar to steps 102-106 of the first scheme. Detailed description about them is omitted here for brevity.

[0089] At step 208, the CSP checks whether the file tag has been previously stored. Since the user U1 is assumed to be a data owner, the CSP finds that the file tag has not been previously stored. Thus, the CSP informs the user U1 to upload the file.

[0090] In response, at step 210, the user U1 encrypts the file m with a first symmetric key k_1 to generate $E_{k_1}(m)$. The user U1 also homomorphically encrypts the first symmetric key k_1 . This may be done by encrypting k_1 with the AP's public key $pk_{AP} = Y$. For example, by calling the Encryption algorithm (Enc), the ciphertext $[k_1]$ can be calculated as:

$$[k_1] = \{C_1, C_2, C_3\} = \{(1 + k_1 * n)g^{H(Z^{r_1}) * r_2} \bmod n^2, g^{r_2} \bmod n^2, Y^{r_1}\}.$$

[0091] At step 212, the user U1 uploads the ciphertexts $E_{k_1}(m)$ and $[k_1]$ to the CSP. Then, at step 214, the CSP stores the data uploaded from the user U1 and the generated file tag. That is, the data package $\{E_{k_1}(m), [k_1] = \{C_1, C_2, C_3\}, Z^{aH(m)}\}$ is kept by the CSP.

[0092] Afterwards, the user U2 wants to upload the same file. Thus, at step 216, the user U2 generates a proof of ownership based on the file's hash value and its secret key. For example, the proof of ownership can be calculated as $pk_{CSP}^{H(m)/u_2}$. At step 218, the user U2 sends to the CSP an upload request for uploading the file. The upload request comprises the generated proof of ownership. In response to the upload request, the CSP generates a file tag based on the proof of ownership and the user U2's public key at step 220. For example, the file tag can be calculated as $e(v^{u_2}, pk_{CSP}^{H(m)/u_2}) = Z^{aH(m)}$. Steps 216-220 are similar to steps 102-106 of the first scheme. Detailed description about them is omitted here for brevity.

[0093] At step 222, the CSP checks whether the file tag has been previously stored. Since the same file tag has been kept for the user U1, the CSP records the user U2 as a data holder of the file.

[0094] Afterwards, the user U2 wants to download the file from the CSP. Thus, at step 224, the user U2 sends to the CSP a download request for downloading the file. The download request may comprise information about the file (e.g., the file tag generated by the user U2) and the information about the user U2 (e.g., its public key). In response to the download request, the CSP sends to the AP a re-encryption key request for generating a re-encryption key for the user U2 at step 226.

[0095] In response to the re-encryption key request, the AP generates the re-encryption key for the user U2 at step 228. This may be done based on the user U2's public key and the AP's secret key. For example, by calling the Re-Encryption Key Generation algorithm (*RKGen*), the re-encryption key can be calculated as $rk_{AP \rightarrow u_2} = v^{u_2/y}$. The user U2's public key v^{u_2} may be extracted from the re-encryption key request from the CSP.

[0096] Then, at step 230, the AP issues the generated re-encryption key to the CSP. At step 232, the CSP re-encrypts the ciphertext $[k_1]$ with the re-encryption key. For example, by calling the Re-Encryption algorithm (*ReEnc*), the re-encrypted ciphertext can be calculated as:

$$\{C'_1, C'_2, C'_3\} = \{C_1, C_2, Z^{r_1 * u_2}\}.$$

[0097] Then, at step 234, the CSP sends $E_{k_1}(m)$ and $\{C'_1, C'_2, C'_3\}$ to the user U2. At step 236, the user U2 decrypts the received data to get the file m . Specifically, the user U2 decrypts the re-encrypted ciphertext $\{C'_1, C'_2, C'_3\}$ with its secret key. For example, by calling the Decryption algorithm (*Dec*), the first symmetric key k_1 can be obtained as follows:

$$k_1 = L(C_1 / (C'_2)^{C''_3} \bmod n^2),$$

$$\text{where } C''_3 = H((C'_3)^{1/u_2}) \text{ and } L(u) = (u - 1)/n.$$

Then, the user U2 decrypts $E_{k_1}(m)$ with k_1 to get the file m .

[0098] In this way, the above second scheme can guarantee the data security and user privacy through symmetric encryption and the AHRE algorithm. It integrates the data

deduplication and access control, which can support data sharing among data holders even when the data owner is offline. All users only need to keep their own key pairs for data access.

[0099] FIG. 3 depicts a flowchart of an overall system process for ownership check according to the third scheme of the disclosure. In this scheme, the user U1 uploaded a file to the CSP and was recorded as a data holder. Then, it deleted the file from the cloud and thus became a revoked user. Now, it uploads the file again to the CSP. Because it is possible that a revoked user only keeps the previous proof of ownership but does not have the real file, another ownership check is performed in this scheme with the help of another data holder. Suppose the user U2 is a data holder who is currently online. Thus, it can cooperate with the CSP to verify the ownership of the revoked user U1.

[00100] In the system initialization process, compared with the second scheme, the AP further chooses a hash function H by calling the System Setup algorithm. Then, the information about the hash function is shared with the other entities of the system.

[00101] At step 302, the user U1 generates a proof of ownership based on the file's hash value and its secret key. At step 304, the user U1 sends to the CSP an upload request for uploading the file. The upload request comprises the generated proof of ownership. At step 306, the CSP generates a file tag based on the proof of ownership and the user U1's public key. Steps 302-306 are similar to steps 102-106 of the first scheme. Detailed description about them is omitted here for brevity.

[00102] At step 308, the CSP checks whether the file tag has been previously stored. As a result, the CSP finds that the same file tag has been previously stored. Then, at step 310, the CSP further checks whether the user U1 is a revoked user. As a result, the CSP finds that the user U1 is a revoked user.

[00103] Then, the CSP requests the data holder U2 and the user U1 to send the file's hash chain. Specifically, at step 312a, the CSP contacts the data holder U2 to check the user U1's ownership. This can be done by sending a verification request to the

data holder U2. The verification request comprises $E_{k_1}(m)$ and $\{C'_1, C'_2, C'_3\}$. Meanwhile, at step 312b, the CSP requests the user U1 to send the file's hash chain.

[00104] In response to the verification request from step 312a, the data holder U2 decrypts the received data to get the file m at step 314. This step is similar to step 236. Then, at step 316, the data holder U2 generates a hash chain from the file. FIG. 4 schematically shows a hash chain according to an embodiment of the disclosure. As shown, by dividing the file m into multiple blocks according to the length of block ℓ , block 1 to block n can be obtained. Then, the hash value corresponding to block 1 can be calculated as $Hash_1 = H(block\ 1)$, where $H()$ is the hash function chosen in the system initialization process. The hash value corresponding to block 2 can be calculated as $Hash_2 = H(Hash_1 + block\ 2)$. Likewise, the hash value corresponding to block n can be calculated as $Hash_n = H(Hash_{n-1} + block\ n)$. Since $Hash_n$ depends on all the blocks of the file m , $Hash_n$ can be taken as the file's hash chain $HC(m)$. In this way, the communication cost can be saved. However, it is also possible that $HC(m)$ is generated as $\{Hash_1, Hash_2, \dots, Hash_n\}$ or a combination of one or more components thereof. The parameter ℓ may be extracted from the verification request from the CSP. It may be randomly chosen by the CSP to challenge a revoked user. Then, at step 318, the data holder U2 sends the generated hash chain to the CSP.

[00105] In response to the request from step 312b, the user U1 generates a hash chain from the file at step 320. This step is similar to step 316. Then, at step 322, the user U1 sends the generated hash chain to the CSP.

[00106] At step 324, the CSP checks whether the two hash chains are the same. If the two hash chains are the same, the CSP records the user U1 as a data holder again. On the other hand, if the two hash chains are different from each other, the CSP refuses the user U1 to be a data holder. In this way, the scheme can check the ownership of revoked users who want to re-upload duplicated data.

[00107] It should be noted that the present disclosure is not so limited. As another example, step 310 can be omitted. In this way, the hash chain challenge can be strictly

performed with respect to every data uploader to verify its ownership. In this way, the replay attack of the file tag can be resisted.

[00108] FIG. 5 depicts a flowchart of an overall system process for encrypted data updating according to the fourth scheme of the disclosure. In this scheme, after the data packet $\{E_{k_1}(m), [k_1] = \{C_1, C_2, C_3\}\}$ is stored at the CSP as shown in FIG. 2, the user U1 acts as the first requester to request the CSP to update the stored data. After the stored data is updated by the CSP, the user U2 further requests the CSP to update the stored data. Thus, there are two ciphertext updating processes in FIG. 5. It should be noted that in addition to the users, the CSP can also trigger the ciphertext updating process to guarantee the data security.

[00109] In the first ciphertext updating process, at step 502, the user U1 sends an updating request to the CSP. In response to the updating request, the CSP updates the file's ciphertext $E_{k_1}(m)$ at step 504. This can be done by choosing a random k_2 as a second symmetric key and encrypting $E_{k_1}(m)$ with k_2 to get $E_{k_2}(E_{k_1}(m))$, where k_2 is kept by the CSP.

[00110] At step 506, the CSP sends an updating key request to the AP. The updating key request may comprise the third component ($C_3 = Y^{r_1}$) of the key ciphertext that is encrypted with the AP's public key. In response to the updating key request, the AP generates a first updating key at step 508. This can be done by decrypting the third component Y^{r_1} with the AP's secret key ($sk_{AP} = y$). For example, by calling the Updating Key Issue algorithm (*UKI*), the first updating key can be calculated as:

$$uKey = g^{H(Z^{r_1})} = g^{H(e(v^{1/y}, Y^{r_1}))}.$$

Then, at step 510, the AP issues the first updating key to the CSP.

[00111] At step 512, the CSP updates $[k_1]$ with $uKey$ based on additive homomorphism. For example, the CSP can scale k_2 by $2^{\mathcal{L}(k)}$, where k is a security parameter generated in the system initialization process and $\mathcal{L}(\cdot)$ denotes the bit length of input data. Then, by calling the Additive Homomorphism algorithm (*AH*), the ciphertext of $(k_1 + 2^{\mathcal{L}(k)}k_2)$ can be generated. The CSP can keep the packet

$\{E_{k_2}(E_{k_1}(m)), [k_1 + 2^{\mathcal{L}(k)}k_2], k_2\}$, where k_2 is kept secret in its storage. In this way, the two symmetric keys (i.e., the inner key k_1 and the outer key k_2) can be concatenated in a secure way.

[00112] In the second ciphertext updating process, at step 514, the user U2 sends an updating request to the CSP. In response to the updating request, the CSP updates the $E_{k_2}(E_{k_1}(m))$ at step 516. This can be done by decrypting $E_{k_2}(E_{k_1}(m))$ with k_2 to get $E_{k_1}(m)$, choosing a random k_3 as a third symmetric key and encrypting $E_{k_1}(m)$ with k_3 to get $E_{k_3}(E_{k_1}(m))$, where k_3 is kept by the CSP.

[00113] At step 518, the CSP updates $[k_1 + 2^{\mathcal{L}(k)}k_2]$ with the first updating key $uKey$ based on additive homomorphism. Since $uKey$ has been obtained from the AP during the first ciphertext updating process, it is not required to request the AP to issue it again. For example, the CSP can calculate $(n - 2^{\mathcal{L}(k)}k_2 + 2^{\mathcal{L}(k)}k_3)$. Then, by calling the Additive Homomorphism algorithm (AH),

$[(k_1 + 2^{\mathcal{L}(k)}k_2) + (n - 2^{\mathcal{L}(k)}k_2 + 2^{\mathcal{L}(k)}k_3)] = [k_1 + 2^{\mathcal{L}(k)}k_3]$ can be generated. The CSP can keep the packet $\{E_{k_3}(E_{k_1}(m)), [k_1 + 2^{\mathcal{L}(k)}k_3], k_3\}$.

[00114] Afterwards, the CSP can always employ the second ciphertext updating process to update the encrypted data in the future. In this way, through the interaction between the CSP and the AP, the scheme can efficiently update the ciphertexts without data user's intervention and support. The data security can be enhanced by refreshing the encrypted stored data irregularly.

[00115] FIG. 6 depicts a flowchart of an overall system process for user revocation management according to the fifth scheme of the disclosure. This scheme is proposed in view of the fact that if use i deletes its storage in the cloud, the ciphertext update described above with reference to FIG. 5 may be not secure enough, which still cannot prevent the access of user i if it has kept some previous important information. In this scheme, after the data packet $\{E_{k_1}(m), [k_1] = \{C_1, C_2, C_3\}\}$ is stored at the CSP as shown in FIG. 2, the user U1 requests the CSP to delete the stored data and thus

becomes a revoked user. In response to this user revocation event, the CSP updates the stored data to guarantee the data security.

[00116] At step 602, the user U1 sends a delete request to the CSP. In response to this delete request, at step 604, the CSP records the user U1 as a revoked user. The CSP also updates the file's ciphertext. This can be done by choosing a random k_4 as a fourth symmetric key and encrypting $E_{k_1}(m)$ with k_4 to get $E_{k_4}(E_{k_1}(m))$, where k_4 is kept by the CSP.

[00117] At step 606, the CSP sends an updating key request to the AP. In response to the updating key request, the AP generates a first updating key $uKey$ at step 608. Then, at step 610, the AP issues the first updating key to the CSP. Steps 606-610 are similar to steps 506-510 of the fourth scheme. Detailed description about them is omitted here for brevity.

[00118] At step 612, the CSP updates $[k_1]$ with $uKey$ based on additive homomorphism. For example, the CSP can choose a random number r_3 . Then, by calling the Additive Homomorphism algorithm (AH), $[k_1 + r_3]$ can be generated, where r_3 is kept by the CSP. Then, at step 614, the CSP sends $[k_1 + r_3]$ to the AP.

[00119] At step 616, the AP decrypts $[k_1 + r_3]$ to get $(k_1 + r_3)$. The ciphertext can be decrypted with the AP's secret key. For example, the AP can calculate $C'_3 = e(Y^{r_1}, v) = Z^{r_1 * y}$ and $C''_3 = H((C'_3)^{1/y}) = H(Z^{r_1})$. Then, by calling the Decryption algorithm (Dec), $(k_1 + r_3)$ can be obtained.

[00120] The AP further encrypts $(k_1 + r_3)$ with newly chosen random numbers by calling the Encryption algorithm (Enc). As a result, another ciphertext $[k_1 + r_3]'$ can be calculated as follows:

$$[k_1 + r_3]' = \{C_1, C_2, C_3\} = \{(1 + (k_1 + r_3) * n)g^{H(Z^{r_1}) * r'_2 \bmod n^2}, g^{r'_2 \bmod n^2}, Y^{r'_1}\},$$

where the superscript “'” in $[k_1 + r_3]'$ means that this ciphertext is generated with newly chosen random numbers.

[00121] At step 618, the AP generates a second updating key. This can be done by decrypting the third component $Y^{r'_1}$ with the AP's secret key $sk_{AP} = y$. For example, by calling the Updating Key Issue algorithm (*UKI*), the second updating key can be calculated as:

$$uKey' = g^{H(e(v^{1/y}, Y^{r'_1}))} = g^{H(Z^{r'_1})}.$$

Then, at step 620, the AP sends $[k_1 + r_3]'$ and $uKey'$ to the CSP.

[00122] At step 622, the CSP updates $[k_1 + r_3]'$ with $uKey'$ based on additive homomorphism. For example, the CSP can calculate $(n - r_3 + 2^{\mathcal{L}(k)}k_4)$. Then, by calling the Additive Homomorphism algorithm (*AH*),

$[(k_1 + r_3) + (n - r_3 + 2^{\mathcal{L}(k)}k_4)]' = [k_1 + 2^{\mathcal{L}(k)}k_4]'$ can be generated. The CSP can keep the packet $\{E_{k_4}(E_{k_1}(m)), [k_1 + 2^{\mathcal{L}(k)}k_4]', k_4\}$.

[00123] In this way, through the interaction between the CSP and the AP, the scheme can efficiently block the access of users who have deleted their data stored at the CSP. Since the scheme updates the ciphertext of keys with the *AH* algorithm, it does not incur more computation cost for data holders to get the symmetric keys. Compared with previous work, it does not need data holders to download the encrypted file for fulfilling ciphertext update. This greatly saves the communication cost and computation overhead, especially from the viewpoint of cloud data holders and owners.

[00124] FIG. 7 depicts a flowchart of an overall system process for user revocation management according to the sixth scheme of the disclosure. This scheme is proposed in view of the reasons similar to the fifth scheme. In this scheme, after the packet $\{E_{k_2}(E_{k_1}(m)), [k_1 + 2^{\mathcal{L}(k)}k_2], k_2\}$ is stored at the CSP as shown in FIG. 5 (that is, after the first ciphertext updating process has been finished), the user U1 requests the CSP to delete the stored data and thus becomes a revoked user. In response to this user revocation event, the CSP updates the stored data to guarantee the data security.

[00125] At step 702, the user U1 sends a delete request to the CSP. In response to this delete request, at step 704, the CSP records the user U1 as a revoked user. The CSP also updates the file's ciphertext. This can be done by decrypting $E_{k_2}(E_{k_1}(m))$ with k_2 to get $E_{k_1}(m)$, choosing a random k_5 as a fifth symmetric key and encrypting $E_{k_1}(m)$ with k_5 to get $E_{k_5}(E_{k_1}(m))$, where k_5 is kept by the CSP.

[00126] At step 712, the CSP updates $[k_1 + 2^{\mathcal{L}(k)}k_2]$ with the first updating key $uKey$ based on additive homomorphism. For example, the CSP can choose a random number r_4 . Then, by calling the Additive Homomorphism algorithm (AH),

$[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]$ can be generated, where r_4 is kept by the CSP. Then, at step 714, the CSP sends $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]$ to the AP.

[00127] At step 716, the AP decrypts $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]$ to get $(k_1 + 2^{\mathcal{L}(k)}k_2 + r_4)$. This can be done with the AP's secret key. For example, the AP can calculate $C'_3 = e(Y^{r_1}, v) = Z^{r_1 * y}$ and $C''_3 = H((C'_3)^{1/y}) = H(Z^{r_1})$. Then, by calling the Decryption algorithm (Dec), $(k_1 + 2^{\mathcal{L}(k)}k_2 + r_4)$ can be obtained.

[00128] The AP further encrypts $(k_1 + 2^{\mathcal{L}(k)}k_2 + r_4)$ with newly chosen random numbers by calling the Encryption algorithm (Enc). As a result, another ciphertext $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]'$ can be calculated as follows:

$$[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]' = \{C_1, C_2, C_3\} = \left\{ \left(1 + (k_1 + 2^{\mathcal{L}(k)}k_2 + r_4) * n \right) g^{H(Z^{r'_1}) * r'_2 \bmod n^2} \bmod n^2, g^{r'_2 \bmod n^2}, Y^{r'_1} \right\},$$

where the superscript “'” in $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]'$ means that this ciphertext is generated with newly chosen random numbers.

[00129] At step 718, the AP generates a second updating key $uKey'$. Then, at step 720, the AP sends $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]'$ and $uKey'$ to the CSP. Steps 718-720 are similar to steps 618-620 of the fifth scheme. Detailed description about them is omitted here for brevity.

[00130] At step 722, the CSP updates $[k_1 + 2^{\mathcal{L}(k)}k_2 + r_4]'$ with $uKey'$ based on additive homomorphism. For example, the CSP can calculate $(n - r_4 - 2^{\mathcal{L}(k)}k_2 + 2^{\mathcal{L}(k)}k_5)$. Then, by calling the Additive Homomorphism algorithm (AH),

$[(k_1 + 2^{\mathcal{L}(k)}k_2 + r_4) + (n - r_4 - 2^{\mathcal{L}(k)}k_2 + 2^{\mathcal{L}(k)}k_5)]' = [k_1 + 2^{\mathcal{L}(k)}k_5]'$ can be generated. The CSP can keep the packet $\{E_{k_5}(E_{k_1}(m)), [k_1 + 2^{\mathcal{L}(k)}k_5]', k_5\}$.

[00131] In this way, through the interaction between the CSP and the AP, the scheme can efficiently block the access of users who have deleted their data stored at the CSP. Since the scheme updates the ciphertext of keys with the AH algorithm, it does not incur more computation cost for data holders to get the symmetric keys. Compared with previous work, it does not need data holders to download the encrypted file for fulfilling ciphertext update. This greatly saves the communication cost and computation overhead, especially from the viewpoint of cloud data holders and owners.

[00132] IV. The processes at individual parties

FIG. 8 shows a flowchart of a process implemented at a CSP according to an embodiment of the disclosure. This embodiment corresponds to the second scheme shown in FIG. 2. At step 802, an upload request for uploading a file is received from a user. The upload request may comprise suitable information about the file (e.g., its name, hash value and so on) to support deduplication. Then, at step 804, it is determined whether to perform deduplication for the file.

[00133] If it is determined at step 804 that deduplication is not required for the file, the file's first ciphertext and a first cipher key (CK) is obtained from the user as a data owner at step 806. The file's first ciphertext is generated by encrypting the file with a first symmetric key. The first cipher key is generated by homomorphically encrypting the first symmetric key. Step 806 corresponds to steps 210 and 212 of FIG. 2.

[00134] On the other hand, if it is determined at step 804 that deduplication is required for the file, a re-encryption key for the user as a data holder is obtained from an AP at step 808. This step corresponds to steps 226-230 of FIG. 2. Then, at step 810,

the first CK is re-encrypted with the re-encryption key, such that the re-encrypted first CK can be decrypted with the data holder's secret key (SK). This step corresponds to step 232 of FIG. 2. Then, in response to a download request from the data holder, the file's first ciphertext and the re-encrypted first CK are sent to the data holder at step 812. This step corresponds to step 234 of FIG. 2.

[00135] FIG. 9 shows a flowchart of a process implemented at a CSP according to another embodiment of the disclosure. This embodiment corresponds to the first to third schemes shown in FIGs. 1-4. At step 902, an upload request for uploading a file is received from a user. The upload request comprises a proof of ownership that is generated based on the file's hash value and the user's SK. This step corresponds to steps 202 and 204 of FIG. 2. At step 903, a file tag is generated based on the proof of ownership and the user's public key (PK). This step corresponds to step 206 of FIG. 2. At step 904, it is determined whether the file tag has been previously stored. This step corresponds to step 208 of FIG. 2.

[00136] If it is determined at step 904 that the file tag has not been previously stored, the file's first ciphertext and a first CK is obtained from the user as a data owner at step 906. This step is similar to step 806 of FIG. 8. On the other hand, if it is determined at step 904 that the file tag has been previously stored, hash chains of the file are acquired from the user and another data holder at step 914. This step corresponds to steps 312-322 of FIG. 3.

[00137] Then, at step 916, it is checked whether the two hash chains are the same. This step corresponds to step 324 of FIG. 3. If the two hash chains are different from each other, the user is refused to be a data holder at step 918. On the other hand, if the two hash chains are the same, the user is recorded as a data holder at step 920. Then, the process proceeds to step 808 and follows similar steps as shown in FIG. 8.

[00138] FIGs. 10A-10B show flowcharts of processes implemented at a CSP according to another embodiment of the disclosure. This embodiment corresponds to the fourth scheme shown in FIG. 5. FIG. 10A corresponds to the first ciphertext

updating process, while FIG. 10B corresponds to the second ciphertext updating process.

[00139] At step 1002, the file's first ciphertext is encrypted with a second symmetric key to update the file's first ciphertext to its second ciphertext. This step corresponds to step 504 of FIG. 5. At step 1004, the CSP cooperates with the AP to update the first CK to a second CK based on additive homomorphism, wherein the second CK is a ciphertext of a weighted sum of the first and second symmetric keys. This step corresponds to steps 506-512 of FIG. 5.

[00140] At step 1006, the file's second ciphertext is decrypted with the second symmetric key to generate the file's first ciphertext. Then, at step 1008, the file's first ciphertext is encrypted with a third symmetric key to update the file's second ciphertext to its third ciphertext. Steps 1006 and 1008 correspond to step 516 of FIG. 5. Then, at step 1010, the second CK is updated to a third CK based on additive homomorphism, wherein the third CK is a ciphertext of a weighted sum of the first and third symmetric keys. This step corresponds to step 518 of FIG. 5.

[00141] FIGs. 11A-11B show flowcharts of processes implemented at a CSP according to another embodiment of the disclosure. FIG. 11A corresponds to the fifth scheme shown in FIG. 6. At step 1102, in response to a user revocation event, the file's first ciphertext is encrypted with a fourth symmetric key to update the file's first ciphertext to its fourth ciphertext. This step corresponds to step 604 of FIG. 6. At step 1104, the CSP cooperates with the AP to update the first CK based on additive homomorphism, wherein the updated first CK is a ciphertext of a weighted sum of the first symmetric key and a third random number. This step corresponds to steps 606-612 of FIG. 6. At step 1106, the CSP cooperates with the AP to generate a fourth CK from the updated first CK based on additive homomorphism, wherein the fourth CK is a ciphertext of a weighted sum of the first and fourth symmetric keys. This step corresponds to steps 614-622 of FIG. 6.

[00142] FIG. 11B corresponds to the sixth scheme shown in FIG. 7. At step 1108, in response to a user revocation event, the file's second ciphertext is decrypted with the

second symmetric key to generate the file's first ciphertext. At step 1110, the file's first ciphertext is encrypted with a fifth symmetric key to update the file's second ciphertext to its fifth ciphertext. Steps 1108 and 1110 correspond to step 704 of FIG. 7. Then, at step 1112, the second CK is updated based on additive homomorphism, wherein the updated second CK is a ciphertext of a weighted sum of the first and second symmetric keys and a fourth random number. This step corresponds to step 712 of FIG. 7. At step 1114, the CSP cooperates with the AP to generate a fifth CK from the updated second CK based on additive homomorphism, wherein the fifth CK is a ciphertext of a weighted sum of the first and fifth symmetric keys. This step corresponds to steps 714-722 of FIG. 7.

[00143] FIG. 12 shows a flowchart of a process implemented at an AP according to an embodiment of the disclosure. This embodiment corresponds to the second scheme shown in FIG. 2. At step 1202, a request for generating a re-encryption key for a user as a data holder is received from a CSP. This step corresponds to step 226 of FIG. 2. At step 1204, the re-encryption key is generated for the data holder. This step corresponds to step 228 of FIG. 2. At step 1206, the re-encryption key is sent to the CSP. This step corresponds to step 230 of FIG. 2.

[00144] FIGs. 13A-13B show flowcharts of processes implemented at an AP according to another embodiment of the disclosure. FIG. 13A corresponds to the fourth and fifth schemes shown in FIGs. 5-6. At step 1302, a request for generating an updating key is received from the CSP. This step corresponds to step 506 of FIG. 5 and step 606 of FIG. 6. At step 1304, a first updating key is generated. This step corresponds to step 508 of FIG. 5 and step 608 of FIG. 6. Then, at step 1306, the first updating key is sent to the CSP. This step corresponds to step 510 of FIG. 5 and step 610 of FIG. 6.

[00145] FIG. 13B corresponds to the fifth and sixth schemes shown in FIGs. 6 and 7. After step 1306 is performed, the updated CK is received from the CSP at step 1308. This step corresponds to step 614 of FIG. 6 and step 714 of FIG. 7. At step 1310, the updated CK is decrypted to its plaintext. Then, at step 1312, the plaintext is

homomorphically encrypted to a further updated CK. Steps 1310 and 1312 correspond to step 616 of FIG. 6 and step 716 of FIG. 7. At step 1314, a second updating key is generated. This step corresponds to step 618 of FIG. 6 and step 718 of FIG. 7. At step 1316, the further updated CK and the second updating key are sent to the CSP. This step corresponds to step 620 of FIG. 6 and step 720 of FIG. 7.

[00146] FIG. 14 shows a flowchart of a process implemented at a user according to an embodiment of the disclosure. This embodiment corresponds to the second scheme shown in FIG. 2. It corresponds to a case where an upload request for uploading a file is sent from the user to a CSP, and a notification indicating the user to be a data owner is received from the CSP. At step 1402, the file is encrypted with a symmetric key to generate its ciphertext. Then, at step 1404, the symmetric key is homomorphically encrypted to generate a CK. Steps 1402 and 1404 correspond to step 210 of FIG. 2. Then, at step 1406, the ciphertext and the CK are sent to the CSP. This step corresponds to step 212 of FIG. 2.

[00147] FIG. 15 shows a flowchart of a process implemented at a user according to another embodiment of the disclosure. This embodiment corresponds to the second scheme shown in FIG. 2. It corresponds to a case where an upload request for uploading a file is sent from the user to a CSP, and a notification indicating the user to be a data holder is received from the CSP. At step 1502, a download request for downloading the file is sent to the CSP. This step corresponds to step 224 of FIG. 2. At step 1504, the file's ciphertext and a re-encrypted CK are received from the CSP. This step corresponds to step 234 of FIG. 2. Then, at step 1506, the re-encrypted CK is decrypted to a symmetric key with the user's SK. At step 1508, the file's ciphertext is decrypted to its plaintext with the symmetric key. Steps 1506 and 1508 correspond to step 236 of FIG. 2.

[00148] FIGs. 16A-16B show flowcharts of processes implemented at a user according to another embodiment of the disclosure. This embodiment corresponds to the third scheme shown in FIG. 3. FIG. 16A corresponds to the user U1, while FIG. 16B corresponds to the user U2. At step 1602, in response to a request from the CSP

for verifying the user's ownership of the file, a hash chain is generated from the file. This step corresponds to step 320 of FIG. 3. At step 1604, the hash chain is sent to the CSP. This step corresponds to step 322 of FIG. 3.

[00149] At step 1606, a request for verifying another user's ownership is received from the CSP, wherein the request comprises the file's ciphertext and a re-encrypted CK. This step corresponds to step 312a of FIG. 3. At step 1608, the re-encrypted CK is decrypted to a symmetric key with the user's SK. Then, at step 1610, the file's ciphertext is decrypted to its plaintext with the symmetric key. Steps 1608 and 1610 correspond to step 314 of FIG. 3. At step 1612, a hash chain is generated from the file. This step corresponds to step 316 of FIG. 3. Then, at step 1614, the hash chain is sent to the CSP. This step corresponds to step 318 of FIG. 3.

[00150] V. The system structure

FIG. 17 shows an exemplary system into which at least one embodiment of the disclosure is applicable. As shown in FIG. 17, the system may comprise a CSP 1702, an AP 1704, a data owner 1706 and data holders 1708. The CSP 1702 may be implemented as at least one cloud server configured to perform the steps shown in FIGs. 8-10 and 11A-11B. The AP 1704 may be implemented as a server configured to perform the steps shown in FIGs. 12 and 13A-13B.

[00151] The data owner 1706 and data holders 1708 may be implemented as mobile phone, tablet computer, personal digital assistant (PDA), desktop computer, laptop computer, and so on. The data owner 1706 may be configured to perform the steps shown in FIGs. 14 and 16A-16B. The data holder 1708 may be configured to perform the steps shown in FIGs. 15 and 16A-16B.

[00152] The communication network by which the CSP, the AP, the data owner and the data holders communicate with each other may include wired and/or wireless networks. These networks may include, but not limited to, a local area network (LAN), a metropolitan area network (MAN), a wide area network (WAN), a public data network (for example, the Internet), a self-organized mobile network, or any other suitable packet-switched network, such as a commercially owned, proprietary packet-

switched network, for example, a proprietary cable or fiber-optic network. The wireless network may be, for example, a cellular network and may employ various technologies including enhanced data rates for global evolution (EDGE), general packet radio service (GPRS), global system for mobile communications (GSM), Internet protocol multimedia subsystem (IMS), universal mobile telecommunications system (UMTS), etc., as well as any other suitable wireless medium, for example, worldwide interoperability for microwave access (WiMAX), wireless local area network (WLAN), Long Term Evolution (LTE) networks, code division multiple access (CDMA), wideband code division multiple access (WCDMA), wireless fidelity (WiFi), satellite, mobile ad-hoc network (MANET), and so on.

[00153] FIG. 18 is a simplified block diagram showing an apparatus that is suitable for use in practicing some exemplary embodiments of the present disclosure. For example, any one of the CSP, the AP, the data owner and the data holders may be implemented through the apparatus 1800. As shown, the apparatus 1800 may include a data processor 1810, a memory 1820 that stores a program 1830, and a communication interface 1840 for communicating data with other external devices through wired and/or wireless communication.

[00154] The program 1830 is assumed to include program instructions that, when executed by the data processor 1810, enable the apparatus 1800 to operate in accordance with the embodiments of this disclosure, as discussed above. That is, the embodiments of this disclosure may be implemented at least in part by computer software executable by the data processor 1810, or by hardware, or by a combination of software and hardware.

[00155] The memory 1820 may be of any type suitable to the local technical environment and may be implemented using any suitable data storage technology, such as semiconductor based memory devices, flash memory, magnetic memory devices and systems, optical memory devices and systems, fixed memory and removable memory. The data processor 1810 may be of any type suitable to the local technical environment, and may include one or more of general purpose computers,

special purpose computers, microprocessors, digital signal processors (DSPs) and processors based on multi-core processor architectures, as non-limiting examples.

[00156] In general, the various exemplary embodiments may be implemented in hardware or special purpose circuits, software, logic or any combination thereof. For example, some aspects may be implemented in hardware, while other aspects may be implemented in firmware or software which may be executed by a controller, microprocessor or other computing device, although the disclosure is not limited thereto. While various aspects of the exemplary embodiments of this disclosure may be illustrated and described as block diagrams, flow charts, or using some other pictorial denotation, it is well understood that these blocks, apparatus, systems, techniques or methods described herein may be implemented in, as non-limiting examples, hardware, software, firmware, special purpose circuits or logic, general purpose hardware or controller or other computing devices, or some combination thereof.

[00157] As such, it should be appreciated that at least some aspects of the exemplary embodiments of the disclosure may be practiced in various components such as integrated circuit chips and modules. It should thus be appreciated that the exemplary embodiments of this disclosure may be realized in an apparatus that is embodied as an integrated circuit, where the integrated circuit may comprise circuitry (as well as possibly firmware) for embodying at least one or more of a data processor, a digital signal processor, baseband circuitry and radio frequency circuitry that are configurable so as to operate in accordance with the exemplary embodiments of this disclosure.

[00158] It should be appreciated that at least some aspects of the exemplary embodiments of the disclosure may be embodied in computer-executable instructions, such as in one or more program modules, executed by one or more computers or other devices. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types when executed by a processor in a computer or other device. The computer

executable instructions may be stored on a computer readable medium such as a hard disk, optical disk, removable storage media, solid state memory, RAM, etc. As will be appreciated by one of skill in the art, the function of the program modules may be combined or distributed as desired in various embodiments. In addition, the function may be embodied in whole or in part in firmware or hardware equivalents such as integrated circuits, field programmable gate arrays (FPGA), and the like.

[00159] The present disclosure includes any novel feature or combination of features disclosed herein either explicitly or any generalization thereof. Various modifications and adaptations to the foregoing exemplary embodiments of this disclosure may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings. However, any and all modifications will still fall within the scope of the non-Limiting and exemplary embodiments of this disclosure.

Claims

What is claimed is:

1. A method implemented at a cloud service provider, the method comprising:

receiving, from a user, an upload request for uploading a file;

determining whether to perform deduplication for the file;

in response to a negative determination result, obtaining, from the user as a data owner, the file's first ciphertext and a first cipher key, the file being encrypted with a first symmetric key to generate its first ciphertext, the first symmetric key being homomorphically encrypted to generate the first cipher key;

in response to a positive determination result, obtaining, from an authorized party, a re-encryption key for the user as a data holder;

re-encrypting the first cipher key with the re-encryption key, such that the re-encrypted first cipher key can be decrypted with the data holder's secret key; and

in response to a download request from the data holder, sending to the data holder the file's first ciphertext and the re-encrypted first cipher key.

2. The method according to claim 1, wherein the upload request comprises a proof of ownership that is generated based on the file's hash value and the user's secret key; and

wherein the step of determining comprises:

generating a file tag based on the proof of ownership and the user's public key; and

determining whether the file tag has been previously stored.

3. The method according to claim 1 or 2, further comprising:

in response to the positive determination result, acquiring, from the user and another data holder, their hash chains of the file;

checking whether the two hash chains are the same;

in response to a positive check result, recording the user as a data holder; and

in response to a negative check result, refusing the user to be a data holder.

4. The method according to claim 3, wherein the step of acquiring is performed only when the user is a revoked user.

5. The method according to claim 3 or 4, wherein the step of acquiring comprises:

sending to the another data holder the file's first ciphertext and a re-encrypted first cipher key corresponding to the another data holder; and
receiving a hash chain from the another data holder.

6. The method according to any one of claims 1 to 5, wherein the first symmetric key is homomorphically encrypted with the authorized party's public key;

wherein the re-encryption key is generated based on the data holder's public key and the authorized party's secret key; and

wherein the proof of ownership is generated based further on the cloud service provider's public key.

7. The method according to any one of claims 1 to 6, wherein the public key and secret key pairs of the cloud service provider, the authorized party and the user are generated based on bilinear mapping;

wherein the step of re-encrypting is performed based on bilinear mapping; and
wherein the file tag is generated based on bilinear mapping.

8. The method according to any one of claims 1 to 7, further comprising:
encrypting the file's first ciphertext with a second symmetric key to update the file's first ciphertext to its second ciphertext; and

cooperating with the authorized party to update the first cipher key to a second cipher key based on additive homomorphism, the second cipher key being a ciphertext of a weighted sum of the first and second symmetric keys.

9. The method according to claim 8, wherein the step of cooperating comprises:

obtaining a first updating key from the authorized party;
homomorphically encrypting the second symmetric key with the first updating key; and

calculating a product between the first cipher key and the encrypted second symmetric key to generate the second cipher key.

10. The method according to claim 8 or 9, further comprising:

decrypting the file's second ciphertext with the second symmetric key to generate the file's first ciphertext;

encrypting the file's first ciphertext with a third symmetric key to update the file's second ciphertext to its third ciphertext; and

updating the second cipher key to a third cipher key based on additive homomorphism, the third cipher key being a ciphertext of a weighted sum of the first and third symmetric keys.

11. The method according to claim 10, wherein the step of updating comprises:

homomorphically encrypting, with the first updating key, a difference value between the third and second symmetric keys; and

calculating a product between the second cipher key and the encrypted difference value to generate the third cipher key.

12. The method according to any one of claims 1 to 7, further comprising:

in response to a user revocation event, encrypting the file's first ciphertext with a fourth symmetric key to update the file's first ciphertext to its fourth ciphertext;

cooperating with the authorized party to update the first cipher key based on additive homomorphism, the updated first cipher key being a ciphertext of a weighted sum of the first symmetric key and a third random number; and

cooperating with the authorized party to generate a fourth cipher key from the updated first cipher key based on additive homomorphism, the fourth cipher key being a ciphertext of a weighted sum of the first and fourth symmetric keys.

13. The method according to claim 12, wherein cooperating with the authorized party to update the first cipher key comprises:

obtaining a first updating key from the authorized party;

homomorphically encrypting the third random number with the first updating key; and

calculating a product between the first cipher key and the encrypted third random number to generate the updated first cipher key; and

wherein cooperating with the authorized party to generate the fourth cipher key comprises:

sending the updated first cipher key to the authorized party, wherein the authorized party is able to decrypt the updated first cipher key to its plaintext, and homomorphically encrypt the plaintext to a further updated first cipher key;

receiving the further updated first cipher key and a second updating key from the authorized party;

homomorphically encrypting, with the second updating key, a difference value between the fourth symmetric key and the third random number; and

calculating a product between the further updated first cipher key and the encrypted difference value to generate the fourth cipher key.

14. The method according to claim 8 or 9, further comprising:

in response to a user revocation event, decrypting the file's second ciphertext with the second symmetric key to generate the file's first ciphertext;

encrypting the file's first ciphertext with a fifth symmetric key to update the file's second ciphertext to its fifth ciphertext;

updating the second cipher key based on additive homomorphism, the updated second cipher key being a ciphertext of a weighted sum of the first and second symmetric keys and a fourth random number; and

cooperating with the authorized party to generate a fifth cipher key from the updated second cipher key based on additive homomorphism, the fifth cipher key being a ciphertext of a weighted sum of the first and fifth symmetric keys.

15. The method according to claim 14, wherein updating the second cipher key comprises:

homomorphically encrypting the fourth random number with the first updating key; and

calculating a product between the second cipher key and the encrypted fourth random number to generate the updated second cipher key; and

wherein cooperating with the authorized party to generate the fifth cipher key comprises:

sending the updated second cipher key to the authorized party, wherein the authorized party is able to decrypt the updated second cipher key to its plaintext, and homomorphically encrypt the plaintext to a further updated second cipher key;

receiving the further updated second cipher key and a second updating key from the authorized party;

homomorphically encrypting, with the second updating key, a difference value between the fifth symmetric key and a weighted sum of the second symmetric key and the fourth random number; and

calculating a product between the further updated second cipher key and the encrypted difference value to generate the fifth cipher key.

16. A method implemented at an authorized party, the method comprising:

receiving, from a cloud service provider, a request for generating a re-encryption key for a user as a data holder;

generating the re-encryption key for the data holder; and

sending the re-encryption key to the cloud service provider, wherein the cloud service provider is able to re-encrypt a first cipher key with the re-encryption key, such that the re-encrypted first cipher key can be decrypted with the data holder's secret key to a first symmetric key that can be used to decrypt a file's first ciphertext.

17. The method according to claim 16, wherein the re-encryption key is generated based on the data holder's public key and the authorized party's secret key.

18. The method according to claim 16 or 17, further comprising:

receiving, from the cloud service provider, a request for generating an updating key;

generating a first updating key; and

sending the first updating key to the cloud service provider, wherein the cloud service provider is able to use the first updating key to update a cipher key of the file.

19. The method according to claim 18, further comprising:

receiving the updated cipher key from the cloud service provider;

decrypting the updated cipher key to its plaintext;

homomorphically encrypting the plaintext to a further updated cipher key;

generating a second updating key;

sending the further updated cipher key and the second updating key to the cloud service provider, wherein the cloud service provider is able to use the further updated cipher key and the second updating key to further update the cipher key of the file.

20. The method according to claim 18 or 19, wherein the first and second updating keys are generated based on homomorphic encryption and bilinear mapping.

21. A method implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider, and a notification indicating the user to be a data owner is received from the cloud service provider, the method comprising:

encrypting the file with a symmetric key to generate its ciphertext;
homomorphically encrypting the symmetric key to generate a cipher key; and
sending the ciphertext and the cipher key to the cloud service provider.

22. The method according to claim 21, further comprising:
generating a proof of ownership based on the file's hash value and the user's secret key;

wherein the upload request comprises the proof of ownership.

23. The method according to claim 21 or 22, further comprising:
in response to a request from the cloud service provider for verifying the user's ownership of the file, generating a hash chain from the file; and
sending the hash chain to the cloud service provider.

24. The method according to any one of claims 21 to 23, wherein the symmetric key is homomorphically encrypted with an authorized party's public key;
and

wherein the proof of ownership is generated based further on the cloud service provider's public key.

25. A method implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider, and a notification indicating the user to be a data holder is received from the cloud service provider, the method comprising:

sending to the cloud service provider a download request for downloading the file;

receiving, from the cloud service provider, the file's ciphertext and a re-encrypted cipher key;

decrypting, with the user's secret key, the re-encrypted cipher key to a symmetric key; and

decrypting, with the symmetric key, the file's ciphertext to its plaintext.

26. The method according to claim 25, further comprising:

generating a proof of ownership based on the file's hash value and the user's secret key;

wherein the upload request comprises the proof of ownership.

27. The method according to claim 25 or 26, further comprising:

in response to a request from the cloud service provider for verifying the user's ownership of the file, generating a hash chain from the file; and

sending the hash chain to the cloud service provider.

28. The method according to any one of claims 25 to 27, further comprising:

receiving, from the cloud service provider, a request for verifying another user's ownership, the request comprising the file's ciphertext and a re-encrypted cipher key;

decrypting, with the user's secret key, the re-encrypted cipher key to a symmetric key;

decrypting, with the symmetric key, the file's ciphertext to its plaintext;

generating a hash chain from the file; and

sending the hash chain to the cloud service provider.

29. An apparatus implemented at a cloud service provider, the apparatus comprising:

at least one processor; and

at least one memory including computer-executable code,

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to:

receive, from a user, an upload request for uploading a file;

determine whether to perform deduplication for the file;

in response to a negative determination result, obtain, from the user as a data owner, the file's first ciphertext and a first cipher key, the file being encrypted with a first symmetric key to generate its first ciphertext, the first symmetric key being homomorphically encrypted to generate the first cipher key;

in response to a positive determination result, obtain, from an authorized party, a re-encryption key for the user as a data holder;

re-encrypt the first cipher key with the re-encryption key, such that the re-encrypted first cipher key can be decrypted with the data holder's secret key; and

in response to a download request from the data holder, send to the data holder the file's first ciphertext and the re-encrypted first cipher key.

30. The apparatus according to claim 29, wherein the upload request comprises a proof of ownership that is generated based on the file's hash value and the user's secret key; and

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of determining by:

generating a file tag based on the proof of ownership and the user's public key; and

determining whether the file tag has been previously stored.

31. The apparatus according to claim 29 or 30, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

in response to the positive determination result, acquire, from the user and another data holder, their hash chains of the file;

check whether the two hash chains are the same;

in response to a positive check result, record the user as a data holder; and

in response to a negative check result, refuse the user to be a data holder.

32. The apparatus according to claim 31, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of acquiring only when the user is a revoked user.

33. The apparatus according to claim 31 or 32, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of acquiring by:

sending to the another data holder the file's first ciphertext and a re-encrypted first cipher key corresponding to the another data holder; and

receiving a hash chain from the another data holder.

34. The apparatus according to any one of claims 29 to 33, wherein the first symmetric key is homomorphically encrypted with the authorized party's public key;

wherein the re-encryption key is generated based on the data holder's public key and the authorized party's secret key; and

wherein the proof of ownership is generated based further on the cloud service provider's public key.

35. The apparatus according to any one of claims 29 to 34, wherein the public key and secret key pairs of the cloud service provider, the authorized party and the user are generated based on bilinear mapping;

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of re-encrypting based on bilinear mapping; and

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to generate the file tag based on bilinear mapping.

36. The apparatus according to any one of claims 29 to 35, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

encrypt the file's first ciphertext with a second symmetric key to update the file's first ciphertext to its second ciphertext; and

cooperate with the authorized party to update the first cipher key to a second cipher key based on additive homomorphism, the second cipher key being a ciphertext of a weighted sum of the first and second symmetric keys.

37. The apparatus according to claim 36, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of cooperating by:

obtaining a first updating key from the authorized party;

homomorphically encrypting the second symmetric key with the first updating key; and

calculating a product between the first cipher key and the encrypted second symmetric key to generate the second cipher key.

38. The apparatus according to claim 36 or 37, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

decrypt the file's second ciphertext with the second symmetric key to generate the file's first ciphertext;

encrypt the file's first ciphertext with a third symmetric key to update the file's second ciphertext to its third ciphertext; and

update the second cipher key to a third cipher key based on additive homomorphism, the third cipher key being a ciphertext of a weighted sum of the first and third symmetric keys.

39. The apparatus according to claim 38, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to perform the step of updating by:

homomorphically encrypting, with the first updating key, a difference value between the third and second symmetric keys; and

calculating a product between the second cipher key and the encrypted difference value to generate the third cipher key.

40. The apparatus according to any one of claims 29 to 35, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

in response to a user revocation event, encrypt the file's first ciphertext with a fourth symmetric key to update the file's first ciphertext to its fourth ciphertext;

cooperate with the authorized party to update the first cipher key based on additive homomorphism, the updated first cipher key being a ciphertext of a weighted sum of the first symmetric key and a third random number; and

cooperate with the authorized party to generate a fourth cipher key from the updated first cipher key based on additive homomorphism, the fourth cipher key being a ciphertext of a weighted sum of the first and fourth symmetric keys.

41. The apparatus according to claim 40, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to cooperate with the authorized party to update the first cipher key by:

obtaining a first updating key from the authorized party;

homomorphically encrypting the third random number with the first updating key; and

calculating a product between the first cipher key and the encrypted third random number to generate the updated first cipher key; and

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to cooperate with the authorized party to generate the fourth cipher key by:

sending the updated first cipher key to the authorized party, wherein the authorized party is able to decrypt the updated first cipher key to its plaintext, and homomorphically encrypt the plaintext to a further updated first cipher key;

receiving the further updated first cipher key and a second updating key from the authorized party;

homomorphically encrypting, with the second updating key, a difference value between the fourth symmetric key and the third random number; and

calculating a product between the further updated first cipher key and the encrypted difference value to generate the fourth cipher key.

42. The apparatus according to claim 36 or 37, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

in response to a user revocation event, decrypt the file's second ciphertext with the second symmetric key to generate the file's first ciphertext;

encrypt the file's first ciphertext with a fifth symmetric key to update the file's second ciphertext to its fifth ciphertext;

update the second cipher key based on additive homomorphism, the updated second cipher key being a ciphertext of a weighted sum of the first and second symmetric keys and a fourth random number; and

cooperate with the authorized party to generate a fifth cipher key from the updated second cipher key based on additive homomorphism, the fifth cipher key being a ciphertext of a weighted sum of the first and fifth symmetric keys.

43. The apparatus according to claim 42, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to update the second cipher key by:

homomorphically encrypting the fourth random number with the first updating key; and

calculating a product between the second cipher key and the encrypted fourth random number to generate the updated second cipher key; and

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to cooperate with the authorized party to generate the fifth cipher key by:

sending the updated second cipher key to the authorized party, wherein the authorized party is able to decrypt the updated second cipher key to its plaintext, and homomorphically encrypt the plaintext to a further updated second cipher key;

receiving the further updated second cipher key and a second updating key from the authorized party;

homomorphically encrypting, with the second updating key, a difference value between the fifth symmetric key and a weighted sum of the second symmetric key and the fourth random number; and

calculating a product between the further updated second cipher key and the encrypted difference value to generate the fifth cipher key.

44. An apparatus implemented at an authorized party, the apparatus comprising:

at least one processor; and
at least one memory including computer-executable code,
wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to:
receive, from a cloud service provider, a request for generating a re-encryption key for a user as a data holder;
generate the re-encryption key for the data holder; and
send the re-encryption key to the cloud service provider, wherein the cloud service provider is able to re-encrypt a first cipher key with the re-encryption key, such that the re-encrypted first cipher key can be decrypted with the data holder's secret key to a first symmetric key that can be used to decrypt a file's first ciphertext.

45. The apparatus according to claim 44, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to generate the re-encryption key based on the data holder's public key and the authorized party's secret key.

46. The apparatus according to claim 44 or 45, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

receive, from the cloud service provider, a request for generating an updating key;
generate a first updating key; and
send the first updating key to the cloud service provider, wherein the cloud service provider is able to use the first updating key to update a cipher key of the file.

47. The apparatus according to claim 46, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

receive the updated cipher key from the cloud service provider;

decrypt the updated cipher key to its plaintext;
homomorphically encrypt the plaintext to a further updated cipher key;
generate a second updating key;
send the further updated cipher key and the second updating key to the cloud service provider, wherein the cloud service provider is able to use the further updated cipher key and the second updating key to further update the cipher key of the file.

48. The apparatus according to claim 46 or 47, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to generate the first and second updating keys based on homomorphic encryption and bilinear mapping.

49. An apparatus implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider, and a notification indicating the user to be a data owner is received from the cloud service provider, the apparatus comprising:

at least one processor; and
at least one memory including computer-executable code,
wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to:
encrypt the file with a symmetric key to generate its ciphertext;
homomorphically encrypt the symmetric key to generate a cipher key; and
send the ciphertext and the cipher key to the cloud service provider.

50. The apparatus according to claim 49, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

generate a proof of ownership based on the file's hash value and the user's secret key;
wherein the upload request comprises the proof of ownership.

51. The apparatus according to claim 49 or 50, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

in response to a request from the cloud service provider for verifying the user's ownership of the file, generate a hash chain from the file; and
send the hash chain to the cloud service provider.

52. The apparatus according to any one of claims 49 to 51, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to homomorphically encrypt the symmetric key with an authorized party's public key; and

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to generate the proof of ownership based further on the cloud service provider's public key.

53. An apparatus implemented at a user, wherein an upload request for uploading a file is sent from the user to a cloud service provider, and a notification indicating the user to be a data holder is received from the cloud service provider, the apparatus comprising:

at least one processor; and

at least one memory including computer-executable code,

wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus to:

send to the cloud service provider a download request for downloading the file;

receive, from the cloud service provider, the file's ciphertext and a re-encrypted cipher key;

decrypt, with the user's secret key, the re-encrypted cipher key to a symmetric key; and

decrypt, with the symmetric key, the file's ciphertext to its plaintext.

54. The apparatus according to claim 53, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

generate a proof of ownership based on the file's hash value and the user's secret key;

wherein the upload request comprises the proof of ownership.

55. The apparatus according to claim 53 or 54, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

in response to a request from the cloud service provider for verifying the user's ownership of the file, generate a hash chain from the file; and

send the hash chain to the cloud service provider.

56. The apparatus according to any one of claims 53 to 55, wherein the at least one memory and the computer-executable code are configured to, with the at least one processor, cause the apparatus further to:

receive, from the cloud service provider, a request for verifying another user's ownership, the request comprising the file's ciphertext and a re-encrypted cipher key;

decrypt, with the user's secret key, the re-encrypted cipher key to a symmetric key;

decrypt, with the symmetric key, the file's ciphertext to its plaintext;

generate a hash chain from the file; and

send the hash chain to the cloud service provider.

57. An apparatus implemented at a cloud service provider, the apparatus comprising:

means for performing all steps of the method according to any one of claims 1 to 15.

58. An apparatus implemented at an authorized party, the apparatus comprising:

means for performing all steps of the method according to any one of claims 16 to 20.

59. An apparatus implemented at a user, the apparatus comprising:

means for performing all steps of the method according to any one of claims 21 to 28.

60. A computer program product comprising at least one non-transitory computer-readable storage medium having computer-executable code stored therein, the computer-executable code being configured to, when being executed, cause an apparatus to operate according to any one of claims 1 to 28.

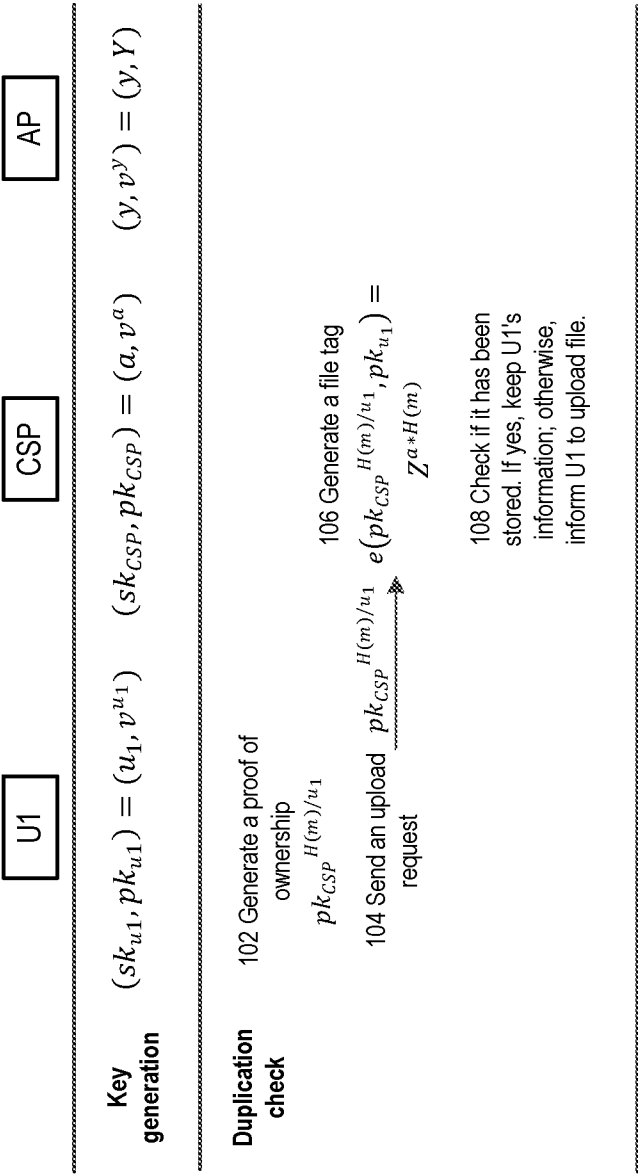


FIG. 1

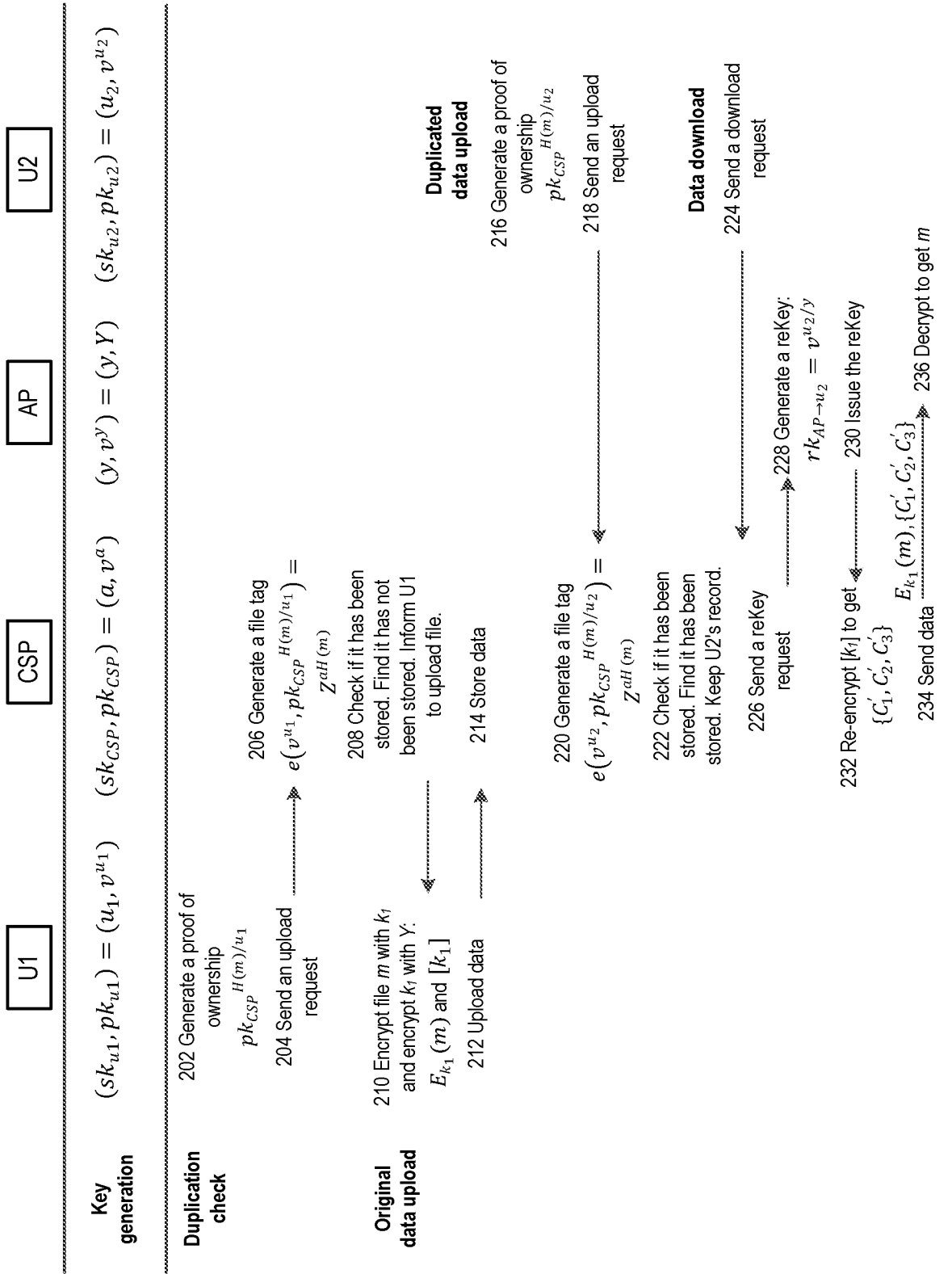


FIG. 2

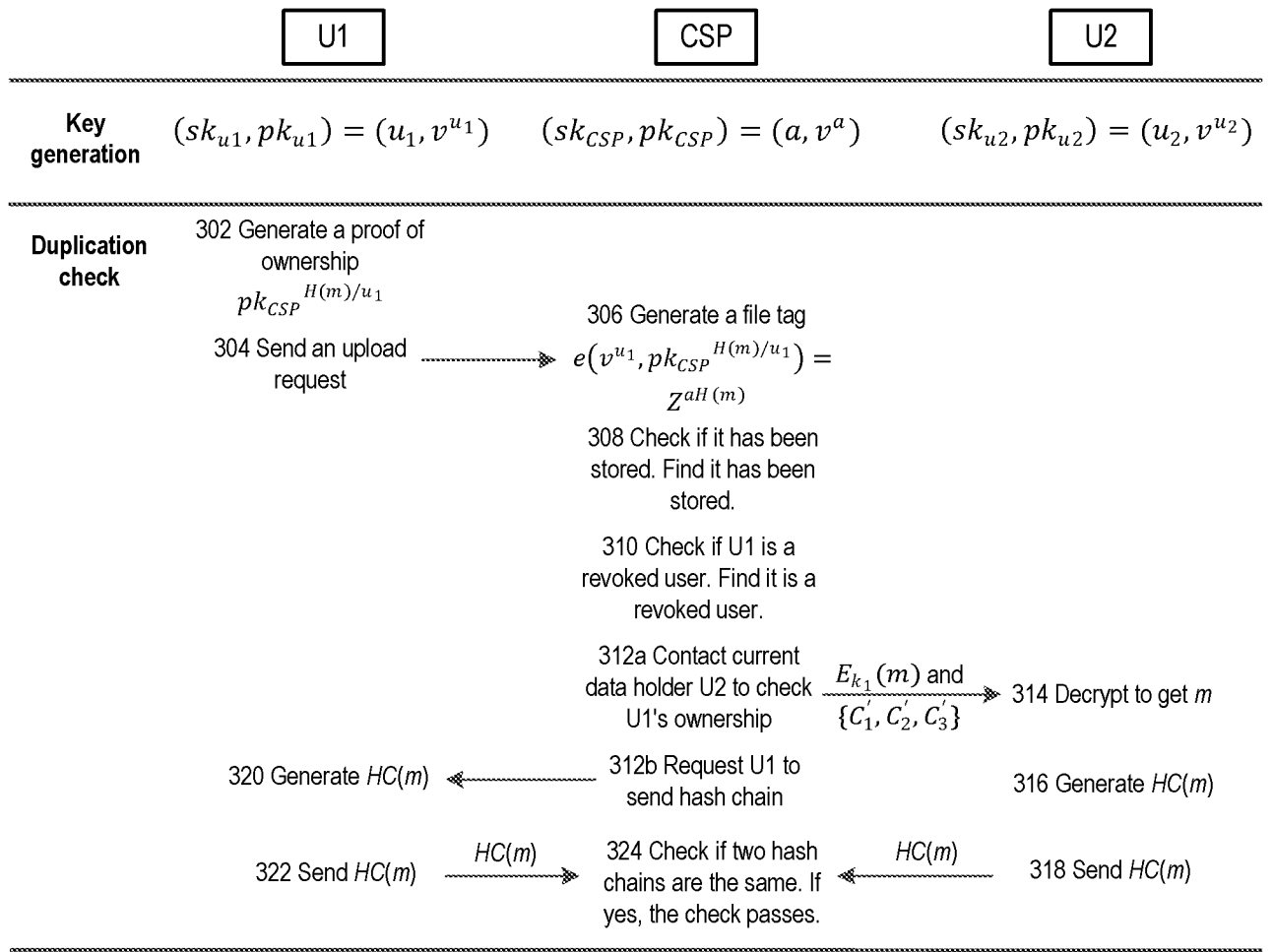


FIG. 3

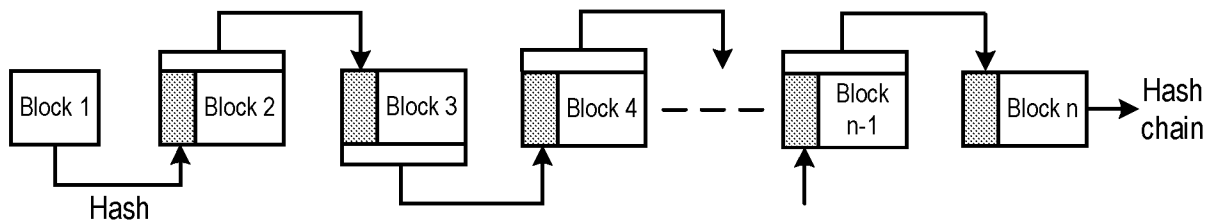


FIG. 4

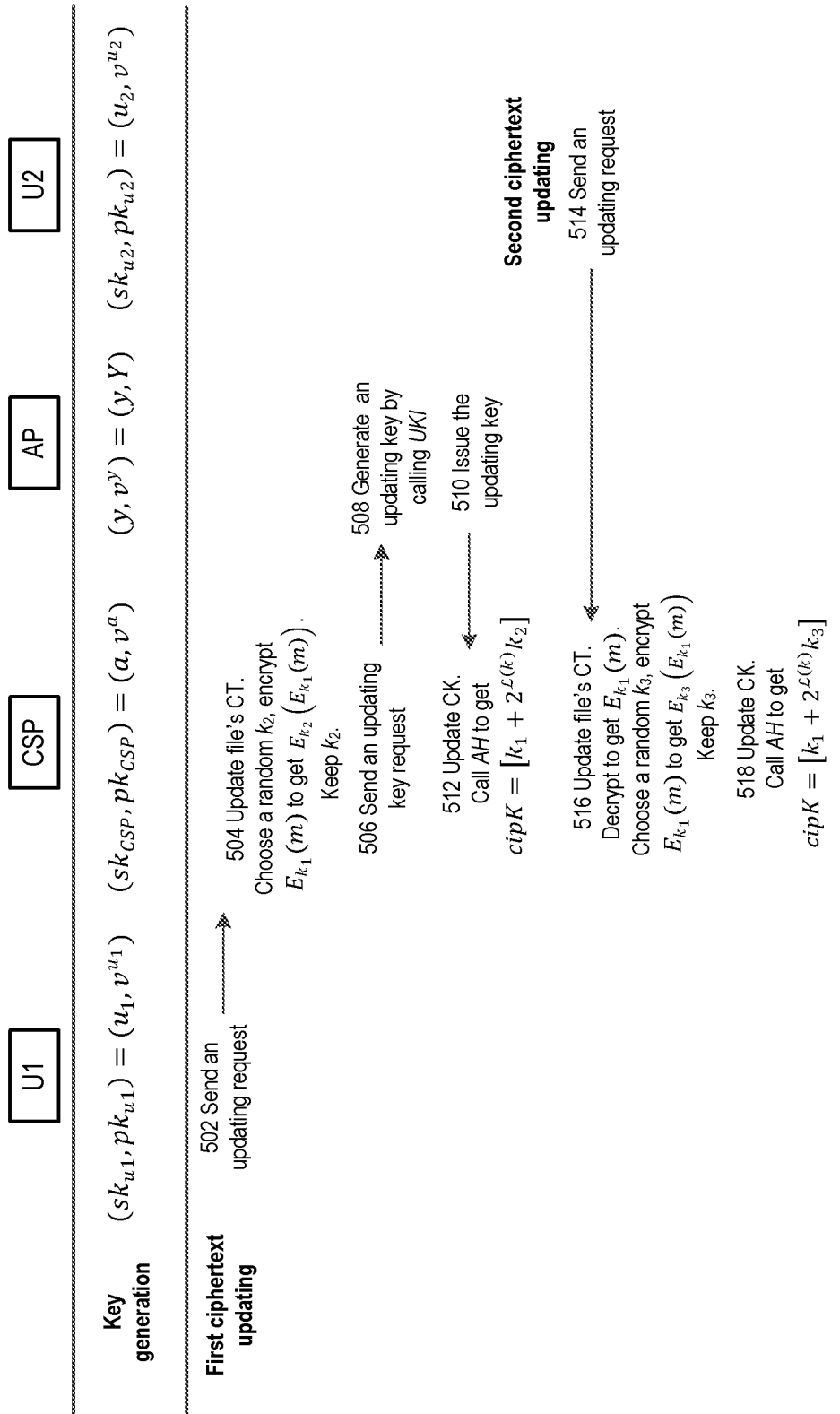


FIG. 5

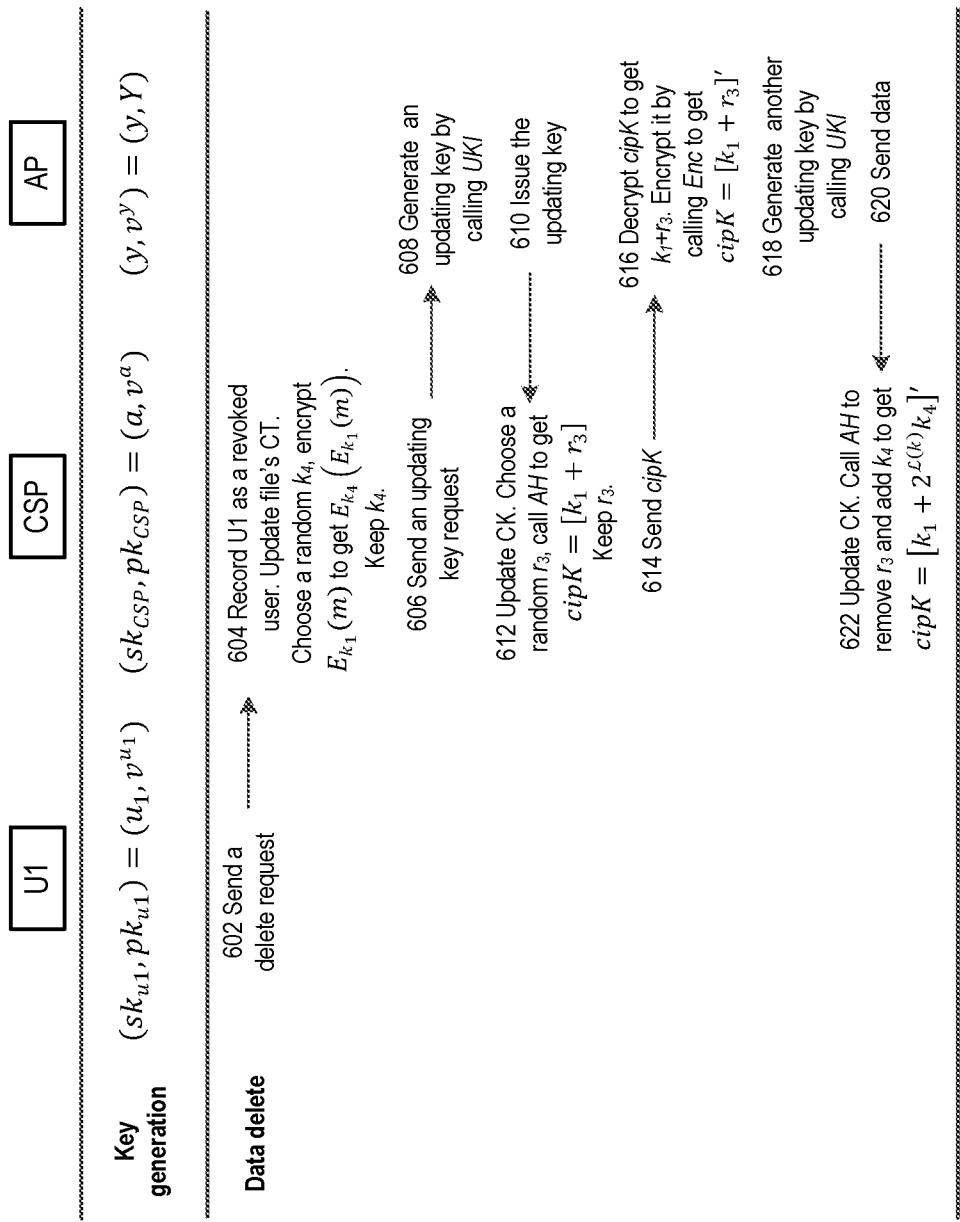


FIG. 6

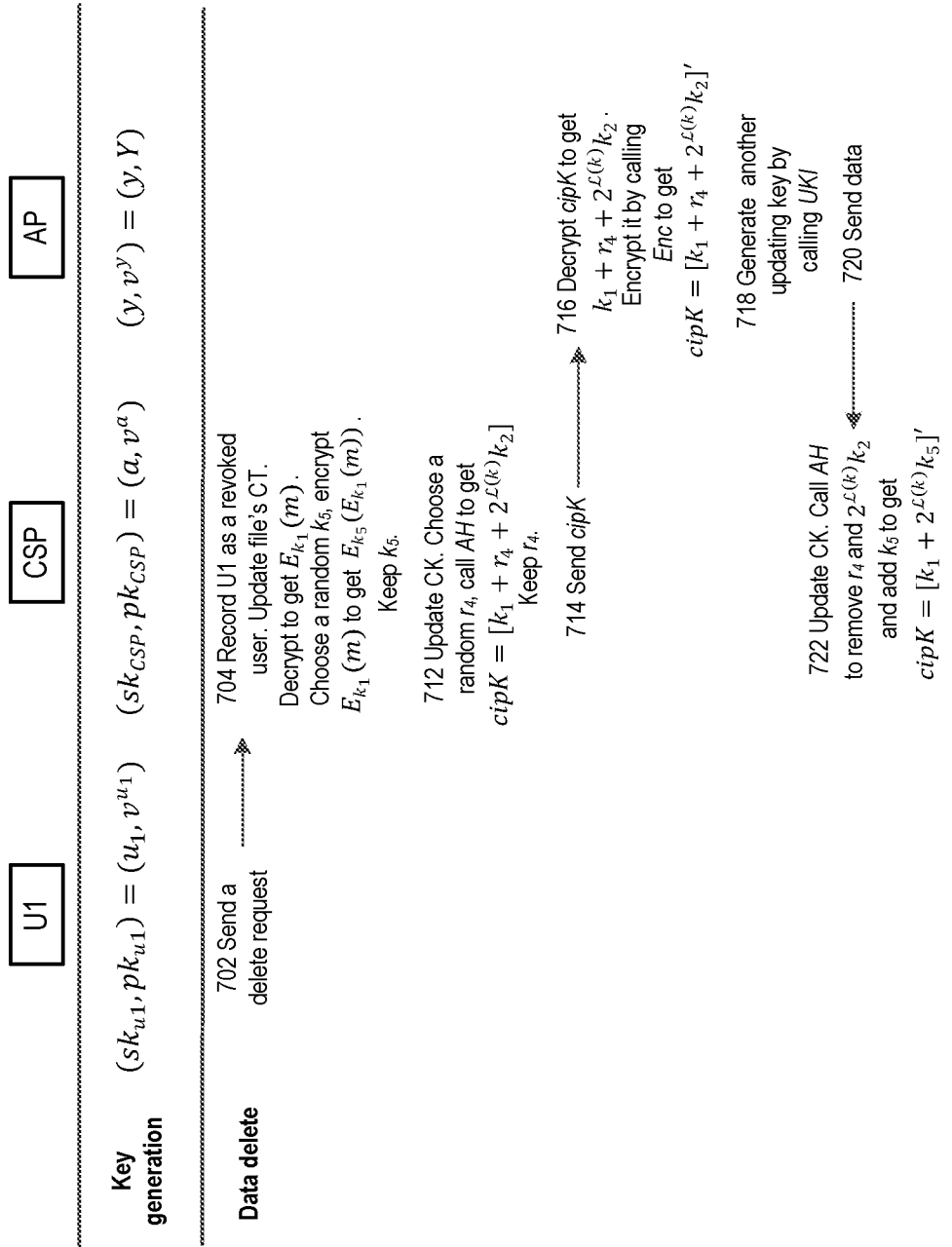


FIG. 7

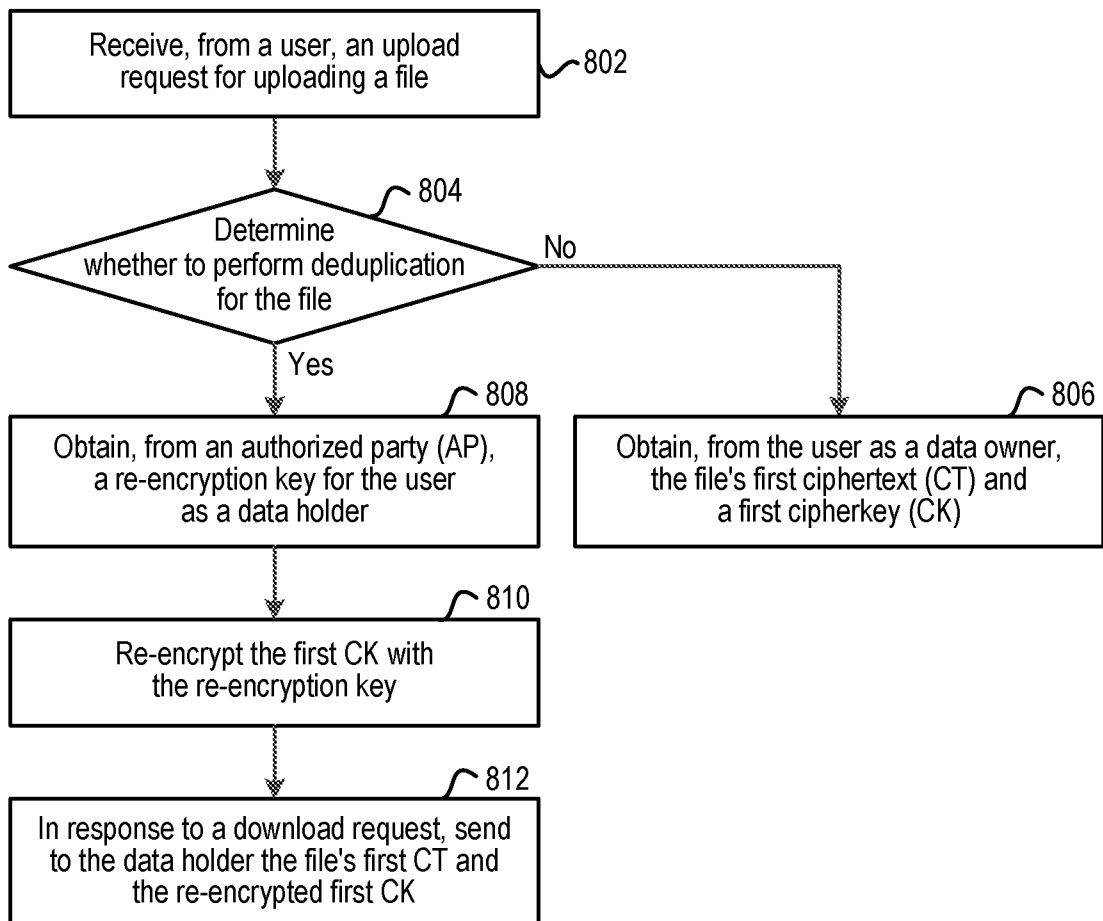


FIG. 8

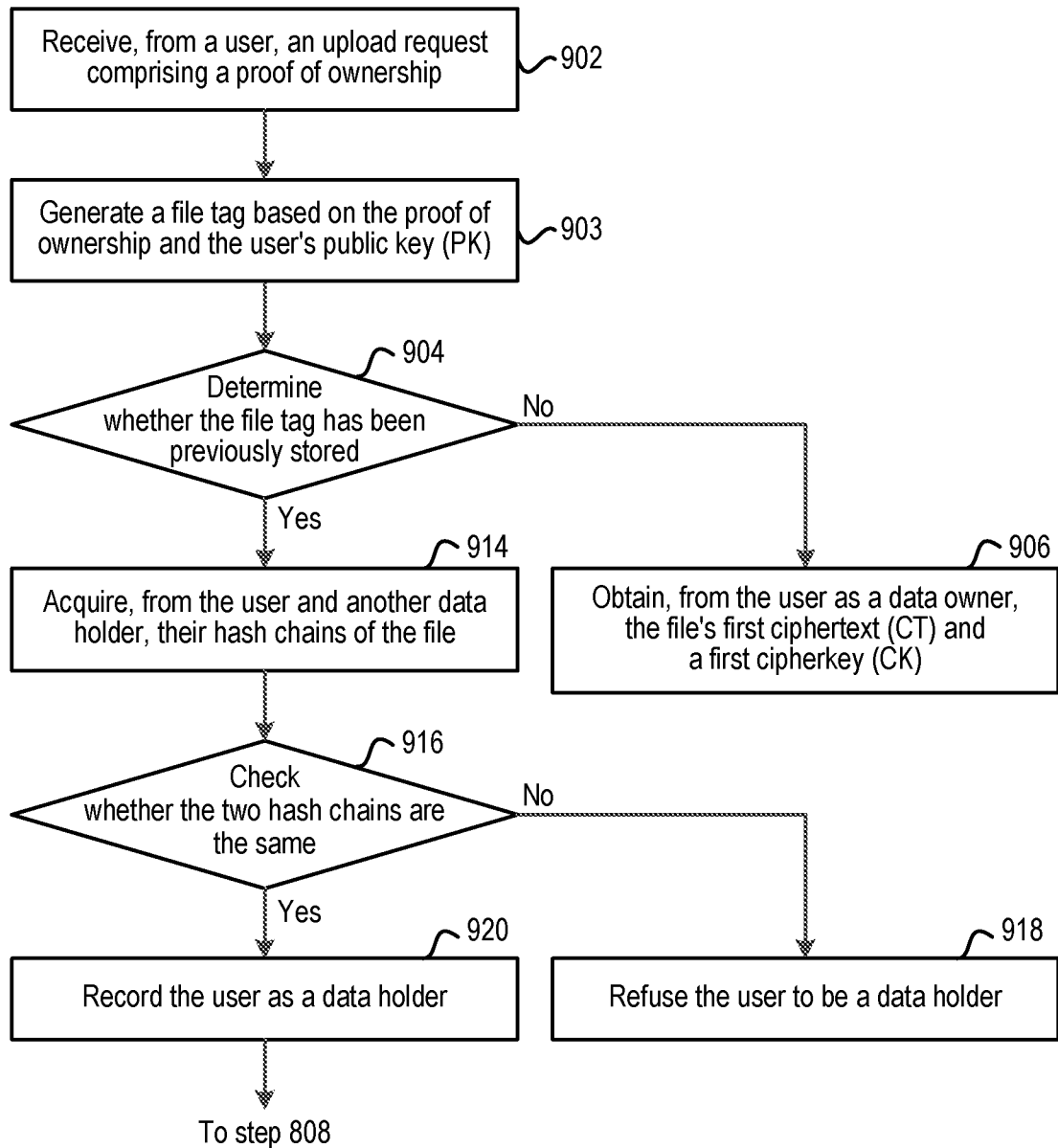


FIG. 9

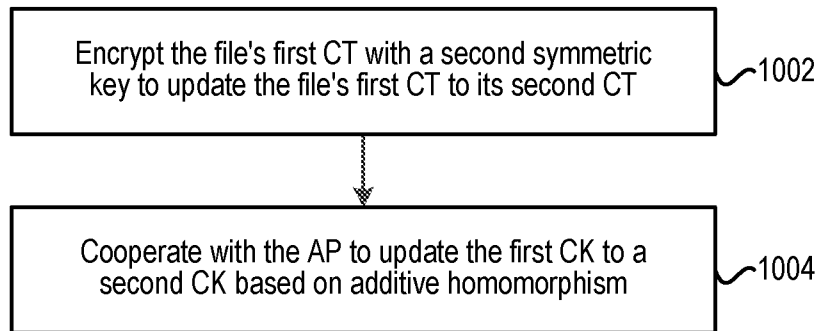


FIG. 10A

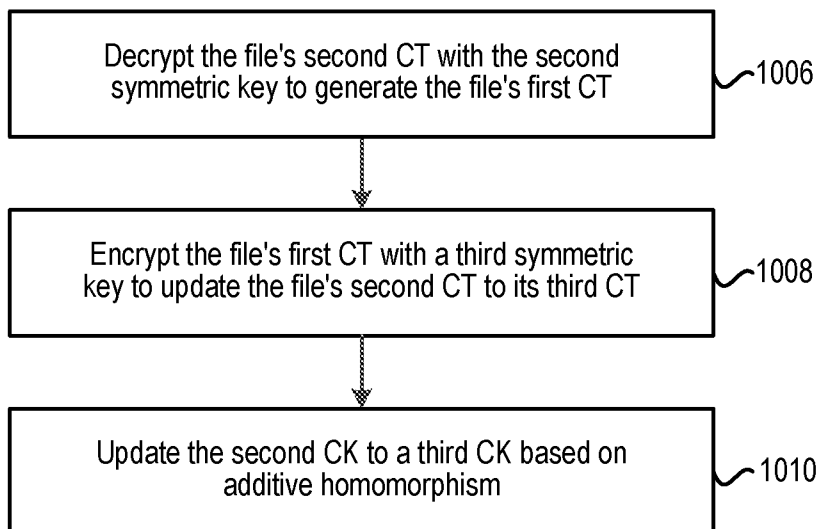


FIG. 10B

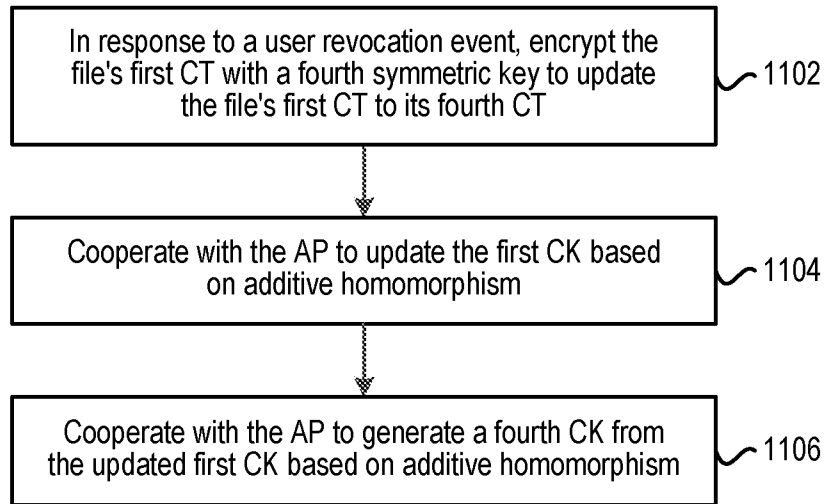


FIG. 11A

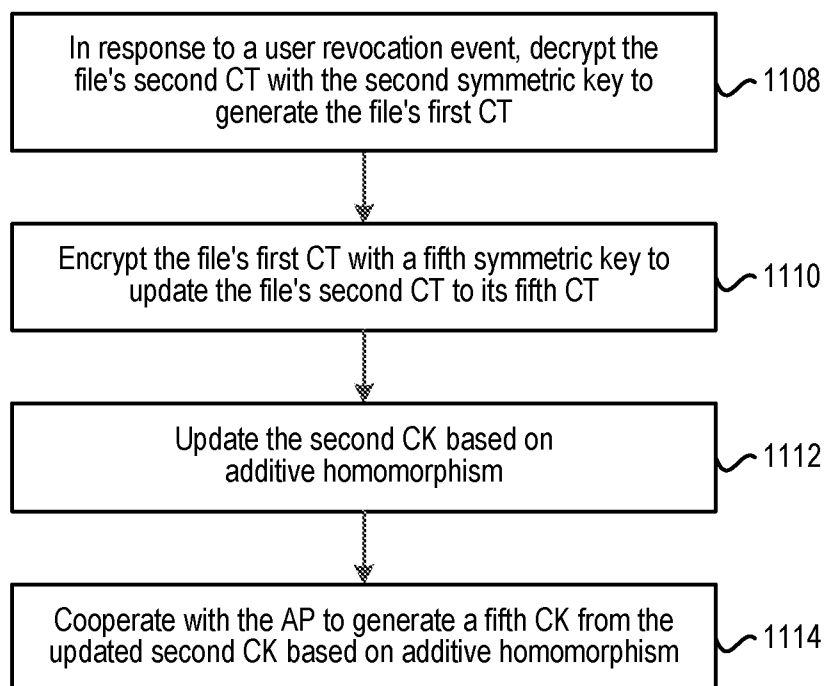


FIG. 11B

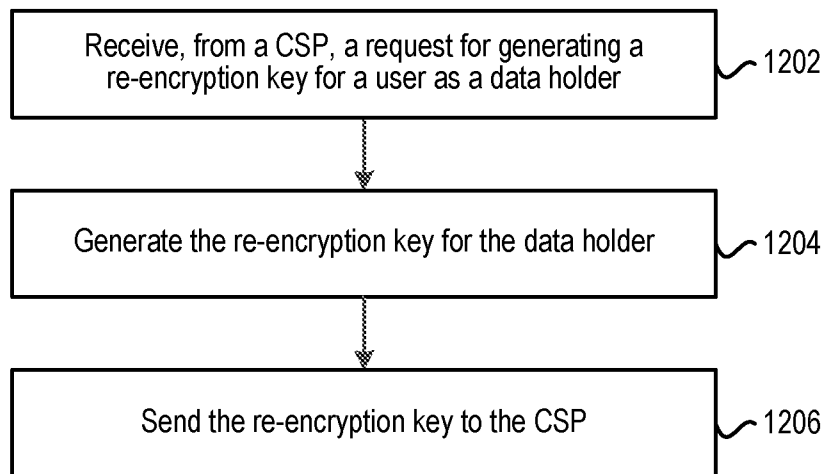


FIG. 12

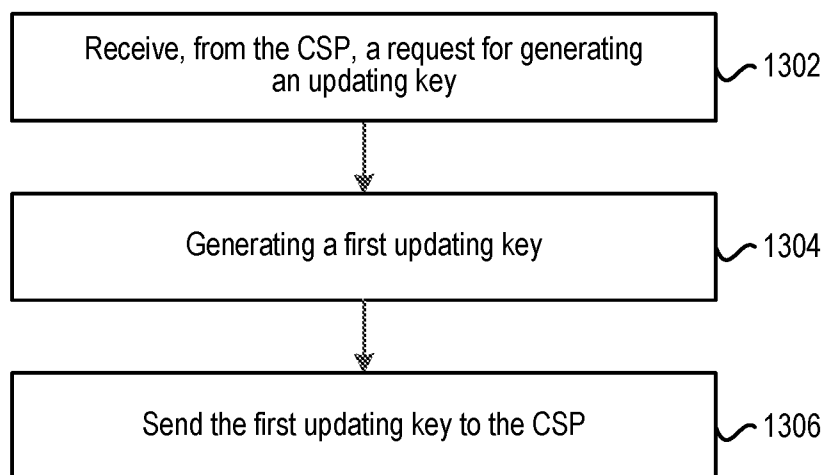


FIG. 13A

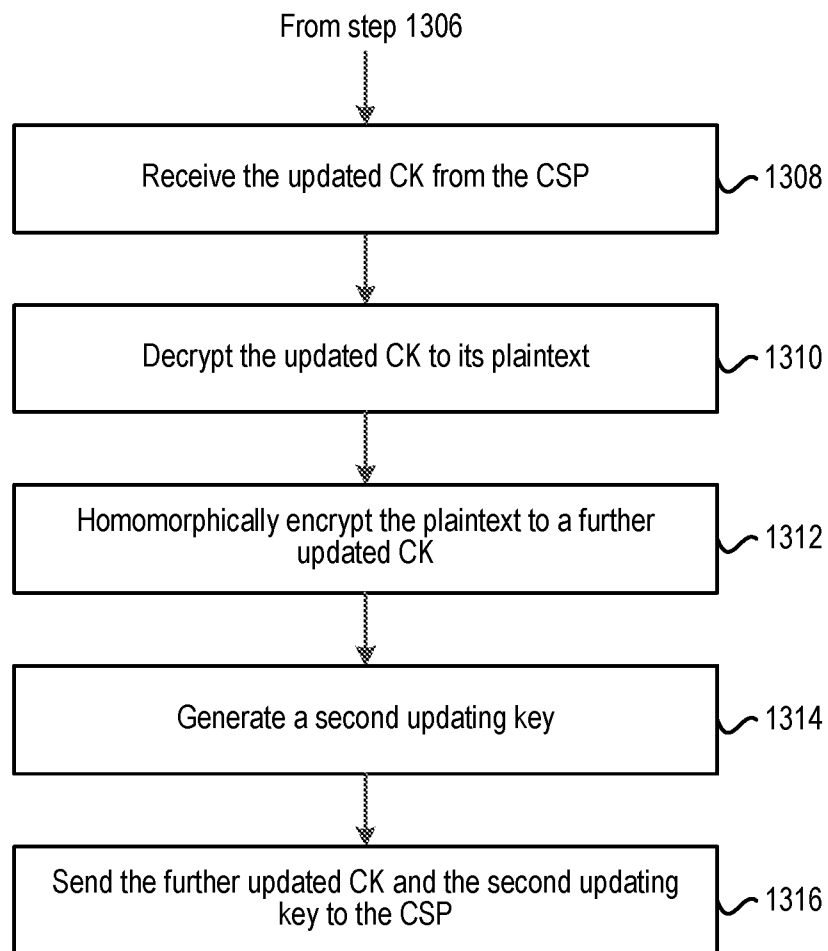


FIG. 13B

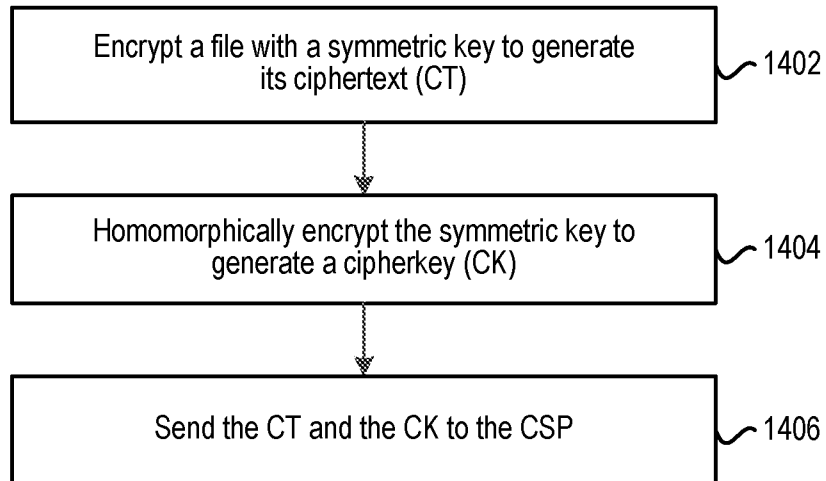


FIG. 14

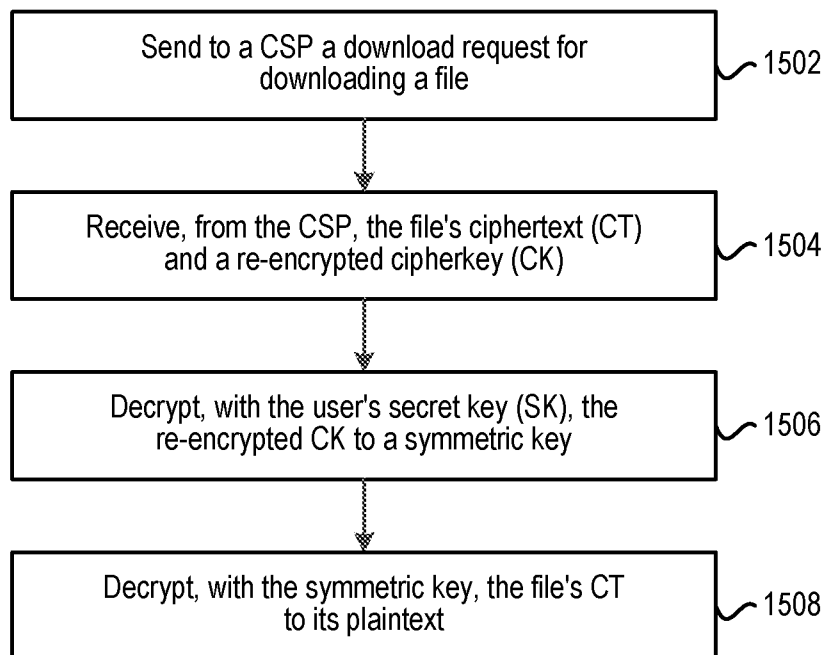


FIG. 15

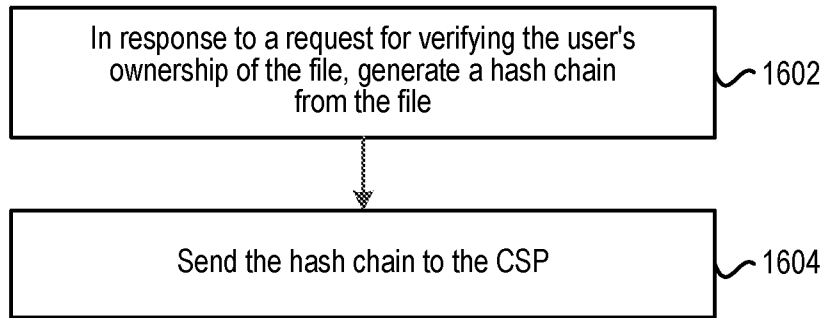


FIG. 16A

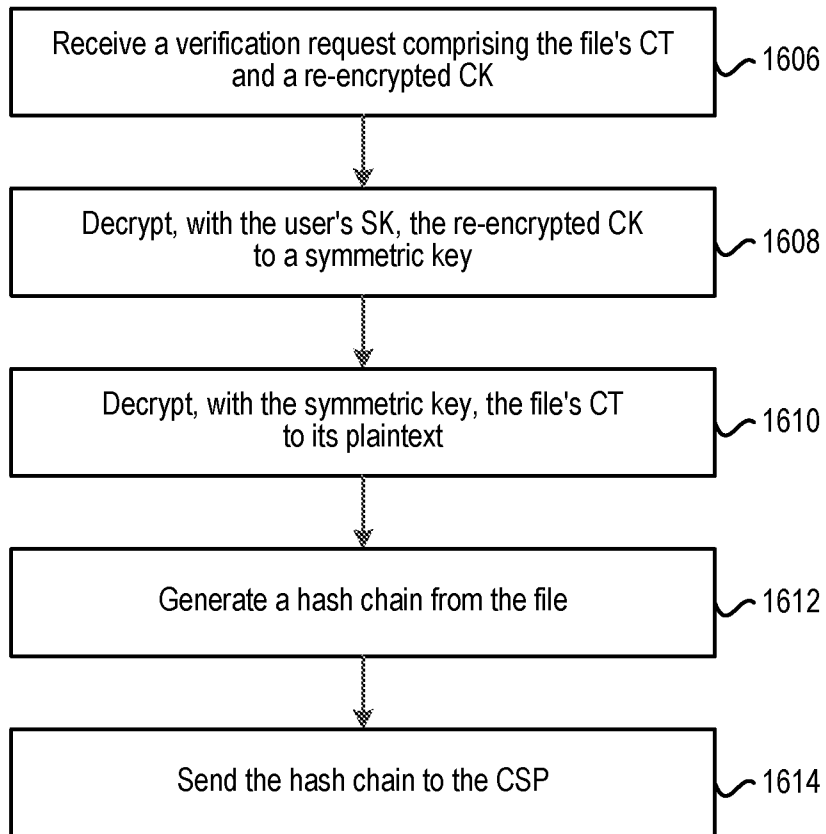


FIG. 16B

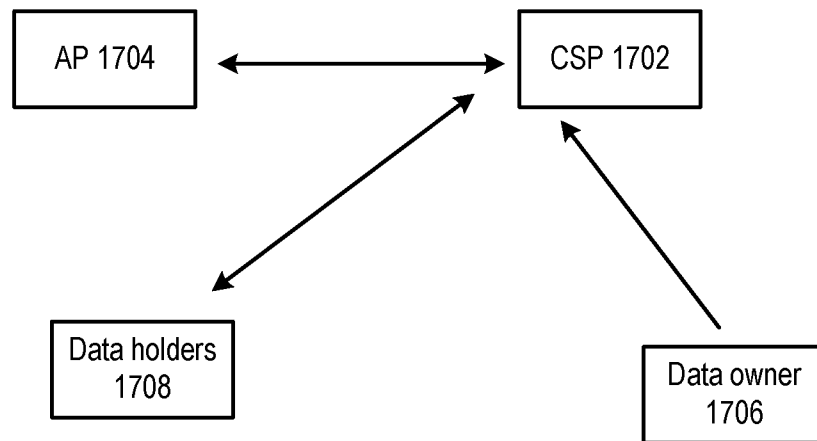


FIG. 17

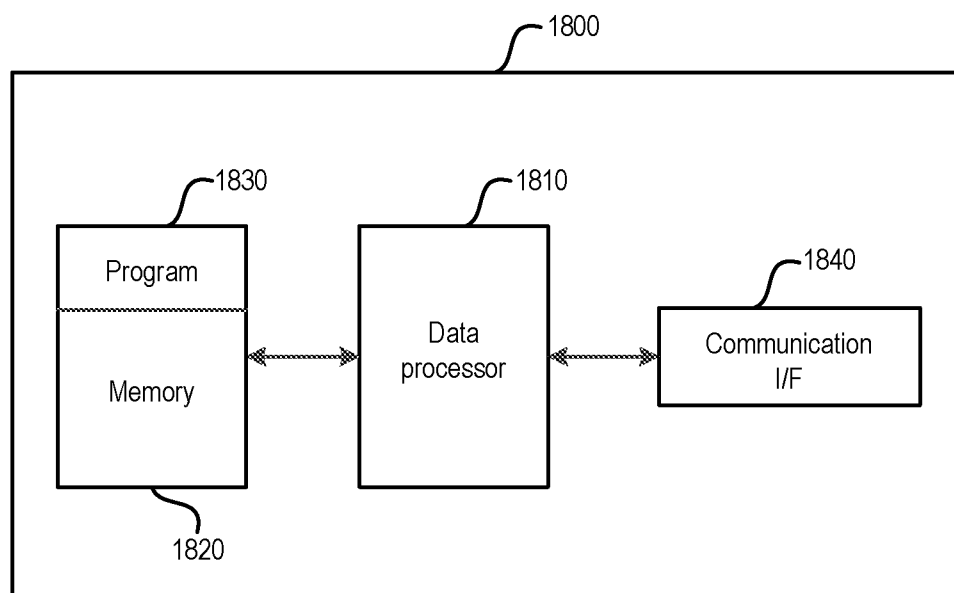


FIG. 18

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2017/080574

A. CLASSIFICATION OF SUBJECT MATTER

H04L 29/06(2006.01)i

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

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

H04L 29/-; H04L 9/-

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNTXT,CNABS, CNKI,SIPOABS,DWPI:cloud, deduplication, owner, holder, homomorphic, encrypt, decrypt, key, authorized party, hash

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2016115663 A1 (NOKIA TECHNOLOGIES OY ET AL.) 28 July 2016 (2016-07-28) claims 1-64, description, paragraphs [0045]-[0155], figures 1-11	16-18, 25-28, 44-46, 53-56, 58-60
Y	WO 2016115663 A1 (NOKIA TECHNOLOGIES OY ET AL.) 28 July 2016 (2016-07-28) claims 1-64, description, paragraphs [0045]-[0155], figures 1-11	1-15, 19-24, 29-43, 47-52, 57-60
Y	CN 106254073 A (WUHAN UNIVERSITY OF TECHNOLOGY) 21 December 2016 (2016-12-21) claims 1-10, description, paragraphs [0002]-[0025]	1-15, 19-24, 29-43, 47-52, 57-60

☐ Further documents are listed in the continuation of Box C.☒ See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

11 January 2018

Date of mailing of the international search report

18 January 2018

Name and mailing address of the ISA/CN

STATE INTELLECTUAL PROPERTY OFFICE OF THE
P.R.CHINA
6, Xitucheng Rd., Jimen Bridge, Haidian District, Beijing
100088
China

Authorized officer

SONG,Yunyun

Facsimile No. (86-10)62019451

Telephone No. (86-10)62089120

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2017/080574

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
WO	2016115663	A1	28 July 2016	EP	3248354	A1	29 November 2017
				CN	107113314	A	29 August 2017
				IN	201747025538	A	28 July 2017
CN	106254073	A	21 December 2016	None			