REPUBLIC OF SOUTH AFRICA REPUBLIEK VAN SUID AFRIKA

PATENTS ACT, 1978

CERTIFICATE

In accordance with section 44 (1) of the Patents Act, No. 57 of 1978, it is hereby certified that:

Jinan University

Has been granted a patent in respect of an invention described and claimed in complete specification deposited at the Patent Office under the number

2023/00638

A copy of the complete specification is annexed, together with the relevant Form P2.

In testimony thereof, the seal of the Patent Office has been affixed at Pretoria with effect from the **29th** day of **March 2023**

[illegible]

Registrar of Patents

FORM P2

REPUBLIC OF SOUTH AFRICA
PATENTS ACT, 1978
REGISTER OF PATENTS

Official application No.			Lodging date: Provisional		Acceptance date		
21	01	2023/00638	22		47	7 March 2023	
International classification			Lodging date: National phase		Granted date		
51	G06F		23	16 January 2023		29 March 2023	

71	Full name(s) of applicant(s)/Patentee(s):	
Jinan University		
71	Applicant(s) substituted:	Date registrered
71	Assignee(s):	Date registrered
72	Full name(s) of inventor(s):	
(1) YANG Anjia; (2) GUO Zifan; (3) WENG Jian; (4) TONG Yao; (5) PEI Fuqing; (6) JIANG Changkun		

Priority claimed:	Country	Number	Date

54	Title of invention
LIGHTWEIGHT DUPLICATES-REMOVING CRYPTOGRAPH INTEGRITY AUDIT METHOD AND SYSTEM	
Address of applicant(s)/patentee(s):	
No. 601, Huangpu Avenue West, Tianhe District, Guangzhou, Guangdong, China	
74	Address for service
Sibanda and Zantwijk, Oaktree Corner, 9 Kruger Street, Oaklands (PO Box 1615 Houghton 2041), Johannesburg, 2192, SOUTH AFRICA Reference no.: PT_CP_ZA00007072 ([InsID:])	

61	Patent of addition No.	Date of any change
Fresh application based on.		Date of any change

REPUBLIC OF SOUTH AFRICA
PATENTS ACT, 1978
COMPLETE SPECIFICATION
[Section 30(1) – Regulation 28]

FORM P7

OFFICIAL APPLICATION NO.

21	01	2023/00638

LODGING DATE

22	16 January 2023

INTERNATIONAL CLASSIFICATION

51	G06F

FULL NAME(S) OF APPLICANT(S)

71	Jinan University

FULL NAME(S) OF INVENTORS(S)

72	YANG Anjia GUO Zifan WENG Jian TONG Yao PEI Fuqing JIANG Changkun

TITLE OF INVENTION

54	LIGHTWEIGHT DUPLICATES-REMOVING CRYPTOGRAPH INTEGRITY AUDIT METHOD AND SYSTEM

LIGHTWEIGHT DUPLICATES-REMOVING CRYPTOGRAPH INTEGRITY AUDIT METHOD AND SYSTEM

TECHNICAL FIELD

The application relates to the technical field of cyberspace security, and in particular to a lightweight duplicates-removing cryptograph integrity audit method and a system.

BACKGROUND

With the progress of cloud computing technology, big data has been integrated into people's daily life, and the user data is growing rapidly.

On the one hand, in order to reduce the operation and maintenance cost, individuals or enterprises tend to outsource data storage to the platform of cloud storage service providers and often do not keep the original data. Once the original data is deleted, it also means that the data owner loses control of the original data. Because cloud storage service providers have a strong motivation to destroy the integrity of users' data (for example, by compressing the pictures or videos stored in the cloud to save the actual storage space and obtain additional profits), the above scenarios pose a huge threat to users' data integrity. Message verification code and digital signature are the effective means used by cryptography to solve data integrity in the past. However, these two traditional schemes may not be directly used in massive data scenarios. These two schemes require users to have all the original data. As the data throughput of a single disk (the throughput is the sum of the reading and writing speeds) of cloud servers used for archiving and storage is 30-40 MB/s at present, even in an ideal situation, the reading speed of a single disk is 40 MB/s, and it takes about 7 hours to read 1TB files, so it is extremely costly in performance, and it is unrealistic. In 2007, Juels and Ateniese proposed PoR and PDP schemes. With the assistance of erasure codes, they both used the probabilistic algorithm in the form of random sampling to avoid the problem that the data to be tested needs to be read in full, and initially solved the problem of data integrity audit of large files. Subsequently, many researchers at home and abroad iterated the scheme based on their ideas, realizing security reinforcement, efficiency optimization and expansion of security model.

On the other hand, cloud storage service providers have found that the files uploaded by users are similar, and there are a large number of duplicate files. The big data platform already has mature technology and related practices for deduplication of unencrypted files. However, in recent years, domestic laws and regulations have strengthened the protection of the confidentiality of user data, requiring enterprises to encrypt and store user data. Therefore, the traditional plaintext deduplication technology has been difficult to adapt to the development of the Internet for a long time, and the deduplication technology of secret data will be gradually applied to commercial applications. The basic idea of deduplication is to generate a key for deduplication encryption from file data, and then encrypt the file data with the key. The above operation encrypts the same file to obtain the same ciphertext file, thus realizing deduplication.

With the deepening of the research on integrity audit and cryptograph deduplication, more and more scholars consider the system that supports both cryptograph deduplication and integrity audit. A direct way to support both functions is to simply superimpose deduplication and audit schemes. But every time a user uploads a file, even if it is deduplicated, the system needs to recalculate a set of tags for the file. The calculation time of tags is linearly related to the file size, so it is extremely inefficient. In recent years, some scholars have proposed to generate audit private key by deduplication key, that is, the generation of integrity tag is directly related to deduplication key. When files have duplicates, there is no need to recalculate the tag, because the integrity tag of the same file is completely the same. Although this method improves the efficiency of file uploading, it also makes the management of audit public key difficult because of rigid tag generation. More specifically, because audit key is derived from file content, the audit public key of each file is different. If a user owns 100 files, the user needs to manage 100 audit public keys, and this is unrealistic. In addition, creating personalized tags for duplicate files for users also represents the user's ownership of the files.

SUMMARY

The application provides a lightweight duplicates-removing cryptograph integrity audit method and a system, and they are suitable for various cryptograph deduplication systems. In the case of duplicate data, the reproduction of a group of user personalized integrity tags of the file only needs constant level calculation cost, the communication cost is also reduced to constant level, and the storage service provider only needs to store constant level tag auxiliary variables, instead of storing a group of user tags as in the traditional scheme, so as to approximately realize duplicate tag deletion.

To achieve the above objective, this application provides the following solutions.

A lightweight duplicates-removing cryptograph integrity audit method, including:

selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key;

calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server;

judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials and uploading them to the storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; and

sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

Preferably, the calculation method of the audit public key includes: randomly selecting a number *x* as the audit private key, selecting a generator *g* from a cyclic multiplication group $\mathbb{G}$, and calculating g^x as the audit public key.

Preferably, the method for calculating the deduplication key includes: calculating a hash value of the message plaintext as the deduplication key.

Preferably, the method of the first encryption processing includes: encrypting, cutting and coding the original file based on the deduplication key.

Preferably, the first operation includes:

calculating the block tag based on the block data in the secret file data;

generating a file digest, and calculating the user tag conversion auxiliary variables based on the audit private key and the file digest; and

encrypting key parameters based on the user tag conversion auxiliary variables to obtain the auxiliary variables.

Preferably, the second operation includes:

decrypting encrypted variables in the auxiliary variables based on the deduplication key to obtain the user tag conversion auxiliary variables; and

calculating based on the user tag conversion auxiliary variables to obtain the user tag transform auxiliary materials.

Preferably, a generation method of the integrity certificate includes:

generating a block subscript set and a random number set based on the total data block; and

generating the integrity certificate based on the block subscript set, the random number set and the secret file data.

The application also provides a lightweight duplicates-removing cryptograph integrity audit system, including: a private key generation module, an encryption module, a judgment module and a verification module;

the private key generation module is used for selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key;

the encryption module is used for calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server;

the judging module is used for judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials, and uploading them to a storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; and

the verification module is used for sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

The application has the following beneficial effects.

Firstly, the public key used for auditing is no longer determined by the number of files as in the previous schemes, but by the number of users. Each user's file has its own personalized integrity tag, and this is extremely flexible, thus solving the problems of many audit public keys and difficult management of users when there are many files.

Secondly, in the case of duplicate files, users do not need to calculate a set of personalized integrity tags from scratch, but realize tag conversion by constant level calculation, greatly reducing the time required for file uploading and the communication cost. It is worth mentioning that the storage service provider also chooses to store only the auxiliary materials for tag conversion, achieving the effect similar to that of integrity tag deduplication and reducing the storage cost.

Thirdly, the specific scheme of the system cryptograph deduplication technology is decoupled, and it may be applied to most cryptograph deduplication schemes.

BRIEF DESCRIPTION OF THE FIGURES

In order to explain the embodiments of the application or the technical solutions in the prior art more clearly, the drawings that need to be used in the embodiments will be briefly introduced in the following. It is obvious that the drawings described below are only some embodiments of the application. For those of ordinary skill in this field, other drawings may be obtained according to these drawings without creative efforts.

FIG. 1 is a schematic flow diagram of the method of the Embodiment 1 of the present application.

FIG. 2 is a flowchart of the file uploading stage of the Embodiment 2 of the application.

FIG. 3 is a flowchart of the audit stage of the Embodiment 2 of this application.

FIG. 4 is a schematic diagram of the system structure of Embodiment 3 of this application.

DESCRIPTION OF THE application

The technical solutions in the embodiments of the present application will be clearly and completely described below with reference to the drawings in the embodiments of the present application. Obviously, the described embodiments are only part of the embodiments of the present application, but not all of them. Based on the embodiment of the present application, all other embodiments obtained by ordinary technicians in the field without making creative efforts are within the protection scope of the present application.

To make the above objectives, features and advantages of the present application more obvious and understandable, the present application will be explained in further detail below with reference to the drawings and detailed description.

In order to make the above objectives, features and advantages of this application more obvious and understandable, the application will be further explained in detail below with reference to the drawings and detailed description.

Firstly, the definitions of some letters and formulas involved in the present application are explained.

$\mathbb{G}$: a cyclic group of order p, where p is a prime number, and the group is defined as a multiplication group.

g: a generator of the multiplication group G, and it is a public security parameter shared by the whole system.

x,y: secret parameters, they are used as audit private keys of user A and user B respectively, and are selected from Z_p.

pk_A, pk_B, sk_A, sk_B : public keys and private keys used for integrity audit, where pk_A is g^x, sk_A is x, pk_B is g^y, and sk_B is y.

$h(\cdot)$: an arbitrary hash function, and its function is to map data m to a cluster of deduplication keys.

k_{dedup}: a key used when the file is performed deduplication encryption.

ku: a user privater key, the privater key of the encryption algorithm used when encrypting the user's deduplication privater key k_{dedup}, and it has an optional range related to the encryption algorithm used.

C_{data}: confidential data obtained by performing deduplication encryption on the contents of the file.

C_{key}: cryptograph obtained by encrypting k_{dedup} by the user private key ku with encryption algorithm.

dir, *filename*, and *ma* represent respectively: file storage directory, file name, file digest.

σ_i: integrity tag of the ith data in the file.

ω、ω'、W : represent respectively $H_2(x||ma)$, $H_2(y||ma)$ and $\omega' - \omega$, wherein the function of $H_2(\cdot)$ is to map arbitrary data to Z_p.

$H_1(\cdot)$: the function is to map arbitrary data to $\mathbb{G}$.

r, r': temporary secret parameters, random numbers, selected from Z_p.

v, v', h, h', h_1, h_1', V: abbreviations for u^r, $u^{r'}$, g^{ω}, $g^{\omega'}$, $g^{x\omega}$, $g^{y\omega'}$, and vv' respectively.

trans: $-(\frac{1}{x+H_2(ma||h)} + r)$, a user tag conversion auxiliary variable, and it has two functions: first, users use this variable of other users to derive their own tag conversion auxiliary variable; second, the server uses this variable to transform the integrity standard of a specific user into the integrity tag of the variable producer (For example, if user B generates user B’s user tag conversion auxiliary variable via user A’s user tag conversion auxiliary variable, the storage service provider calculates the integrity tag of user B by using user A’s integrity tag and user B’s user tag conversion auxiliary variable, that is, user A is regarded as a specific user here).

trans': $\frac{1}{y+H_2(ma||h')} - r' + trans$.

C_{trans}: $E_{k_{dedup}}(\omega)||trans||v$, user tag conversion auxiliary variable after partial secret parameters are encrypted.

TAM, TAM': respectively $(ma||n||h||h_1)$, $(ma||n||h'||h_1')$, and they are auditor's audit materials.

v_i: ith random number in the challenge generated by the auditor.

$\bar{\rho}$: $\sum m_i v_i$.

$\bar{\sigma}$: aggregation tag, and it is used to verify integrity.

Embodiment 1

The Embodiment 1 includes three roles: user, auditor and storage server, wherein the storage server selects the storage service provider; as shown in FIG. 1, a lightweight duplicates-removing cryptograph integrity audit method includes:

S1: selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key.

In this embodiment, the user randomly selects the number ku as the user private key, randomly selects a number x as the audit private key, selects a generator g from a cyclic multiplication group $\mathbb{G}$, and calculates g^x as the audit public key.

S2: calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server.

The method for calculating the deduplication key includes: calculating a hash value of the message plaintext as the deduplication key; the method of the first encryption processing includes: encrypting, cutting and coding the original file based on the deduplication key.

In this embodiment, the hash value $h(m)$ of the message plaintext is calculated as the deduplication key and recorded as k_{dedup}; the original file data is encrypted, cut and encoded by using the deduplication key k_{dedup}, and then the secret file data C_{data} is obtained. The user private key ku is used as the encryption key to encrypt the deduplication key k_{dedup} to obtain the secret deduplication key C_{key}; the file directory dir, file name $filename$, secret file data C_{data} and secret key C_{key} are uploaded to the server. The file directory is created according to dir, and C_{key} is stored in the corresponding directory and named with $filename.key$.

S3: judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials and uploading them to the storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server.

Where, the first operation includes: calculating the block tag based on the block data in the secret file data; generating a file digest, and calculating the user tag conversion auxiliary variables based on the audit private key and the file digest; and encrypting key parameters based on the user tag conversion auxiliary variables to obtain the auxiliary variables. The second operation includes: decrypting encrypted variables in the auxiliary variables based on the deduplication key to obtain the user tag conversion auxiliary variables; and calculating based on the user tag conversion auxiliary variables to obtain the user tag transform auxiliary materials.

In this embodiment, if there is no secret file data C_{data} in the server, the user is informed that there is no duplicate file; after the user receives the notification that there is no duplication, the user sequentially calculates the tag σ_i for the block data in C_{data}, generates the file digest ma, calculates the user tag conversion auxiliary variable $trans$ by using the audit private key x and the file digest ma, and then sends all the block tags σ_i, the auxiliary variable C_{trans} encrypted with the key parameter r and the auditor’s audit materials TAM to the storage service provider; if the secret file data exists, the user is informed that the file may be deduplicated and C_{trans} is returned to the user. The user decrypts the encrypted variables in C_{trans} by the deduplication key k_{dedup} to obtain $trans$, calculates the user tag transform auxiliary material $trans'$ through $trans$, and then sends the user tag transform auxiliary material $trans'$ and auditor's audit material TAM' to the storage service provider.

S4: sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

Where, a generation method of the integrity certificate includes: generating a block subscript set and a random number set based on the total data block; and generating the integrity certificate based on the block subscript set, the random number set and the secret file data.

In this embodiment, when the auditor wants to audit data, it is necessary to inform the storage service provider of the filename *filename* of the audit target file and the total n number of data blocks of the storage service provider's response, and the auditor randomly generates a block subscript set and a random number set $Q = \{i, v_i\}_{i\in[1,n]}$ by using the total number of data blocks; the storage service provider generates a data integrity certificate $(\bar{\rho}, \bar{\sigma})$ according to the block subscript set and the random number set Q and the stored corresponding user secret file data C_{data}, and then returns the auditor's audit material *TAM* and the integrity certificate $(\bar{\rho}, \bar{\sigma})$ to the user. At last, the auditor uses the audit public key g^x and the audit material *TAM* to verify the integrity certificate. If it passes the test, it means that the sampled data is complete; otherwise, it means that the sampled data has been destroyed.

Embodiment 2

In Embodiment 2, taking user A and user B as examples, the specific workflow of this application is introduced:

in the uploading stage, as shown in FIG. 2, user A owns the file m, and this file has no duplicate data in the server. First, the data owner performs *hash* calculation on the file m to obtain $h(m)$, and in addition, generates a digest ma of the file m. The deduplication key k_{dedup} is set as $h(m)$, and the file m is encrypted with the deduplication key k_{dedup} to obtain C_{data}; the user key ku is selected, and the specific value of it depends on the specific encryption algorithm used; and the deduplication key k_{dedup} is encrypted by using the user key ku to obtain the secret deduplication key C_{key}; the file storage directory dir, filename $filename$, secret file data C_{data} and secret deduplication key C_{key} are sent to the storage service provider. If the storage service provider finds that there is no duplicate data C_{data}, it names the secret file data C_{data} and secret deduplication key C_{key} with $filename.data$ and $filename.key$ respectively, stores them in the directory dir, informs user A that file m does not contain duplicate files, and records the file path in the database.

After the user A receives the notification of no duplicate files, user A encodes the block data in the secret file data C_{data} to the group Z_p, further calculates the integrity tag $\sigma_i = H_1(ma||i)^{\omega} \cdot (u^{m_i})^{\frac{1}{x+H_2(ma||h)}}$ of the block data, calculates the user tag conversion auxiliary variable $trans$, and then obtains the auxiliary variable C_{trans}; finally, the user A sends $\{\sigma_1, \cdots, \sigma_n\}$, auxiliary variable C_{trans} and audit materials TAM to the storage service provider.

User B uploads the files that may be deduplication encrypted to the storage service provider. This part of the process is the same as user A uploads the files that may be deduplication encrypted to the storage service provider. However, when the storage service provider detects duplicate files, it only stores C_{key} in the directory dir named $filename.key$, and records the actual storage path $dir'/filename'.data$ of duplicate files in the database. when the next time $dir/filename.data$ is retrieved, it will be redirected to $dir'/filename'.data$.

When the storage service provider notifies user B that there are duplicate files, it also returns user A's user tag transform auxiliary material C_{trans}. When user B obtains the user tag transform auxiliary material C_{trans}, the deduplication key k_{dedup} is used to parse and decrypt ω、$trans$、v , $Trans' = \omega'||trans'||V$ is calculated, and finally the audit materials *TAM'* 'and *Trans'* are sent to the storage service provider; the storage service provider stores *TAM'* and *Trans'*.

In the audit stage, as shown in FIG. 3, firstly, the auditor sends the file name *filename* to be audited to the storage service provider, and the storage service provider returns the number n of blocks of the file according to the *filename*. The auditor generates a challenge $Q \in \{(i_1, v_{i_1}), \dots, (i_c, v_{i_c})\}_{i_j \in [1,n]}$ according to n, where c is the total number of challenged data blocks, and then the challenge Q and the file name *filename* are sent to the storage service provider.

Then, the storage service provider retrieves the secret file data C_{data} according to the *filename*, and indexes the corresponding data block according to the subscript set in the challenge Q. At this time, there are two situations: in the first case, the auditor helps the user A (that is, the first storage of the file) to audit, and the calculation proves that $\bar{\rho}$, $\bar{\sigma} = \prod_{(i,v_i)\in Q} {\sigma_i}^{v_i}$; in the second case, the auditor helps the user B to audit, and then the calculation proves that $\bar{\rho}$, $\bar{\sigma} = \prod_{(i,v_i)\in Q} \left(\sigma_i \cdot H_1(ma||i)^W \cdot (u^{m_i})^{trans'} \cdot V^{m_i}\right)^{v_i}$. Finally, $\bar{\rho}$, $\bar{\sigma}$ and the audit material *TAM* or *TAM'* that audits the user tags are returned to the auditor.

Finally, if the auditor receives the audit material as TAM, the following formula may be verified by using the corresponding audit public key:

$$e\left(\bar{\sigma}, pk_A \cdot g^{H_2(ma||h)}\right) = e\left(\prod_{(i,v_i)\in Q}[H_1(ma||i)]^{v_i},\ h_1 \cdot h^{H_2(ma||h)}\right) \cdot e(u^{\bar{\rho}}, g)$$

If the formula is valid, the challenge is passed; if it is not, the challenge fails, indicating that the integrity of the document has been damaged.

If the received audit material is TAM', the following formula is verified whether it is valid:

$$e\left(\bar{\sigma}, pk_B \cdot g^{H_2(ma||h')}\right) = e\left(\prod_{(i,v_i)\in Q}[H_1(ma||i)]^{v_i},\ h_1' \cdot (h')^{H_2(ma||h')}\right) \cdot e(u^{\bar{\rho}}, g)$$

Embodiment 3

In the Embodiment 3, as shown in FIG. 4, a lightweight duplicates-removing cryptograph integrity audit system, including: a private key generation module, an encryption module, a judgment module and a verification module.

The private key generation module is used for selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key; in this embodiment, the user randomly selects the number *ku* as the user private key, randomly selects a number *x* as the audit private key, selects a generator *g* from a cyclic multiplication group $\mathbb{G}$, and calculates g^x as the audit public key.

The encryption module is used for calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server; in this embodiment, the hash value $h(m)$ of the message plaintext is calculated as the deduplication key and recorded as k_{dedup}; the original file data is encrypted, cut and encoded by using the deduplication key k_{dedup}, and then the secret file data C_{data} is obtained. The user private key ku is used as the encryption key to encrypt the deduplication key k_{dedup} to obtain the secret deduplication key C_{key}; the file directory dir, file name $filename$, secret file data C_{data} and secret key C_{key} are uploaded to the server. The file directory is created according to dir, and C_{key} is stored in the corresponding directory and named with $filename.key$.

The judging module is used for judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials, and uploading them to a storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; in this embodiment, if there is no secret file data C_{data} in the server, the user is informed that there is no duplicate file; after the user receives the notification that there is no duplication, the user sequentially calculates the tag σ_i for the block data in C_{data}, generates the file digest ma, calculates the user tag conversion auxiliary variable $trans$ by using the audit private key x and the file digest ma, and then sends all the block tags σ_i, the auxiliary variable C_{trans} encrypted with the key parameter r and the auditor's audit materials TAM to the storage service provider; if the secret file data exists, the user is informed that the file may be deduplicated and C_{trans} is returned to the user. The user decrypts the encrypted variables in C_{trans} by the deduplication key k_{dedup} to obtain $trans$, calculates the user tag transform auxiliary material $trans'$ through $trans$, and then sends the user tag transform auxiliary material $trans'$ and auditor's audit material TAM' to the storage service provider.

The verification module is used for sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result. In this embodiment, when the auditor wants to audit data, it is necessary to inform the storage service provider of the filename $filename$ of the audit target file and the total n number of data blocks of the storage service provider's response, and the auditor randomly generates a block subscript set and a random number set $Q = \{i, v_i\}_{i\in[1,n]}$ by using the total number of data blocks; the storage service provider generates a data integrity certificate $(\bar{\rho}, \bar{\sigma})$ according to the block subscript set and the random number set Q and the stored corresponding user secret file data C_{data}, and then returns the auditor's audit material TAM and the integrity certificate $(\bar{\rho}, \bar{\sigma})$ to the user. At last, the auditor uses the audit public key g^x and the audit material TAM to verify the integrity certificate. If it passes the test, it means that the sampled data is complete; otherwise, it means that the sampled data has been destroyed.

The above-mentioned embodiments only describe the preferred mode of the present application, and do not limit the scope of the present application. Without departing from the design spirit of the present application, all kinds of modifications and improvements made by ordinary technicians in the field to the technical scheme of the present application should fall within the protection scope determined by the claims of the present application.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS

1. A lightweight duplicates-removing cryptograph integrity audit method, characterized by comprising:

selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key;

calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server;

judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials and uploading them to the storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; and

sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

2. The lightweight duplicates-removing cryptograph integrity audit method according to claim 1, characterized in that, the calculation method of the audit public

key comprises: randomly selecting a number x as the audit private key, selecting a generator g from a cyclic multiplication group $\mathbb{G}$, and calculating g^x as the audit public key.

3. The lightweight duplicates-removing cryptograph integrity audit method according to claim 1, characterized in that, the method for calculating the deduplication key comprises: calculating a hash value of the message plaintext as the deduplication key.

4. The lightweight duplicates-removing cryptograph integrity audit method according to claim 1, characterized in that, the method of the first encryption processing comprises: encrypting, cutting and coding the original file based on the deduplication key.

5. The lightweight duplicates-removing cryptograph integrity audit method according to claim 1, characterized in that, the first operation comprises:

calculating the block tag based on the block data in the secret file data;

generating a file digest, and calculating the user tag conversion auxiliary variables based on the audit private key and the file digest; and

encrypting key parameters based on the user tag conversion auxiliary variables to obtain the auxiliary variables.

6. The lightweight duplicates-removing cryptograph integrity audit method according to claim 5, characterized in that, the second operation comprises:

decrypting encrypted variables in the auxiliary variables based on the deduplication key to obtain the user tag conversion auxiliary variables; and calculating based on the user tag conversion auxiliary variables to obtain the user tag transform auxiliary materials.

7. The lightweight duplicates-removing cryptograph integrity audit method according to claim 1, characterized in that, a generation method of the integrity certificate comprises:

generating a block subscript set and a random number set based on the total data block; and

generating the integrity certificate based on the block subscript set, the random number set and the secret file data.

8. A lightweight duplicates-removing cryptograph integrity audit system, characterized by comprising: a private key generation module, an encryption module, a judgment module and a verification module;

the private key generation module is used for selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key;

the encryption module is used for calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server;

the judging module is used for judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials, and uploading them to a storage server; if yes, performing a second operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; and

the verification module is used for sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

FIGURES

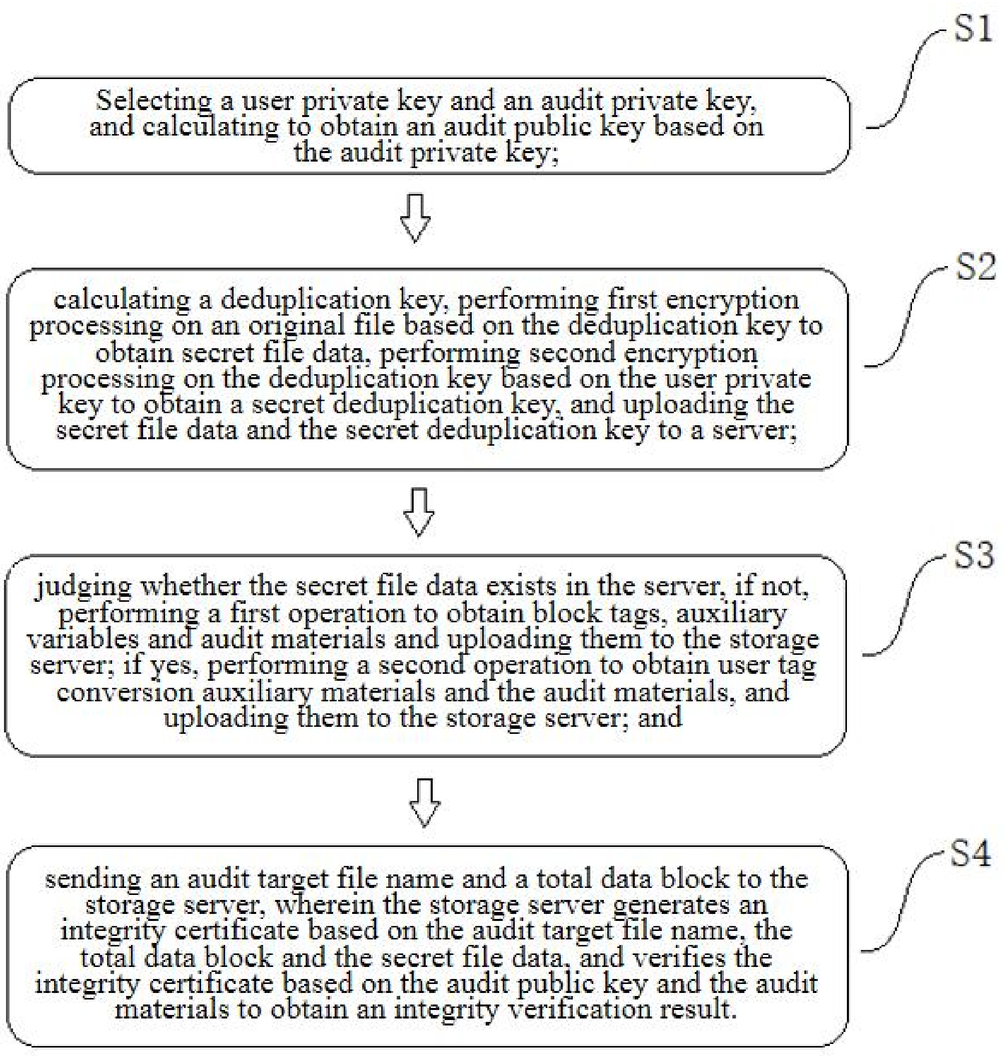
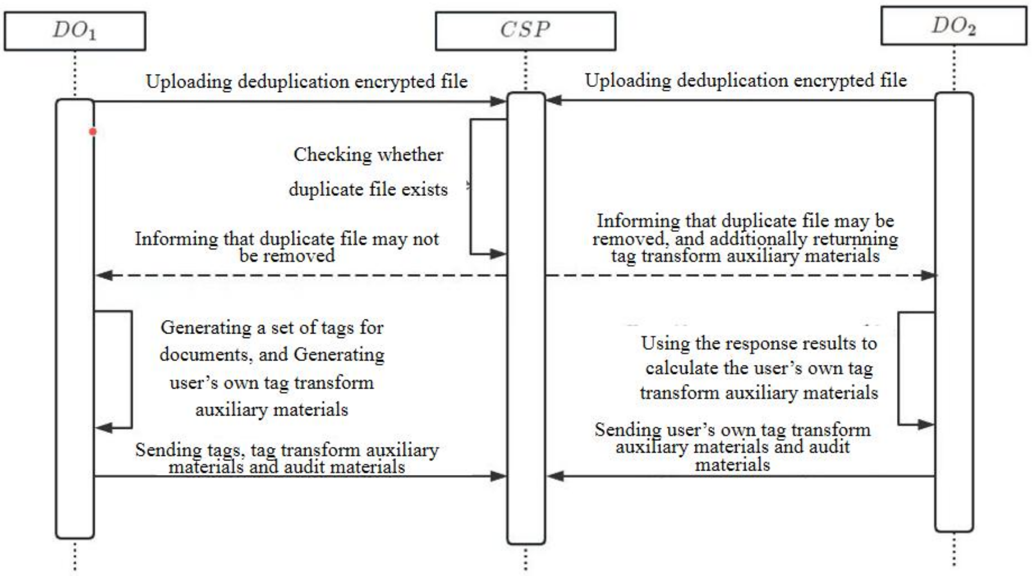
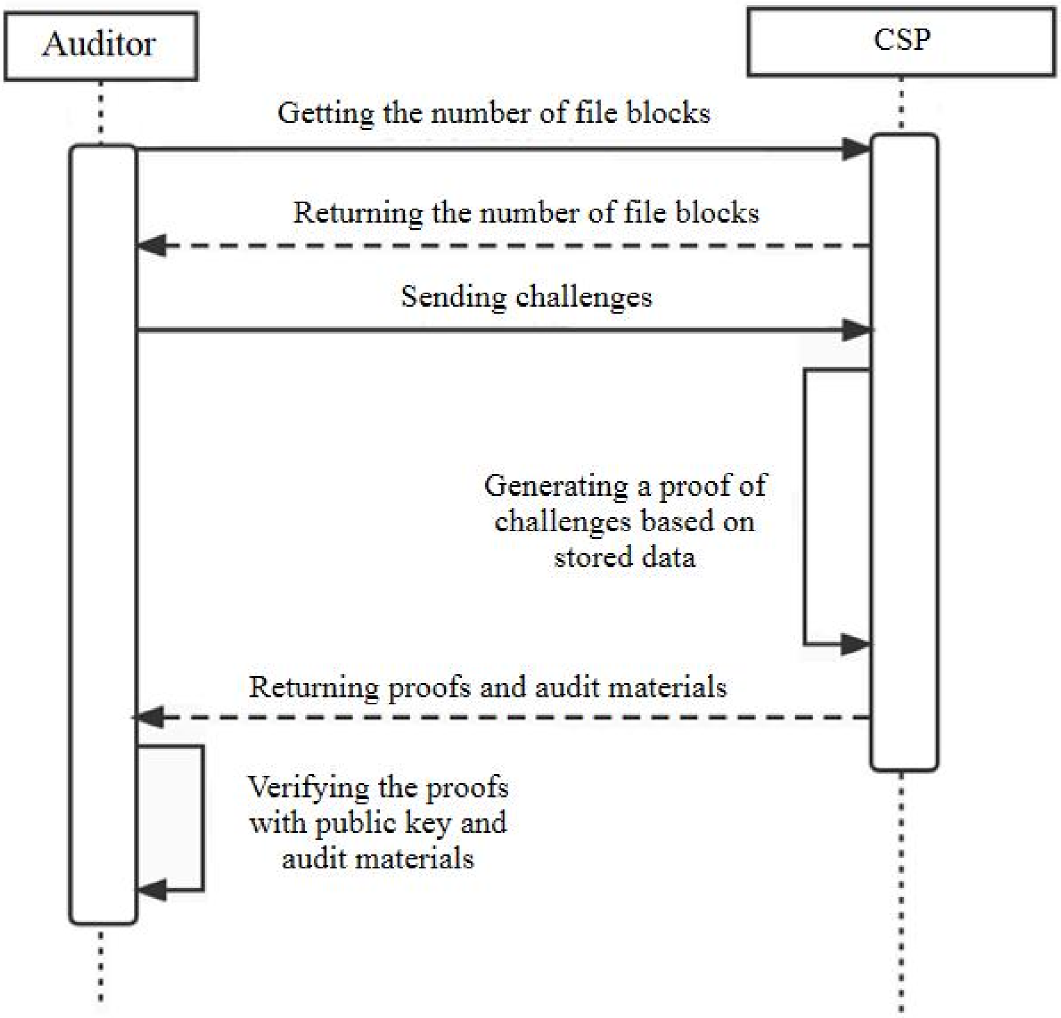

S1
Selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key;
S2
calculating a deduplication key, performing first encryption processing on an original file based on the deduplication key to obtain secret file data, performing second encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server;
S3
judging whether the secret file data exists in the server, if not, performing a first operation to obtain block tags, auxiliary variables and audit materials and uploading them to the storage server; if yes, performing a second operation to obtain user tag conversion auxiliary materials and the audit materials, and uploading them to the storage server; and
S4
sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials to obtain an integrity verification result.

FIG. 1



DO_1
CSP
DO_2
Uploading deduplication encrypted file
Uploading deduplication encrypted file
Checking whether duplicate file exists
Informing that duplicate file may not be removed
Informing that duplicate file may be removed, and additionally returnning tag transform auxiliary materials
Generating a set of tags for documents, and Generating user's own tag transform auxiliary materials
Using the response results to calculate the user's own tag transform auxiliary materials
Sending user's own tag transform auxiliary materials and audit materials
Sending tags, tag transform auxiliary materials and audit materials

FIG. 2



Auditor
CSP
Getting the number of file blocks
Returning the number of file blocks
Sending challenges
Generating a proof of challenges based on stored data
Returning proofs and audit materials
Verifying the proofs with public key and audit materials

FIG. 3

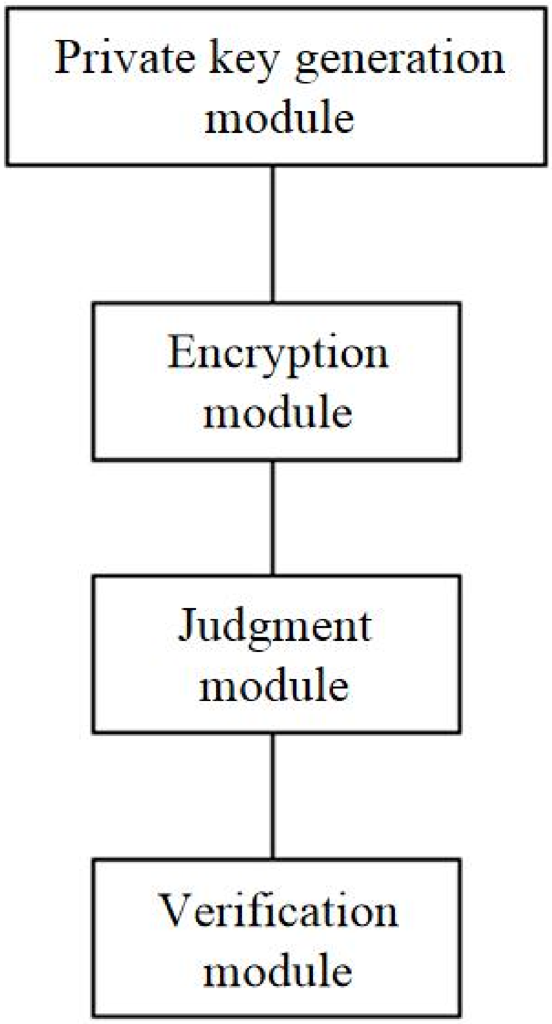

Private key generation module
Encryption module
Judgment module
Verification module

FIG. 4

ABSTRACT

Disclosed is a lightweight duplicates-removing cryptograph integrity audit method and a system. The method includes: selecting a user private key and an audit private key, and calculating to obtain an audit public key based on the audit private key; calculating a deduplication key, performing encryption processing on an original file based on the deduplication key to obtain secret file data, performing encryption processing on the deduplication key based on the user private key to obtain a secret deduplication key, and uploading the secret file data and the secret deduplication key to a server; judging whether the secret file data exists in the server, if not, performing an operation to obtain block tags, auxiliary variables and audit materials and uploading them to the storage server; if yes, performing another operation to obtain user tag transform auxiliary materials and the audit materials, and uploading them to the storage server; and sending an audit target file name and a total data block to the storage server, wherein the storage server generates an integrity certificate based on the audit target file name, the total data block and the secret file data, and verifies the integrity certificate based on the audit public key and the audit materials.