Chinese Journal of Electronics
Vol.32, No.2, Mar. 2023

Remote Data Auditing for Cloud-Assisted WBANs with Pay-as-You-Go Business Model

LI Yumei[1] and ZHANG Futai[2]

(1. *School of Computer Science, Hubei University of Technology, Wuhan 430068, China*)

(2. *Fujian Provincial Key Laboratory of Network Security and Cryptology, College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China*)

Abstract — As an emerging technology, cloud-assisted wireless body area networks (WBANs) provide more convenient services to users. Recently, many remote data auditing protocols have been proposed to ensure the data integrity and authenticity when data owners outsourced their data to the cloud. However, most of them cannot check data integrity periodically according to the pay-as-you-go business model. These protocols also need high tag generation computation overhead, which brings a heavy burden for data owners. Therefore, we construct a lightweight remote data auditing protocol to overcome all above drawbacks. Our work can be deployed in the public environment without secret channels. It makes use of certificate-based cryptography which gets rid of certificate management problems, key escrow problems, and secret channels. The security analysis illustrates that the proposed protocol is secure. Moreover, the performance evaluation implies that our work is available in cutting down computation and communication overheads.

Key words — Cloud-assisted wireless body area networks, Periodic auditing, Certificate-based cryptography, Pay-as-you-go business model, Public environment.

I. Introduction

Wireless body area networks (WBANs) are often used to improve the quality of medical treatment and guarantee the human health-care service [1]. It relies on all kinds of sensors to collect medical data and fulfill remote vital signs monitoring of patients. The scale of medical data grows over time, and the storage burden will lead the device to be inefficient. Cloud computing as an auxiliary means provides flexible storage capability and cheap services for data owner. Cloud-assisted WBANs overcome the inherent weaknesses of traditional WBANs and enable the data owner to store and process the collected data conveniently [2], [3]. However, it faces kinds of internal attackers and external attackers. A dishonest cloud service provider (CSP) can mask medical data corruption or data loss to maintain an excellent reputation. Even worse, a malicious adversary can disturb diagnostic results through falsifying some medical data. The incorrect diagnostic results may delay the treatment of patients and cause serious medical incidents. Among these security issues, the integrity auditing of outsourced data is crucial.

Downloading the entire file is the intuitive method to check the integrity of data [4]. However, this is not practical because the solution is inefficient. Remote data auditing is a popular model that allows a party to check data integrity without downloading all data contents [5]. It generates probabilistic proof by sampling random sets of data blocks. Recently, scholars have proposed many schemes to check outsourced data integrity [6]–[12], each has its pros and cons. A common weakness of these schemes is that they only support content integrity checking. It is not applicable to the CSP which using the pay-as-you-go business model [13]. In pay-as-you-go business model, a data owner pays fees for the data in each period based on the storage volume indiscriminately. Therefore, a third party auditor (TPA) should have the ability to check the data integrity and authenticity periodically.

In the pay-as-you-go model, data owners only need to pay for uncorrupted files according to actual storage volume. The CSP charges the storage fee based on the data storage conditions. As shown in Fig.1, storage fee

Manuscript Received Sept. 24, 2020; Accepted Dec. 14, 2021. This work was supported by the National Natural Science Foundation of China (62172096).

should comply with the following principles: 1) The data owner pays the storage fee for each period in a regular way if the file keeps intact; 2) If any data error is detected in auditing phase, the data owner will not pay the storage fee and the CSP should compensate for the damaged file; 3) If the data owner removes a file from the CSP, he/she pays the storage fee to CSP on demand by this date [14]. A remote data auditing (RDA) protocol which satisfies the above application should support integrity auditing of both content and time of storage (i.e., timestamp). An obvious solution is to attach a timestamp at the end of the outsourced data to mark the storage time. However, a weakness is that the timestamp may be lost or corrupted as there is no relationship between the timestamp and individual data blocks. The solution that an authentication tag generated for each data block includes the timestamp is efficient. It ensures a strong binding between the timestamp and individual data block.

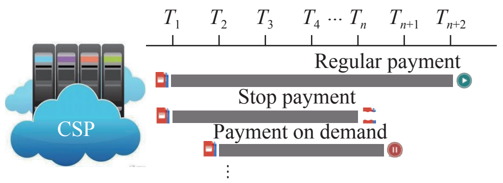

Fig. 1. Pay-as-you-go model integrated with RDA.

Moreover, there exists another neglected problem that a data owner cannot prove the file has been uploaded to CSP. If a file is lost, the CSP may deny the fact that the file was stored in CSP. Most of existing auditing protocols only focus on partial data damage and loss. The case that the entire file was erased by the CSP is ignored. In this situation, the CSP can claim that it never received the file from the data owner. An effective way to solve the problem is that the CSP returns an unforgeable voucher to the data owner while receiving complete file. Using this method, there is no dispute that the CSP should undertake the obligations for all data damage and loss. Besides, the voucher should be updated by the CSP if the file keeps intact in the last period. It also indicates that the data owner has paid the storage fee for the previous period.

In this paper, we design an efficient certificate-based remote data auditing protocol. Considering the computing power of WBANs, one of the main motivations of our work is to reduce the computation cost in tag generation. In the real world, the storage fee is one of the important factors for data owners to select a service provider. To increase competitiveness, it is necessary to reduce the size of relevant auditing information. Besides, a certificate-based cryptosystem (CBC) is preferable in cloud-assisted WBANs. It can be deployed in public channels without a honest and trusted third party.

1. Our contributions

We construct a practical remote data auditing protocol for cloud-assisted WBANs with pay-as-you-go business model, which can audit data integrity regularly. The innovations of this paper are as follows.

1) We design a novel auditing model that the TPA will audit data integrity spontaneously periodically according to the auditing period. Besides, the model can prevent CSP from hiding data damaged or lost to evade compensation.

2) We construct a homomorphic verifiable tag, which has low tag generation overhead and verification overhead. Besides, the size of a data block tag is short.

3) We put forward an efficient remote data auditing protocol according to pay-as-you-go business model. It is a valuable application of CBC in checking data integrity.

Moreover, we show the correctness of our protocol and provide rigorous security proof under the random oracle model. We also compare the overhead of our protocol with several other related protocols in computation, communication, and storage. Besides, we show the performance of our work is desirable through experiments.

2. Related work

All sensors in WBANs are usually assigned to collect and monitor users' physical information [15]. However, low storage power and computing power limit the development of WBANs. With the development of cloud computing, the CSP can assist users in storing these sensors' data. The integrity and authenticity of data stored in the CSP is widely concerned by users. The technique of proof of storage (PoS) allows a verifier to check the data integrity without holding a local copy [16]. It offers an optional solution for auditing outsourced data integrity [4].

There are two interesting PoS models, namely proof of retrievability (PoR) and provable data possession (PDP). The notion of the former was presented by Juels *et al.* [17] in 2007. In this protocol, the data owner can retrieve the entire file. The notion of PDP was first proposed by Ateniese *et al.* [5] in the same year. They proposed two different PDP protocols which based on RSA cryptosystem and homomorphic verifiable tags (HVTs). The two protocols can both randomly choose some data blocks to detect files, and entire files are not required. To reduce the computation and communication overheads, Shacham and Waters [18] constructed a novel HVTs using BLS signature [19]. Later, various PDP protocols based on the construction has been proposed [20]–[30]. Yu *et al.* [21] proposed an identity-based remote data integrity verification protocol which reaches the perfect data privacy-preserving. Although the protocol is shaken off the burden of certificate management, the verification overhead increases linearly with the number of the challenged data blocks. Zhang *et al.* [29] introduced an identity-based cloud storage auditing protocol for shared big data with efficient user revocation. Li *et al.* [30] proposed a certificateless public data integrity checking protocol for data shared among a group. He *et al.* [26] presented a certificateless public auditing protocol for cloud-assisted WBANs. Their protocol does not suffer from public key certificate management and key escrow problem. Huang *et al.* [31] proposed a certificateless public verification scheme for data storage and sharing in the cloud. However, the storage space occupied by the data block tag in these protocols is larger than the data block itself.

In 2013, Wang *et al.* [7] assumed that each block consisting of n sectors in their protocol and comprises n sectors' signatures into one using homomorphism. That is to say, the size of a data block is $(n \times q)$-bits and the size of its corresponding signature is q-bits. It also achieves the shortest query for a challenge. Wang *et al.* [32] proposed a remote integrity auditing protocol, which not only permits checking data content, but also allows checking the log information about the origin, type, and consistency of the data. Yan *et al.* [33] introduced a remote data possession checking protocol which supports dynamic data. Wu *et al.* [14] presented two protocols for pay-as-you-go business model which allows the data owner or the TPA to verify the integrity of data content and its timestamp. The protocol leaks no information on data content and timestamp to TPA. Thokchom *et al.* [34] proposed a data checking protocol for storage of shared dynamic data in untrusted CSP with privacy-preserving and revocation of users. However, there still exists an issue that the tag generation cost increases as the size of the file grows. Besides, the CSP has to store at least additional $(n \times q)$-bits related information for file auditing.

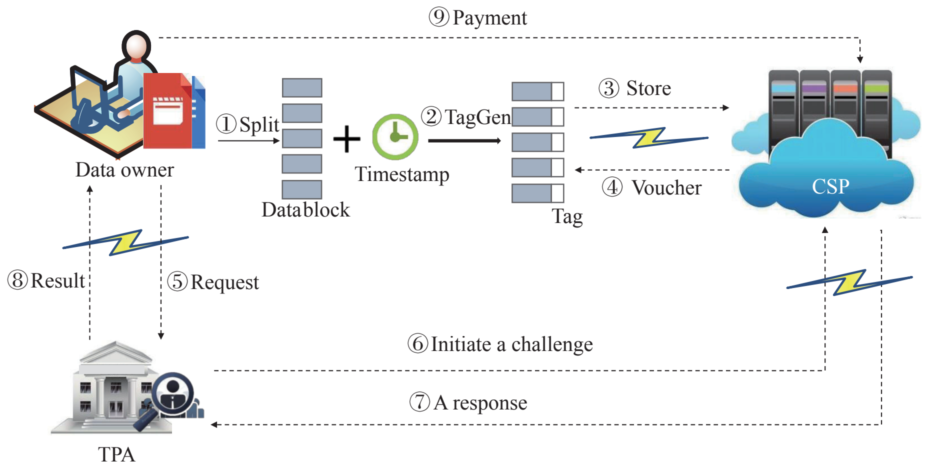

Fig. 2. The remote data auditing system.

II. System Model and Design Goals

1. System model

There are three types of parties called the data owner, the CSP, and the TPA in a remote data auditing system. The CSP is dishonest, it may generate forged proofs that can pass the verification. The TPA is honest, it performs the data auditing periodically on behalf of the data owner. Fig.2 shows the system model and we give the working process below.

1) The data owner splits a file into data blocks, and generates tags for all data blocks and timestamp. The data owner then uploads these data blocks with the corresponding tags to the CSP.

2) The CSP will store these information if the file is correct and return a voucher to the data owner. The voucher indicates that the data is intact at this time. Note that, the CSP will update and return the voucher after receiving the storage fee.

3) The TPA initiates a challenge to the CSP for file auditing at the end of each period. The CSP generates a proof and sends it to TPA.

4) The TPA checks the correctness of the proof and returns a result to the data owner. The data owner pays the storage fee according to the pay-as-you-go model.

2. Design goals

In a practical remote data auditing protocol for cloud-assisted WBANs with a pay-as-you-go business model, the following objectives are required.

1) Correctness: It is possible to generate valid proofs if and only if the CSP possesses the original file and timestamp.

2) Verifiability: The TPA can check the file integrity using partial data blocks without accessing the original file.

3) Periodic auditing: The TPA can audit data integrity spontaneously according to the timestamp and auditing period.

4) Accountability: There is no dispute that the CSP is the responsible party if any error is detected.

5) User friendly: The storage space occupied by data block tags should be smaller than the data block itself. The data owner pays as few storage fees as possible for secure storage without hindering data integrity auditing.

III. Preliminaries and Definition

1. Mathematic background

Definition 1 (Bilinear map) Given three prime order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with q elements, and for all $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map if the following properties hold:

1) Bilinearity: The equation $e(g^a, h^b) = e(g, h)^{ab}$ holds if a, b are randomly choosed in $\mathbb{Z}_q^*$.

2) Non-degeneracy: $e(g, h) \neq 1$ for some g, h.

3) Computability: There exists an efficient algorithm to compute $e(g, h)$.

Definition 2 (Collusion attack algorithm with k traitors (k-CAA) problem assumption) For an integer $x \in \mathbb{Z}_q^*$, given $\{g \in \mathbb{G}_1, h^x \in \mathbb{G}_2, h_1, \dots, h_k \in \mathbb{Z}_q^*, g^{\frac{1}{x+h_1}}, \dots, g^{\frac{1}{x+h_k}}\}$, to compute $g^{\frac{1}{x+h^*}}$ when $h^* \notin \{h_1, \dots, h_k\}$.

There is no algorithm to solve the k-CAA problem with a non-negligible advantage in probabilistic polynomial-time.

Definition 3 (Modified k-CAA problem assumption) For three integers $x, a, b \in \mathbb{Z}_q^*$, given $\{g, g^a \in \mathbb{G}_1, h^x, h^b \in \mathbb{G}_2, h_1, \dots, h_k \in \mathbb{Z}_q^*, g^{\frac{ab}{x+h_1}}, \dots, g^{\frac{ab}{x+h_k}}\}$, to compute $g^{\frac{ab}{x+h^*}}$ when $h^* \notin \{h_1, \dots, h_k\}$ or g^{ab}.

There is no algorithm to solve the modified k-CAA problem with a non-negligible advantage in probabilistic polynomial-time.

2. Formal definition

We define a certificate-based remote data auditing protocol (CB-RDAP), which consists of eight polynomial-time algorithms.

1) Setup: The CSP takes input a security parameter 1^λ and outputs (pp, msk), where the parameter pp is published in system and the master private key msk is only known by the CSP.

2) UserKeyGen: The user takes input (pp, ID) and outputs a user's public/private key pair (upk_{ID}, usk_{ID}), where ID denotes a user's identity.

3) Certify: The CSP takes input (pp, msk, ID, upk_{ID}) and outputs the corresponding certificate $Cert_{ID}$.

4) TagGen: The user takes input $(pp, ID, usk_{ID}, Cert_{ID}, fname, t, F)$ and outputs a verifiable label τ and a file's signatures $\{\sigma_i\}$, where $fname$ denotes the filename, $F = (\boldsymbol{m}_1, \dots, \boldsymbol{m}_m)$ denotes a file including m data blocks, and t denotes the timestamp.

5) Confirm: The CSP takes input (pp, msk, ID, τ) and outputs the auditing period T and a voucher π if the file keeps intact, outputs "failure" otherwise.

6) Challenge: The TPA takes input (pp, ID, τ, T) and outputs a challenge $chal$ periodically according to T.

7) ProofGen: The CSP takes input $(pp, ID, \tau, \{\sigma_i\}, chal)$ and outputs a possession proof PF.

8) ProofCheck: The TPA takes input $(pp, ID, upk_{ID}, \tau, chal, PF)$ and outputs 1 if PF passes the verification, otherwise outputs 0.

Correctness: For any

$$\begin{cases} (pp, msk) \leftarrow \text{Setup}(1^\lambda) \\ (usk_{ID}, upk_{ID}) \leftarrow \text{UserKeyGen}(pp, ID) \\ Cert_{ID} \leftarrow \text{Certify}(pp, msk, ID, upk_{ID}) \end{cases}$$

if $(\tau, \{\sigma_i\}) \leftarrow \text{TagGen}(pp, ID, usk_{ID}, Cert_{ID}, fname, t, \{\boldsymbol{m}_i\})$, $chal \leftarrow \text{Challenge}(pp, ID, \tau, T)$ and $PF \leftarrow \text{ProofGen}(pp, ID, \tau, \{\sigma_i\}, chal)$, then $1 \leftarrow \text{ProofCheck}(pp, ID, upk_{ID}, \tau, chal, PF)$.

A CB-RDAP is secure if it meets the following requirements:

1) If the challenged file stored in CSP is intact and PF is generated by $\text{ProofGen}(pp, ID, \tau, \{\sigma_i\}, chal)$ honestly, the probability of $\text{ProofCheck}(pp, ID, upk_{ID}, \tau, chal, PF) = 1$ is 1.

2) If the challenged file is damaged or deleted, the probability of the CSP could forge a valid proof PF is negligible.

3) The CSP cannot deny it has received a file from the data owner successfully if the voucher π is generated by the algorithm $\text{Confirm}(pp, msk, ID, \tau)$.

3. Security model

To ensure the correctness and integrity of the data, a secure CB-RDAP should resist the following attacks: 1) The third party (CSP or system users) can forge a tag of data block. 2) The CSP can replace the new challenge response with the expired valid proof to deceive the data owner. 3) The CSP can generate a valid proof PF by using non-challenging data blocks and tags.

In a secure CB-RDAP, three types of adversaries that can be involved to cover these attacks. The Type I adversary models system user's ability to forge data block tag. It can change the public key for some users, and the target user's certificate keeps secret from the adversary. The Type II adversary who plays the CSP has the ability to forge data block tags. It holds the master secret key and does not have permission to substitute the target user's public key. The Type III adversary models the ability of CSP to forge a valid proof, it attempts to generate a valid proof when some data blocks are damaged. Generally, the following oracles are provided for adversaries.

- User-key-gen Oracle. The adversary sends a user's identity ID to the oracle. The oracle generates the user's public/private key (upk_{ID}, usk_{ID}) by running the algorithm UserKeyGen and returns this public/private key to the adversary.
- Corruption Oracle. On input a user's identity ID, the oracle outputs the corresponding private key usk_{ID} if ID has been queried a User-key-gen oracle. Otherwise nothing is returned.
- Certification Oracle. The adversary sends a user's identity ID and its public key upk_{ID} to the oracle. The oracle will run the algorithm Certify to obtain $Cert_{ID}$ and return it to the adversary.
- Key-replace Oracle. The adversary provides a user identity ID and a novel public/private key pair $(upk_{ID}^{'}, usk_{ID}^{'})$ to the oracle. The oracle records the newly public/private key pair.
- TagGen Oracle. It inputs a user's identity ID, the filename $fname$ with the timestamp t and data block $\boldsymbol{m} \in F$. The oracle outputs the data block's tag σ.
- ProofCheck Oracle. The adversary generates a proof PF, and returns PF and the corresponding challenge $chal$ to the oracle. The oracle outputs 0 or 1.

We define the security model of CB-RDAP by the game (Game 1, Game 2, Game 3) between the adversary $\mathcal{A}(\mathcal{A}_{\rm I}, \mathcal{A}_{\rm II}, \mathcal{A}_{\rm III})$ and the challenger $\mathcal{C}$. Besides, we claim that the advantage of $\mathcal{A}$ wins the game is the probability of $\mathcal{A}$ breaks the scheme.

1) Game 1 (Type I adversary $\mathcal{A}_{\rm I}$):

Initialization: Taking a security parameter 1^λ as input, the challenger returns the public parameter pp to $\mathcal{A}_{\rm I}$ by running Setup(1^λ) and keeps system master private key msk secret.

Query: $\mathcal{A}_{\rm I}$ is allowed to make the query to User-key-gen oracle, Corruption oracle, Certification oracle, Key-replace oracle, and TagGen oracle adaptively.

Forge: $\mathcal{A}_{\rm I}$ outputs $(ID^*, upk_{ID}^*, \tau^*, \boldsymbol{m}^* \in F, \sigma^*)$. $\mathcal{A}_{\rm I}$ wins the game if the following conditions hold,

a) $\mathcal{A}_I$ has never queried the certificate of ID^*.

b) ID^* has never been launched Corruption query by the adversary $\mathcal{A}_{\rm I}$.

c) $(ID^*, fname^*, \boldsymbol{m}^*)$ has never been asked the tag by the adversary $\mathcal{A}_{\rm I}$.

d) ProofCheck$(pp, ID^*, upk_{ID}^*, \tau^*, \perp, \boldsymbol{m}^*, \sigma^*)$=1.

2) Game 2 (Type II adversary $\mathcal{A}_{\rm II}$):

Initialization: Taking a security parameter 1^λ as input, the challenger returns pp and msk to $\mathcal{A}_{\rm II}$ by running Setup(1^λ).

Query: $\mathcal{A}_{\rm II}$ can adaptively make the User-key-gen oracle, Corruption oracle, and TagGen oracle.

Forge: $\mathcal{A}_{\rm II}$ outputs $(ID^*, upk_{ID}^*, \tau^*, \boldsymbol{m}^* \in F, \sigma^*)$. $\mathcal{A}_{\rm II}$ wins the game if the following conditions hold,

a) ID^* has never been launched Corruption query by the adversary $\mathcal{A}_{\rm II}$.

b) $(ID^*, fname^*, \boldsymbol{m}^*)$ has never been asked the tag by the adversary $\mathcal{A}_{\rm II}$.

c) ProofCheck$(pp, ID^*, upk_{ID}^*, \tau^*, \perp, \boldsymbol{m}^*, \sigma^*)$=1.

3) Game 3 (Type III adversary $\mathcal{A}_{\rm III}$):

Initialization: Taking a security parameter 1^λ as input, the challenger returns the public parameter pp to $\mathcal{A}_{\rm III}$.

Query: $\mathcal{A}_{\rm III}$ can adaptively make TagGen oracle, and ProofCheck oracle.

Challenge: $\mathcal{C}$ generates a challenge $chal$ and sends it to $\mathcal{A}_{\rm III}$. On receiving $chal$, $\mathcal{A}_{\rm III}$ computes a proof PF and sends it to $\mathcal{C}$.

Forge: $\mathcal{A}_{\rm III}$ outputs $(ID^*, upk_{ID}^*, \tau^*, chal^*, PF^*)$. $\mathcal{A}_{\rm III}$ wins the game if the following coditions hold,

a) ProofCheck$(ID^*, upk^*, \tau^*, chal, PF) = 1$, where $PF = (\tilde{\boldsymbol{m}} \notin F, \tilde{\sigma})$.

b) There is at least a challenged data block has never been submitted to the TagGen query.

Definition 4 A CB-RDAP is secure if the advantage of the probabilistic polynomial time adversaries $\mathcal{A}_{\rm I}, \mathcal{A}_{\rm II}, \mathcal{A}_{\rm III}$ win Game 1, Game 2, Game 3 is negligible, respectively.

IV. The Proposed Protocol

1. Construction

The construction of linear homomorphic verifiable tags in our protocol is inspired by the work of Ateniese *et al.* [5] and Shacham *et al.* [18]. Our goal is to reduce the signature generation cost and the size of the signature over the file. We describe algorithms of our protocol as follows and present the workflow in Fig.3.

1) Setup: The CSP takes input a security parameter 1^λ, this algorithm selects three cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ with prime order q (the size of q is determined by λ), a generator $g \in \mathbb{G}_1$, a generator $h \in \mathbb{G}_2$, and a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. The CSP then chooses four collision-resistant hash functions $H_1: \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_2: \{0,1\}^* \rightarrow \mathbb{Z}_q^*$, $H_3: \{0,1\}^* \rightarrow \mathbb{G}_1$, $H_4: \{0,1\}^* \rightarrow \mathbb{Z}_q^*$. The CSP sets the system master private key $msk = s$ and the system public key $mpk = h^s$, where s is randomly selected in $\mathbb{Z}_q^*$. The CSP outputs the system public parameters $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h, mpk, H_1, H_2, H_3, H_4)$ and stores the system master private key $msk = s$ secretly.

2) UserKeyGen: The user takes input parameters

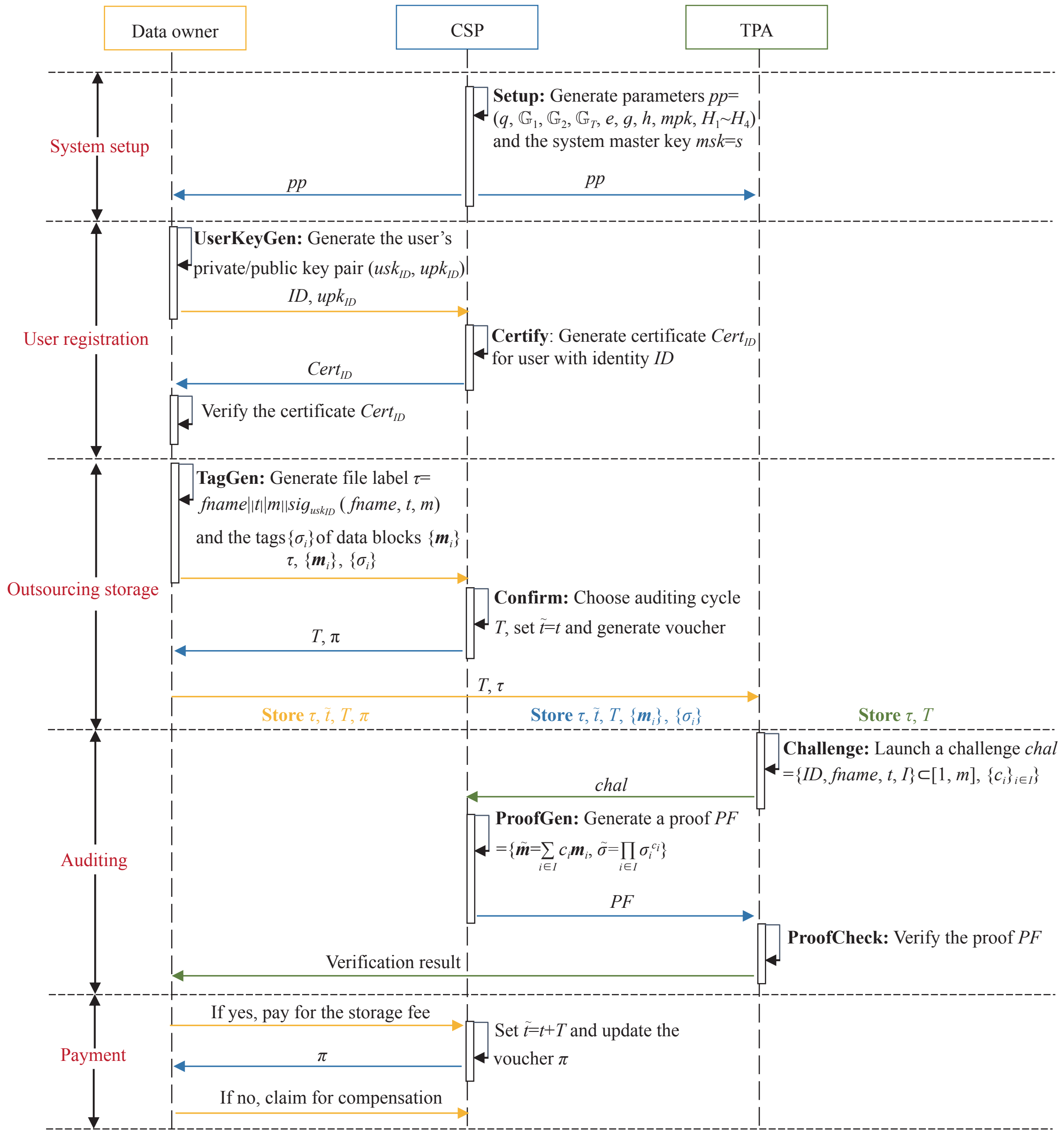
Data owner
CSP
TPA
System setup
Setup: Generate parameters pp=(q, 𝔾1, 𝔾2, 𝔾T, e, g, h, mpk, H1~H4) and the system master key msk=s
pp
pp
User registration
UserKeyGen: Generate the user's private/public key pair (uskID, upkID)
ID, upkID
Certify: Generate certificate CertID for user with identity ID
CertID
Verify the certificate CertID
Outsourcing storage
TagGen: Generate file label τ=fname||t||m||siguskID (fname, t, m) and the tags {σi} of data blocks {mi}
τ, {mi}, {σi}
Confirm: Choose auditing cycle T, set t̃=t and generate voucher
T, π
T, τ
Store τ, t̃, T, π
Store τ, t̃, T, {mi}, {σi}
Store τ, T
Auditing
Challenge: Launch a challenge chal ={ID, fname, t, I}⊂[1, m], {ci}i∈I}
chal
ProofGen: Generate a proof PF ={m̃=∑i∈I cimi, σ̃=∏i∈I σi^ci}
PF
ProofCheck: Verify the proof PF
Verification result
Payment
If yes, pay for the storage fee
Set t̃=t+T and update the voucher π
π
If no, claim for compensation

Fig. 3. Workflow of the remote data auditing protocol.

pp, this algorithm sets his/her private key $usk_{ID} = x$ and his/her public key $upk_{ID} = h^x$, where $x \in \mathbb{Z}_q$ is randomly selected in $\mathbb{Z}_q^*$. The user with an identity ID requests a certificate to the CSP by providing the public key upk_{ID}.

3) Certify: It takes the public parameters pp, system master private key msk and a user identity ID as input, the CSP generates a certificate $Cert_{ID} = H_1(ID, upk_{ID})^s$ for user with identity ID after checking the authenticity of the user identity. Then, the CSP sends a certificate $Cert_{ID}$ to the user.

If $e(Cert_{ID}, h) = e(H_1(ID, upk_{ID}), mpk)$ holds, the certificate is valid.

4) TagGen: Given an encrypted file $F \in \{0,1\}^*$ with the filename $fname$ and the timestamp t, the data owner first splits F into m blocks $\boldsymbol{m}_1, \dots, \boldsymbol{m}_m$ (each block includes n block sectors) and computes the file label $\tau = fname||t||m||sig_{usk_{ID}}(fname, t, m)$. The data owner then computes the tag of $\boldsymbol{m}_i, 1 \le i \le m$ with $\sigma_i = (\beta_i \cdot Cert_{ID}^{\sum_{j=1}^{n} \alpha_j \cdot m_{ij}})^{\frac{1}{usk_{ID}+\hat{u}}}$, where $\beta_i = H_3(ID, i, fname), \alpha_j = H_4(ID, j, upk_{ID})$, $\hat{u} = H_2(ID, fname, t, upk_{ID})$, and sends $(\tau, \boldsymbol{m}_1, \sigma_1, \dots, \boldsymbol{m}_m, \sigma_m)$ to the CSP.

5) Confirm: On receiving a file from the data owner, the CSP will perform verification on it. The CSP returns "failure" if the file is damaged. Otherwise, the CSP sets the latest audit time $\tilde{t} = t$, generates a vouch-

er $\pi = sig_{msk}(ID, fname, t, \hat{t}, T)$ and returns π to the data owner.

Note that, the auditing period T is chosen by the data owner according to the actual demand, and the latest audit time $\hat{t} = \hat{t} + T$ and the voucher π will be updated after receiving the storage fee from the data owner.

Upon receiving the voucher π from the CSP, the data owner stores the file label τ, the voucher π and the auditing period T in local. Meanwhile, the data owner provides (ID, upk_{ID}, τ, T) to TPA for audit periodically.

6) Challenge: The TPA runs this algorithm. For each period, it chooses a subset $I \subseteq [1, m]$ randomly and selects $c_i \in \mathbb{Z}_q{}^*$ for every $i \in I$. At the end of each auditing period T, the TPA launches a challenge $chal = \{ID, fname, t, (i, c_i) : i \in I\}$ to the CSP.

7) ProofGen: Upon receiving a challenge $chal$, the CSP computes and returns the proof $PF = \{\tilde{\boldsymbol{m}}, \tilde{\sigma}\}$ to the TPA, where $\tilde{\boldsymbol{m}} = \sum_{i \in I} c_i \boldsymbol{m}_i = (\tilde{m}_1, \dots, \tilde{m}_n), \tilde{\sigma} = \prod_{i \in I} \sigma_i{}^{c_i}$.

8) ProofCheck: On receiving a response PF, the TPA computes $\beta_i = H_3(ID, i, fname), \alpha_j = H_4(ID, j, upk_{ID}), \hat{u} = H_2(ID, fname, t, upk_{ID})$ and then checks the proof by the following equation

$$\begin{aligned} & e(\tilde{\sigma}, upk_{ID} \cdot h^{\hat{u}}) \\ & = e\left(\prod_{i \in I} \beta_i{}^{c_i}, h\right) \cdot e\left(H_1(ID, upk_{ID})^{\sum_{j=1}^{n} \alpha_j \cdot \tilde{m}_j}, mpk\right) \end{aligned} \tag{1}$$

If the above equation holds, the challenged file is intact. The data owner pays the storage fee to CSP. Otherwise, the data owner claims for compensation.

2. Correctness

Assume all the entities faithfully follows the protocol, we can check the correctness of the verification equation.

$$\begin{aligned} & e\left(\tilde{\sigma}, upk_{ID} \cdot h^{\hat{u}}\right) \\ & = e\left(\prod_{i \in I} \sigma_i{}^{c_i}, h^{x+\hat{u}}\right) \\ & = e\left(\left(\prod_{i \in I} \beta_i^{c_i} \cdot H_1(ID, upk_{ID})^{s \sum_{j=1}^{n} \alpha_j \cdot \tilde{m}_j}\right)^{\frac{1}{x+\hat{u}}}, h^{x+\hat{u}}\right) \\ & = e\left(\prod_{i \in I} \beta_i{}^{c_i} \cdot H_1(ID, upk_{ID})^{s \sum_{j=1}^{n} \alpha_j \cdot \tilde{m}_j}, h\right) \\ & = e\left(\prod_{i \in I} \beta_i{}^{c_i}, h\right) \cdot e\left(H_1(ID, upk_{ID})^{\sum_{j=1}^{n} \alpha_j \cdot \tilde{m}_j}, mpk\right) \end{aligned} \tag{2}$$

3. Security analysis

We prove the proposed protocol is secure under adaptive chosen identity attacks and adaptive chosen file attacks in random oracle model. The security proof is conducted as follows: 1) The single tag of the data block is unforgeable. 2) The proof PF is unforgeable.

Theorem 1 If the advantage of forging the single tag by $\mathcal{A}_I$ is at most ϵ with the running time t, the advantage of the challenger to solve the modified k-CAA problem is $\epsilon' \geq (1 - \frac{1}{q_u})^{q_r + q_e}(1 - \frac{1}{q_t + 1})^{q_t} \frac{1}{(q_t+1)q_u}\epsilon$, where q_u, q_r, q_e, q_t are the times of User-key-gen query, Corruption query, Certification query and TagGen query, respectively.

Proof Appendix A shows the detailed proof.

Theorem 2 If the advantage of a Type II adversary forges the single tag is at most ϵ with the running time t, the advantage of the challenger to solve the k-CAA problem is $\epsilon' \geq (1 - \frac{1}{q_u})^{q_r}(1 - \frac{1}{q_t+1})^{q_t} \frac{1}{(q_t+1)q_u}\epsilon$, and q_u, q_r, q_t are the times of User-key-gen query, Corruption query and TagGen query, respectively.

Proof Appendix B shows the detailed proof.

Theorem 3 If the challenged file is damaged or deleted, the CSP cannot forge the valid proof with a non-negligible probability.

Proof Appendix C shows the detailed proof.

4. Periodic auditing

If the data owner uploads all data blocks and file's related information successfully, the CSP will provide some audit periods (such as a week, a month, etc.) for data owner to choose. The data owner selects appropriate auditing period T according to actual needs.

The data owner with the identity ID must provide public key upk_{ID}, the file name $fname$ and the number of data blocks m to the TPA for public auditing. In our protocol, the data owner provides some additional information including the latest auditing time $\hat{t} = t$ and the auditing period T to TPA. The TPA can compute the latest audit time according to $\hat{t} = \hat{t} + T$, and then realize periodic auditing. The TPA can issue a valid challenge according to $fname$ and m at the latest audit time. In summary, the TPA can audit the file and timestamp stored in CSP periodically.

5. Accountability

The data owner and the CSP are the two entities that can be responsible for files. Once the file is detected to be damaged, the error may be happened during data upload phase or storage phase. To reduce the controversy, we add the confirm function in our protocol. The CSP will generate and return a voucher π to data owner if the file's related information $(fname, t, m)$ and all data blocks are verified to be correct. This voucher can be used to prove that the file has been uploaded successfully. After that, any data error being detected must be the storage error.

6. User friendly

In a remote data auditing system, the CSP stores all data blocks' tags to ensure data integrity. The data owner has to pay storage fee for storage volumes of tags. In our protocol, the size of each data block $\boldsymbol{m}$ is $(n \times q)$-bits (each sector is an element of $\mathbb{Z}_q$ and n is the number of data block's sector) and the size of its corresponding tag σ is about q-bits (i.e., an element of $\mathbb{G}_1$). The data owner only needs to pay the fixed storage fee. We can find that the bigger of the data block the lower of storage fee. However, n should be set a reasonable value since the CSP is required to return a proof $PF = \{\tilde{\boldsymbol{m}}, \tilde{\sigma}\}$ in auditing phase. If the data block's size is too large, it will bring high communication cost. Therefore, to reduce the storage cost, n should be as large as possible without obstructing the data integrity auditing.

V. Performance Evaluation

1. Theoretical analysis

We summarize the efficiency and functionality of our protocol in terms of computation cost, communication cost, storage cost and detection rate. Moreover, we also show a comparison among our protocol, Wang *et al.*'s protocol [7], and Wu *et al.*'s PDP protocol [14]. For simplicity, in this section we assume the data owner stores a file with m data blocks, and each block contains n-sectors. The TPA checks the integrity of file by challenging c different data blocks. Moreover, we denote the notations in Table 1.

Table 1. Notations

Notions	Descriptions
T_H	A map-to-point hash computation cost
T_P	A bilinear pairing computation cost
$T_{E_1}, T_{E_2}, T_{E_T}$	An exponential cost in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively
$T_{M_1}, T_{M_2}, T_{M_T}$	A multiplicative cost in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively
$\lvert\mathbb{G}_1\rvert, \lvert\mathbb{G}_2\rvert, \lvert\mathbb{G}_T\rvert$	The binary length of an element in $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$
$\lvert\mathbb{Z}_q\rvert$	The binary length of an element in $\mathbb{Z}_q$
$\lvert sig\rvert$	The binary length of the cited signature scheme

1) Computation cost: In Table 2, we list the computation cost of our protocol and the other two protocols. The comparison results show that the computation cost of our protocol is independent with the number of data block sectors n. Moreover, our protocol has lower the computation cost.

Table 2. Comparison of computation cost

Ref.	TagGen	ProofGen	ProofCheck
[7]	$mn(2T_{E_1} + T_H + T_{M_1})$	$(c-1)T_{M_1} + cT_{E_1} + nT_{E_T}$	$(n+c+1)T_{E_1} + cT_H$ $+ nT_{M_1} + nT_{M_T} + 2T_P$
[14]	$mn(2T_{E_1} + T_H + T_{M_1})$	$(c+2)T_{E_1} + (c-1)T_{M_1} + 2T_{M_2}$	$(c+n)T_{E_1} + (c+2)T_{M_1}$ $+ 2T_{E_2} + 2T_{M_2} + 5T_P$
Ours	$m(2T_{E_1} + T_H + T_{M_1})$	$cT_{E_1} + (c-1)T_{M_1}$	$(c+1)T_{E_1} + T_H + (c-1)T_{M_1}$ $+ T_{E_2} + T_{M_2} + 3T_P$

2) Communication cost: From the Table 3, we can see that the data owner uploads $mn|\mathbb{Z}_q| + m|\mathbb{G}_1| + |sig|$ bits to the CSP in our protocol, which is fewer $n|\mathbb{G}_1|$ bits than the comparison protocols. In the auditing phase, our protocol costs $(c+n)|\mathbb{Z}_q| + |\mathbb{G}_1|$ bits to transfer information, it is lower than Wang *et al.*'s protocol [7], and acceptable compared to Wu *et al.*'s PDP protocol [14].

Table 3. Comparison of communication cost

Ref.	Outsourcing storage	Auditing
[7]	$mn\lvert\mathbb{Z}_q\rvert + m\lvert\mathbb{G}_1\rvert + \lvert sig\rvert$	$(c+n)\lvert\mathbb{Z}_q\rvert + \lvert\mathbb{G}_1\rvert + n\lvert\mathbb{G}_T\rvert$
[14]	$mn\lvert\mathbb{Z}_q\rvert + (m+n)\lvert\mathbb{G}_1\rvert + \lvert sig\rvert$	$(c+4)\lvert\mathbb{Z}_q\rvert + 3\lvert\mathbb{G}_1\rvert + 2$
Ours	$mn\lvert\mathbb{Z}_q\rvert + m\lvert\mathbb{G}_1\rvert + \lvert sig\rvert$	$(c+n)\lvert\mathbb{Z}_q\rvert + \lvert\mathbb{G}_1\rvert$

3) Storage cost: We only consider the storage cost at CSP side. The CSP will store all verification information which includes all data blocks and data blocks' tags. The CSP costs $mn|\mathbb{Z}_q| + (m+n)|\mathbb{G}_1| + |sig|$ to store verification information in Wang *et al.*'s protocol [7]. While in [14], the CSP will cost about $(mn+1)|\mathbb{Z}_q| + (m+n)|\mathbb{G}_1| + |sig|$ to store verification information. The CSP only needs to cost $mn|\mathbb{Z}_q| + m|\mathbb{G}_1| + |sig|$ to store verification information in our construction. Obviously, the storage cost in our protocol is less than the other two protocols.

4) Detection rate analysis: Suppose m is the number of the total data blocks, and $x \le m$ is the number of the corrupted data blocks, then $k = x/m$ is the file corruption rate. If a file is corrupted, we use P_x to denote the probability of being successfully detected. Let c' denote the number of the corrupted data blocks that are chosen in challenge phase. We can see that $P_x = 1$ if $c > m - x$, and $P_x = P\{c' \ge 1\} = 1 - P\{c' < 1\}$ if $c \le m - x$.

Since $P\{c' < 1\} = \frac{\binom{m-x}{c}}{\binom{m}{c}} = \frac{(m-x)\ldots(m-x-(c-1))}{m\ldots(m-(c-1))}$, and $\frac{m-x-(c-1)}{m-(c-1)} \le \frac{m-x-i}{m-i} \le \frac{m-x}{m}$, P_x follows:

$$1-\left(1-\frac{x}{m}\right)^c \leq P_x \leq 1-\left(1-\frac{x}{m-c+1}\right)^c \qquad (3)$$

From (3), we have $1-P_x \leq (1-k)^c$. Therefore, to make $P_x \geq 0.99$, if the file corruption rate k is 3%, the TPA needs to choose 151 data blocks randomly; If $k=5\%$, the TPA needs to choose 90 data blocks randomly.

2. Experiment analysis

We evaluate experiments on a laptop (4GB RAM, Intel i5 3.2 GHz quad-core processor), and employ the standard paring *f.param* of JPBC [35] to run these protocols. In standard paring *f.param* (80-bit security level), the size of elments in $\mathbb{G}_1$ is 160 bits and in $\mathbb{G}_2$ is 320 bits. We choose a 1 MB (1048576 bytes) file to test the performance of our protocol, and the size of data sector is 160 bits. Therefore, m and n should satisfy $\frac{160(m-1)n}{8} \leq 1048576 \leq \frac{160mn}{8}$.

1) Tag generation cost: Let n vary between 1 and 100. We measure the computation cost of our protocol, Wang *et al.*'s protocol [7], and Wu *et al.*'s protocol [14] in the tag generation phase. We employ the BLS [19] to implement all the cited signature schemes in the three protocols.

As shown in Fig.4, the experimental results show that the tag generation cost is nearly constant for signing a file in [7] and [14], and the time cost will reduce as the growth of n in our protocol. It only requires 2.17 s to sign a 1 MB file with 525 data blocks and 100 sectors. Besides, the computation cost almost unchanged as the growth of n in the comparison protocol. Moreover, we compare the tag generation cost for different file sizes in Fig.5, and our protocol is efficient.

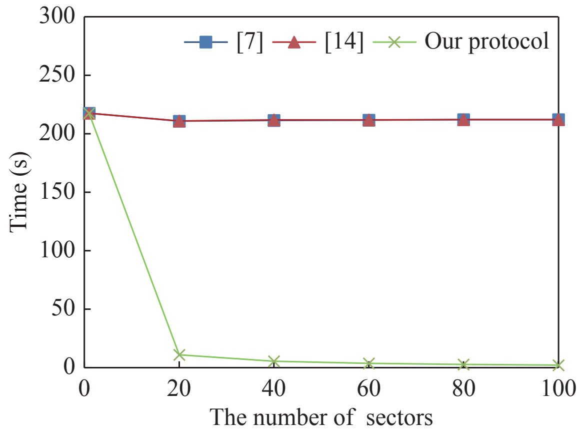

Fig. 4. Compuation cost for tag generation.

2) Proof generation cost: We test the proof generation cost by playing the role of the CSP. In this experiment, we split the file into 2622 data blocks (the sector size is fixed at 20), and choose c at the range of 20 to 160. In Fig.6, the proof generation cost is lower when $c=160$ than $c=20$. Furthermore, the time cost increases linearly in the above three protocols as c grows. The difference of the three protocols is tiny. The proof generation cost is about 0.3 s for 160 challenged data blocks in our protocol.

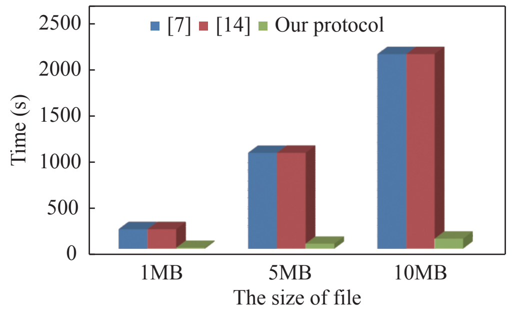

Fig. 5. Tag generation cost for different files.

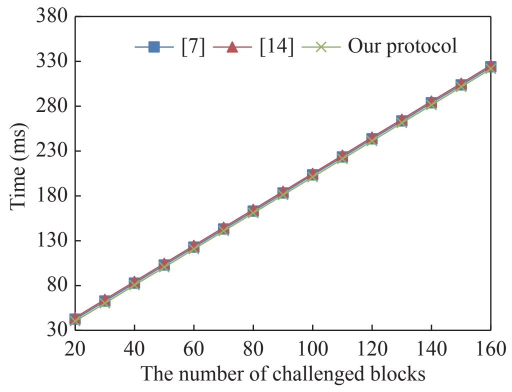

Fig. 6. Proof generation cost.

3) ProofCheck cost: The TPA runs the algorithm ProofCheck to verify the validity of the proof. Let c vary between 20 and 160. Fig.7 illustrates that our protocol takes less cost to detect proofs than the other two protocols. Besides, the proof verification cost increases linearly with c in all three protocols. The proof verification cost is about 0.55 s for 160 challenged data blocks in our protocol.

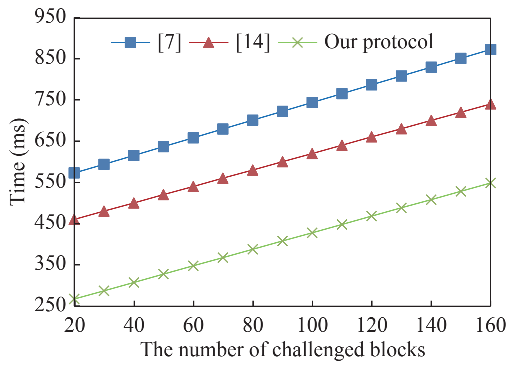

Fig. 7. Proof verification cost.

4) Detection rate: Considering the number of the challenged data block from 20 to 160, we give the probability of successfully detecting whether the file is con-

taminated. We test the file with 3%, 5%, 10%, and 15% corruption rates, respectively. From the Fig.8 , we can observe that the more data blocks are selected, the higher probability of detecting the damaged files. If the file corruption rate is 3%, Fig.8 shows 140 data blocks are required to get a detection rate higher than 0.99, while 100 data blocks are required if the detection rate is not less than 0.97. If the file corruption rate is 15% (i.e., 67 data blocks are corrupted), Fig.8 shows 40 data blocks are required to make the detection rate close to 1, while 20 data blocks if the detection rate is not less than 0.96. The results show no difference from the theoretical analysis.

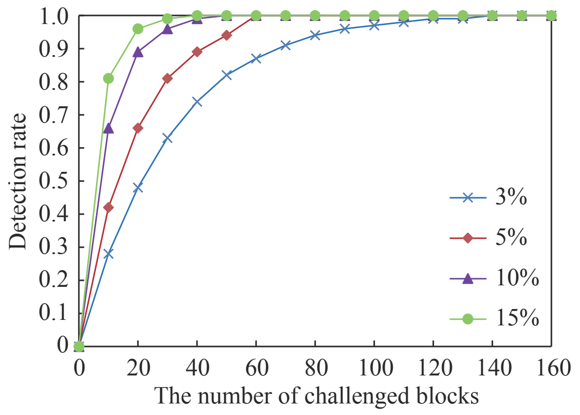

Fig. 8. The probability of detecting successfully.

VI. Conclusions

This paper presents a protocol for auditing data remotely, which is suitable for pay-as-you-go business model. In our construction, the TPA can check whether the data is stored correctly according to the auditing period. The data owner pays storage fee according to the pay-as-you-go business model if the file keeps intact. Once a file is detected error or lost, the data owner will stop to pay storage fee and require the CSP to compensate for the damaged file. We prove the correctness and security of our protocol in random oracle. We then test the computational cost from theoretical and experimental aspects, respectively. The experiment results illustrate that the proposed protocol in this paper is more practical than other two protocols.

Appendix A. Proof of Theorem 1

Theorem 1 proves that our protocol is secure against the Type I adversary which can change the public key and keep the target user's certificate secret.

Proof Suppose $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e)$ are public parameters, and there exists an algorithm $\mathcal{B}$ to settle the modified k-CAA problem in polynomial-time.

For $x, a, b \in \mathbb{Z}_q^*$, given a modified k-CAA problem instance $\{h_1, \dots, h_k \in \mathbb{Z}_q^*,\ g, g^b \in \mathbb{G}_1,\ h^x, h^a \in \mathbb{G}_2, g^{\frac{ab}{x+h_1}}, \dots, g^{\frac{ab}{x+h_k}}\}$, $\mathcal{B}$ can compute $g^{\frac{ab}{x+h^*}}$ when $h^* \notin \{h_1, \dots, h_k\}$ or g^{ab}.

Initialization (Phase 1): $\mathcal{B}$ lets $pk = h^a$, chooses a hash function $H_4 : \{0,1\}^* \times \mathbb{Z}_q \to \mathbb{Z}_q$, and sets three hash functions H_1, H_2, H_3 as random oracle. Let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ denote one-way mapping and sig is a secure signature scheme. $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e)$ are known to both $\mathcal{B}$ and $\mathcal{A}_{\rm I}$.

Oracle simulation (Phase 2): $\mathcal{A}_{\rm I}$ issues oracles adaptively.

- User-key-gen query. Suppose the i-th identity is marked as ID_i, and $\mathcal{B}$ chooses ID_I from $\{ID_1, \dots, ID_{q_u}\}$ as the challenge identity. $\mathcal{B}$ stores (ID_i, usk_i, upk_i) list that is called L_u and initially empty. $\mathcal{B}$ will check the list L_u when receiving a query from $\mathcal{A}_{\rm I}$. If $\mathcal{B}$ finds ID_i in the L_u list, then sends (usk_i, upk_i) to $\mathcal{A}_{\rm I}$. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ chooses $x_i \in \mathbb{Z}_q{}^*$ at random, sets $upk_i = h^{x_i}$ and $usk_i = x_i$. In this case, $\mathcal{B}$ returns (usk_i, upk_i) to $\mathcal{A}_{\rm I}$ and inserts (ID_i, usk_i, upk_i) into L_u.

2) If $i = I$, $\mathcal{B}$ arranges $upk_i = h^x$, where the user private key is unknown to $\mathcal{B}$. In this case, $\mathcal{B}$ adds $(ID_i, \triangle, upk_i)$ into the list L_u and returns upk_i to adversary $\mathcal{A}_{\rm I}$.

- H_1 query. $\mathcal{B}$ stores (ID_i, H_{1i}, d_i) list that is called L_{H1} and initially empty. If $\mathcal{B}$ finds ID_i in the L_{H1} list, then sends H_{1i} to $\mathcal{A}_{\rm I}$. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ chooses $d_i \in \mathbb{Z}_q{}^*$ at random, computes $H_1(ID_i, upk_i) = g^{d_i}$, sends H_{1i} to $\mathcal{A}_{\rm I}$ and inserts (ID_i, H_{1i}, d_i) into L_{H1}.

2) If $i = I$, $\mathcal{B}$ arranges $H_1(ID_i, upk_i) = g^b$, returns H_{1i} to adversary $\mathcal{A}_{\rm I}$ and inserts $(ID_i, H_{1i}, \perp)$ into L_{H1}.

- Corruption query. When receiving a corruption query from $\mathcal{A}_{\rm I}$, $\mathcal{B}$ terminates the simulation and outputs $\perp$ if $ID_i = ID_{\rm I}$ or ID_i is not in L_u, Otherwise, $\mathcal{B}$ finds x_i from L_u and returns it to $\mathcal{A}_I$.

- Certification query. $\mathcal{B}$ holds $(ID_i, Cert_{ID_i})$ list that called L_c and initially empty. If $\mathcal{B}$ finds ID_i in L_c, then sends $Cert_{ID_i}$ to $\mathcal{A}_{\rm I}$. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ calculates $Cert_{ID_i} = H_1(ID_i, upk_i)^a = (\psi(h)^{d_i})^a = \psi(h^a)^{d_i}$, where d_i is extracted from L_{H1} list. The $\mathcal{B}$ returns $Cert_{ID_i}$ to $\mathcal{A}_{\rm I}$ and adds $(ID_i, Cert_{ID_i})$ into L_c list.

2) If $i = I$, $\mathcal{B}$ outputs $\perp$.

- Key-replace Query. $\mathcal{B}$ checks if $g^{usk'} = upk'$ hold when receiving a novel (usk', upk') on ID_i. If the equation holds, then $\mathcal{B}$ inserts $\left(ID_i, usk', upk_i'\right)$ into L_u.

- H_2 query. $\mathcal{B}$ stores $(ID_i, fname, t, upk_i, H_{2i}, c)$ list that is called L_{H2} and initially empty. If $\mathcal{B}$ finds $(ID_i, fname, t)$ in L_{H2}, then returns H_{2i} to the adversary. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ picks $H_{2i} \in \mathbb{Z}_q^*$ randomly, and returns H_{2i}. Also, $\mathcal{B}$ inserts $(ID_i, fname, t, upk_i, H_{2i}, \perp)$ into L_{H2}.

2) If $i = I$, $\mathcal{B}$ flips up a coin. $\mathcal{B}$ uses $c = 1$ to represent heads up with the probability ζ, and uses $c = 0$ to represent tails up with the probability $1 - \zeta$.

a) If $c = 1$, $\mathcal{B}$ selects $H_{2i} \in \mathbb{Z}_q^*$ at random, and $H_{2i} \notin \{h_1, ..., h_k\}$. $\mathcal{B}$ then sends $H_2(ID_i, fname, t, upk_i) = H_{2i}$ to $\mathcal{A}_{\rm I}$, and adds $(ID_i, fname, t, upk_i, H_{2i}, c)$ into L_{H2} list.

b) If $c = 0$, $\mathcal{B}$ chooses $\hat{h} \in \{h_1, \dots, h_k\}$ that has never been selected, and sets $H_2(ID_i, fname, t, upk_i) = \hat{h}$. Finally, $\mathcal{B}$ sends $\hat{h}$ to the adversary, and inserts $\left(ID_i, fname, t, upk_i, \hat{h}, c\right)$ into L_{H2}.

- H_3 query. $\mathcal{B}$ stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ that is called L_{H3} and initially empty. If $\mathcal{B}$ finds $(ID_i, fname)$ in L_{H3}, then sends $\beta_1, \dots, \beta_m$ to $\mathcal{A}_{\mathrm{I}}$. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ randomly chooses $\beta_1, \dots, \beta_m \in \mathbb{G}_1$ and sets $H_{3j}(ID_i, j, fname) = \beta_j, 1 \le j \le m$. $\mathcal{B}$ returns $\beta_1, \dots, \beta_m$ to $\mathcal{A}_{\mathrm{I}}$ and stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ to the list L_{H3}.

2) If $i = I$, $\mathcal{B}$ first takes out H_{2i} from L_{H_2} list and randomly selects $r_j \in \mathbb{Z}_q^*, 1 \le j \le m$. $\mathcal{B}$ then computes $\beta_j = (\psi(h^x) \cdot \psi(h)^{\hat{h}})^{r_j}$ and sets $H_3(ID_i, j, fname) = \beta_j$. $\mathcal{B}$ returns $\beta_1, \dots, \beta_m$ to $\mathcal{A}_{\mathrm{I}}$ and stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ to the list L_{H3}.

- TagGen query. $\mathcal{A}_{\mathrm{I}}$ selects ID_i, the filename $fname$, the file's timestamp t and a data block $\boldsymbol{m}_j \in F, j \in [1, m]$, user's current private key usk_i (If the upk_i has not been replaced, then $usk_i = \perp$). $\mathcal{B}$ outputs $\perp$ if ID_i has not been created.

1) If $i \neq I$, $\mathcal{B}$ first seeks usk_i from L_u, $Cert_{ID_i}$ from L_c and H_{2i} from L_{H2}.

a) $\mathcal{B}$ computes $sig_{usk_i}(fname, t, m)$ and sets the file label $\tau = fname||t||m||sig_{usk_i}(fname, t, m)$.

b) $\mathcal{B}$ computes the tag of the data block $\boldsymbol{m}_j$ by the following equation

$$\sigma_j = \left(\beta_j \cdot Cert_{ID_i}{}^{\sum_{k=1}^{n} H_4(ID,k) \cdot m_{jk}}\right)^{\frac{1}{x_i + H_{2i}}}$$

c) $\mathcal{B}$ responds label τ and data block's tag σ_j to $\mathcal{A}_{\mathrm{I}}$.

2) If $i = I$, the user's certificate $Cert_{IDt}$ of ID_t is unknown to $\mathcal{B}$. $\mathcal{B}$ iterates over L_{H2}.

a) $\mathcal{B}$ aborts the simulation if $c = 1$.

b) *$\mathcal{B}$ takes out $H_{2t} = \hat{h} \in \{h_1, \dots, h_k\}$ from L_{H2} if $c = 0$ and upk_i is original.*

c) $\mathcal{B}$ calculates $sig_{usk_i}(fname, t, m)$ and sets the file label $\tau = fname||t||m||sig_{usk_i}(fname, t, m)$.

d) $\mathcal{B}$ computes the tag of the data block $\boldsymbol{m}_j$ by the following equation

$$\sigma_j = \psi(h)^{r_j} \cdot ((g^{ab})^{\frac{1}{x+\hat{h}}})^{\sum_{k=1}^{n} H_4(ID,j) \cdot m_{jk}}$$

e) $\mathcal{B}$ responds label τ and data block's tag σ_j to $\mathcal{A}_{\mathrm{I}}$.

Output (Phase 3): $\mathcal{A}_{\mathrm{I}}$ outputs $(ID^*, upk^*, \tau^*, \boldsymbol{m}^*, \sigma^*)$. $\mathcal{A}_{\mathrm{I}}$ wins the game if the following conditions are held:

a) A corruption query on ID^* has never been launched.

b) $\mathcal{A}_I$ has never queried the certificate of ID^*.

c) $(ID^*, fname^*, \tau^*, \boldsymbol{m}^*)$ has never been asked the tag.

d) Proof check $\left(pp, ID^*, upk^*_{ID}, \tau^*, \perp, \{\boldsymbol{m}^*, \sigma^*\}\right)$=1. $\mathcal{B}$ knows usk^* even if $\mathcal{A}_{\mathrm{I}}$ has replaced public key. We have:

$$\begin{aligned}
\sigma^* &= (H_3(ID^*, upk^*) \cdot Cert_{ID^*}^{\sum_{k=1}^{n} H_4(ID^*,k) \cdot m_k})^{\frac{1}{usk^*+h^*}} \\
&= (\psi(h)^{r^*(usk^*+h*)} \cdot g^{ab\sum_{k=1}^{n} H_4(ID^*,k) \cdot m_k})^{\frac{1}{usk^*+h^*}} \\
&= \psi(h)^{r^*} \cdot (g^{ab\sum_{k=1}^{n} H_4(ID^*,k) \cdot m_k})^{\frac{1}{usk^*+h^*}} \\
&\Rightarrow \left(\frac{\sigma^*}{\psi(h)^{r^*}}\right)^{(\sum_{k=1}^{n} H_4(ID^*,k) \cdot m_k)^{-1}} \\
&= (g^{ab})^{(x+h^*)^{-1}} \qquad (ID^* = ID_I)
\end{aligned}$$

where $h^* = H_2{}^*(ID^*, fname^*, t, upk^*) = \hat{h}$, $usk^* = x$, $msk = a$ and $H_3(ID^*, upk^*) = \psi(h)^{r^*(x+h^*)}$.

If $ID^* \neq ID_I$ or $ID^* = ID_I$ but $c^* = 0$, $\mathcal{B}$ aborts the game. Otherwise, if $c^* = 1$ and $h^* \notin \{h_1, \dots, h_k\}$, $\mathcal{B}$ can compute the modified k-CAA problem.

1) If $\mathcal{A}_{\mathrm{I}}$ has never replaced the public key upk^*, $\mathcal{B}$ computes $g^{\frac{ab}{x+h^*}}$ as the solution of the modified k-CAA problem.

2) If $\mathcal{A}_{\mathrm{I}}$ has replaced the public key upk^*, $\mathcal{B}$ computes $g^{ab} = \left(g^{\frac{ab}{usk^*+h^*}}\right)^{(usk^*+h^*)}$ as the solution of the modified k-CAA problem.

Probability analysis: Suppose $\mathcal{B}$ can get the solution of the modified k-CAA problem, the following conditions must be held:

a) E_1: The simulation has never been aborted in the Corruption query phase, Certification query phase, and TagGen query phase. And the probability is $\Pr[E_1] = (1 - \frac{1}{q_u})^{q_r+q_e}(1 - \frac{1}{q_u}\zeta)^{q_t} \ge (1 - \frac{1}{q_u})^{q_r+q_e}(1 - \zeta)^{q_t}$.

b) E_2: $\mathcal{A}_{\mathrm{I}}$ forges a valid signature. The probability is $\Pr[E_2|E_1] = \epsilon$.

c) E_3: $ID^* = ID_{\mathrm{I}}$ and $c^* = 1$. The probability is $\Pr[E_3|E_1 \wedge E_2] = \frac{\zeta}{q_u}$.

The probability that $\mathcal{B}$ succeeds is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \wedge E_2] \ge (1 - \frac{1}{q_u})^{q_r+q_e}(1 - \zeta)^{q_t}\frac{\zeta}{q_u}\epsilon$. Since $\zeta = \frac{1}{q_t+1}$ $(1 - \zeta)^{q_t}\zeta$ can take the maximum value, we have

$$\epsilon' \ge \left(1 - \frac{1}{q_u}\right)^{q_r+q_e}\left(1 - \frac{1}{q_t+1}\right)^{q_t}\frac{1}{(q_t+1)q_u}\epsilon$$

Appendix B. Proof of Theorem 2

Theorem 2 proves that our protocol is secure against the Type II adversary which has the master key but cannot change the public key.

Proof Suppose $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e)$ are public parameters, and there exists an algorithm $\mathcal{B}$ to settle the k-CAA problem in polynomial-time.

For $x \in \mathbb{Z}_q^*$, given a k-CAA problem instance $\{h_1, \dots, h_k \in \mathbb{Z}_q^*, g \in \mathbb{G}_1, h^x \in \mathbb{G}_2, g^{\frac{1}{x+h_1}}, \dots, g^{\frac{1}{x+h_k}}\}$, $\mathcal{B}$ can compute $g^{\frac{1}{x+h^*}}$ when $h^* \notin \{h_1, \dots, h_k\}$.

Initialization (Phase 1): $\mathcal{B}$ selects $s \in \mathbb{Z}_q{}^*$ at random and computes $pk = h^s$, chooses a hash function $H_4 : \{0,1\}^* \times \mathbb{Z}_q \to \mathbb{Z}_q$, and sets three hash functions H_1, H_2, H_3 as random oracle. Let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ denote one-way mapping and sig is a secure and efficient signature scheme. Both $\mathcal{B}$ and $\mathcal{A}_{\mathrm{II}}$ hold $pp = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, h, e, pk, H_4)$.

Oracle simulation (Phase 2): $\mathcal{A}_{\mathrm{II}}$ issues oracles adaptively.

- User-key-gen query. Suppose the i-th identity is marked as ID_i, $\mathcal{B}$ chooses ID_I from $\{ID_1, \dots, ID_{q_u}\}$ as the challenge identity. $\mathcal{B}$ stores (ID_i, usk_i, upk_i) list that is called L_u and initially empty. $\mathcal{B}$ will check the list L_u when receiving a query from $\mathcal{A}_{\mathrm{II}}$. If $\mathcal{B}$ finds ID_i in L_u, then responses (usk_i, upk_i). Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ selects $x_i \in \mathbb{Z}_q{}^*$ at random, sets $upk_i = h^{x_i}$ and $usk_i = x_i$. In this case, $\mathcal{B}$ returns (usk_i, upk_i) to $\mathcal{A}_{\mathrm{II}}$ and inserts

(ID_i, usk_i, upk_i) into L_u.

2) If $i = I$, $\mathcal{B}$ arranges $upk_i = h^x$, and $\mathcal{B}$ doesn't know the user's private key. In this case, $\mathcal{B}$ adds $(ID_i, \triangle, upk_i)$ into L_u and returns upk_i to $\mathcal{A}_{\text{II}}$.

- H_1 query. $\mathcal{B}$ stores (ID_i, H_{1i}, d_i) list that is called L_{H1} and initially empty. If $\mathcal{B}$ finds ID_i in L_{H1}, then sends H_{1i} to $\mathcal{A}_{\text{II}}$. Otherwise, $\mathcal{B}$ selects $d_i \in \mathbb{Z}_q^*$ randomly and does as follows:

1) If $i \neq I$, $\mathcal{B}$ computes $H_1(ID_i, upk_i) = g^{d_i}$, sends H_{1i} to the adversary and inserts (ID_i, H_{1i}, d_i) into L_{H1}.

2) If $i = I$, $\mathcal{B}$ computes $H_1(ID_i, upk_i) = g^{d_i/s}$, sends H_{1i} to the adversary and inserts (ID_i, H_{1i}, d_i) into L_{H1}.

- Corruption query. When receiving a corruption query from $\mathcal{A}_{\text{II}}$, $\mathcal{B}$ terminates the simulation and outputs $\perp$ if $ID_i = ID_I$ or ID_i is not in L_u, Otherwise, $\mathcal{B}$ finds x_i from L_u and returns it to $\mathcal{A}_{\text{II}}$.

- H_2 query. $\mathcal{B}$ stores $(ID_i, fname, t, upk_i, H_{2i}, c)$ list that is called L_{H2} and initially empty. If $\mathcal{B}$ finds $(ID_i, fname, t)$ in L_{H2}, then returns H_{2i} to the adversary $\mathcal{A}_{\text{II}}$. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ selects $H_{2i} \in \mathbb{Z}_q^*$ randomly, and returns H_{2i} to the adversary. Also, $\mathcal{B}$ inserts $(ID_i, fname, t, upk_i, H_{2i}, \perp)$ into L_{H2}.

2) If $i = I$, $\mathcal{B}$ flips up a coin. $\mathcal{B}$ uses $c = 1$ to represent heads up with the probability ζ, and uses $c = 0$ to represent tails up with the probability $1 - \zeta$.

a) If $c = 1$, $\mathcal{B}$ selects $H_{2i} \in \mathbb{Z}_q^*$ at random, and $H_{2i} \notin \{h_1, \dots, h_k\}$. $\mathcal{B}$ then sends $H_2(ID_i, fname, t, upk_i) = H_{2i}$ to $\mathcal{A}_{\text{II}}$, and adds $(ID_i, fname, t, upk_i, H_{2i}, c)$ into L_{H2} list.

b) If $c = 0$, $\mathcal{B}$ chooses $\hat{h} \in \{h_1, \dots, h_k\}$ that has never been selected, and sets $H_2(ID_i, fname, t, upk_i) = \hat{h}$. $\mathcal{B}$ then returns $\hat{h}$ to $\mathcal{A}_{\text{II}}$, and inserts $(ID_i, fname, t, upk_i, \hat{h}, c)$ into L_{H2}.

- H_3 query. $\mathcal{B}$ stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ list that is called L_{H3} and initially empty. If $\mathcal{B}$ finds $(ID_i, fname)$ in L_{H3}, then sends $\beta_1, \dots, \beta_m$ to the adversary. Otherwise, $\mathcal{B}$ does as follows:

1) If $i \neq I$, $\mathcal{B}$ selects $\beta_1, \dots, \beta_m \in \mathbb{G}_1$ randomly and sets $H_{3j}(ID_i, j, fname) = \beta_j, 1 \leq j \leq m$. $\mathcal{B}$ returns $\beta_1, \dots, \beta_m$ to $\mathcal{A}_{\text{II}}$ and stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ to the list L_{H3}.

2) If $i = I$, $\mathcal{B}$ first takes out H_{2i} from L_{H_2} list and randomly selects $r_j \in \mathbb{Z}_q^*, 1 \leq j \leq m$. $\mathcal{B}$ then computes $\beta_j = \psi(h^x)^{r_j} \cdot (\psi(h)^{H_{2i}})^{r_j}$ and sets $H_3(ID_i, j, fname) = \beta_j$. $\mathcal{B}$ returns $\beta_1, \dots, \beta_m$ to $\mathcal{A}_{\text{II}}$ and stores $(ID_i, fname, \beta_1, \dots, \beta_m)$ to the list L_{H3}.

- TagGen query. $\mathcal{A}_{\text{II}}$ chooses a user identity ID_i, the filename $fname$, the file's timestamp t and a data block $\boldsymbol{m}_j \in F, j \in [1, m]$, user's private key usk_i.

1) If $i \neq I$, $\mathcal{B}$ first seeks private key usk_i from L_u and the H_{2i} from L_{H2}.

a) $\mathcal{B}$ computes $sig_{usk_i}(fname, t, m)$ and sets the file label $\tau = fname||t||m||sig_{usk_i}(fname, t, m)$.

b) $\mathcal{B}$ computes the tag of the data block $\boldsymbol{m}_j, 1 \leq j \leq m$ by the following equation

$$\sigma_j = \left(\beta_j \cdot \psi(h^s)^{d_i \sum_{k=1}^{n} H_4(ID,k) \cdot m_k}\right)^{\frac{1}{x_i + H_{2i}}}$$

c) $\mathcal{B}$ responds label τ and data block's tag σ_j to $\mathcal{A}_{\text{II}}$.

2) If $i = I$, $\mathcal{B}$ fist looks up L_{H2}.

a) If $c = 1$, $\mathcal{B}$ aborts the game.

b) If $c = 0$, $\mathcal{B}$ computes $sig_{usk_i}(fname, t, m)$ and sets the file label $\tau = fname||t||m||sig_{usk_i}(fname, t, m)$.

c) $\mathcal{B}$ computes the tag of the data block $\boldsymbol{m}_j, 1 \leq j \leq m$ by the following equation

$$\sigma_j = \psi(h)^{r_j} \cdot (g^{\frac{1}{x + H_{2i}}})^{d_i \sum_{k=1}^{n} H_4(ID,k) \cdot m_k}$$

d) $\mathcal{B}$ responds label τ and data block's tag σ_j to $\mathcal{A}_{\text{II}}$.

Output(Phase 3): $\mathcal{A}_{\text{II}}$ outputs $(ID^*, upk^*, \tau^*, \boldsymbol{m}^*, \sigma^*)$. $\mathcal{A}_{\text{II}}$ wins the game if the following conditions are held:

a) A corruption query on ID^* has never been launched.

b) $(ID^*, fname^*, \tau^*, \boldsymbol{m}^*)$ has never been asked the tag.

c) ProofCheck$(ID^*, upk^*, \tau^*, \perp, \{\boldsymbol{m}^*, \sigma^*\})$=1. That is,

$$\begin{aligned}
\sigma^* &= (H_3(ID^*, upk^*) \cdot Cert_{ID^*}^{\sum_{k=1}^{n} H_4(ID,k) \cdot m_j})^{\frac{1}{usk^* + h^*}} \\
&= (\psi(h)^{r^*(usk^* + h^*)} \cdot g^{d \sum_{k=1}^{n} H_4(ID,k) \cdot m_j})^{\frac{1}{usk^* + h^*}} \\
&= \psi(h)^{r^*} \cdot (g^{\frac{1}{usk^* + h^*}})^{d \sum_{j=1}^{n} H_4(ID,k) \cdot m_k} \\
&\Rightarrow \left(\frac{\sigma^*}{\psi(h)^{r^*}}\right)^{(d \sum_{k=1}^{n} H_4(ID,k) \cdot m_k)^{-1}} \\
&= g^{(x + h^*)^{-1}} \qquad (ID^* = ID_I)
\end{aligned}$$

where $h^* = H_2{}^*(ID^*, fname^*, t^*, upk^*) = \hat{h}$ and $H_3(ID^*, upk^*) = \psi(h)^{r^*(x + h^*)}$.

If $ID^* \neq ID_I$ or $ID^* = ID_I$ but $c^* = 0$, $\mathcal{B}$ aborts the game. Otherwise, if $c^* = 1$ and $h^* \notin \{h_1, \dots, h_k\}$, $\mathcal{B}$ looks up L_{H2}, and compute $g^{\frac{1}{x + h^*}} = (\frac{\sigma^*}{\psi(h)^{r^*}})^{(\sum_{k=1}^{n} H_4(ID,k) \cdot m_j)^{-1}}$ as the solution of the k-CAA problem.

Probability analysis: Suppose $\mathcal{B}$ can get the solution of the k-CAA problem, the following conditions must be held:

a) E_1: The simulation has never been aborted in the Corruption query phase and TagGen query phase. The probability is $\Pr[E_1] = (1 - \frac{1}{q_u})^{q_r}(1 - \frac{1}{q_u}\zeta)^{q_t} \geq (1 - \frac{1}{q_u})^{q_r + q_e}(1 - \zeta)^{q_t}$.

b) E_2: $\mathcal{A}_{\text{II}}$ forges a valid signature. The probability is $\Pr[E_2|E_1] = \epsilon$.

c) E_3: $ID^* = ID_I$ and $c^* = 1$. The probability is $\Pr[E_3|E_1 \wedge E_2] = \frac{\zeta}{q_u}$.

In summary, the probability that $\mathcal{B}$ succeeds is $\epsilon' = \Pr[E_1 \wedge E_2 \wedge E_3] = \Pr[E_1] \cdot \Pr[E_2|E_1] \cdot \Pr[E_3|E_1 \wedge E_2] \geq (1 - \frac{1}{q_u})^{q_r}(1 - \zeta)^{q_t}\frac{\zeta}{q_u}\epsilon$. Since $\zeta = \frac{1}{q_t + 1}, (1 - \zeta)^{q_t}\zeta$ can take the maximum value, we have

$$\epsilon' \geq \left(1 - \frac{1}{q_u}\right)^{q_r}\left(1 - \frac{1}{q_t + 1}\right)^{q_t}\frac{1}{(q_t + 1)q_u}\epsilon$$

Appendix C. Proof of Theorem 3

Proof We suppose the adversary $\mathcal{A}_{\text{III}}$ can forge a valid proof successfully.

Simulation: The system initialization and the oracle simulation are the same as in Game 1 or Game 2.

- ProofCheck. $\mathcal{A}_{\text{III}}$ generates the proof PF using some data blocks's tags, and sends PF and challenge to $\mathcal{B}$. $\mathcal{B}$ checks PF and sends the result to $\mathcal{A}_{\text{III}}$.

- Challenge. $chal = \{(i, c_i) : i \in I, I \subseteq [1, m], c_i \in \mathbb{Z}_q\}$ are chosen by $\mathcal{B}$ as a challenge. There is at least a challenged data block that never been queried tag. $\mathcal{B}$ then sends the challenge to $\mathcal{A}$.

Forge: The adversary $\mathcal{A}$ outs a valid proof $\overline{PF}=\left\{\boldsymbol{m}=\sum_{i\in I}c_i\boldsymbol{m}_i,\ \bar{\sigma}=\prod_{i\in I}\bar{\sigma}_i{}^{c_i}\right\}$ and sends it to $\mathcal{B}$.

Probability analysis: Since the forged proof is valid, it satisfies

$$e(\bar{\sigma}, upk^*\cdot h^{h^*})/e\left(\prod_{i\in I}\beta_i{}^{c_i}, h\right)$$
$$= e(H_1(ID^*, upk^*)^{\sum_{j=1}^{n}H_4(ID^*,j)\cdot m_j}, pk)$$

Assume that the real proof for the challenge $chal$ is $PF=\{\mathbf{m},\sigma\}$, it can also make the equation holds.

$$e(\sigma, upk^*\cdot h^{h^*})/e\left(\prod_{i\in I}\beta_i{}^{c_i}, h\right)$$
$$= e(H_1(ID^*, upk^*)^{\sum_{j=1}^{n}H_4(ID^*,j)\cdot m_j}, pk)$$

Since the hash function is collision resistance, the adversary $\mathcal{A}$ can get the only response when it issue a H_1 query on the same input. Similarly, H_2 query and H_3 query. Obviously, the above two equations are equal, i.e., $\sigma=\bar{\sigma}$, that is, $\prod_{i\in I}\sigma_i^{c_i}=\prod_{i\in I}\bar{\sigma}_i{}^{c_i}$. Because $\sigma_i,\bar{\sigma}_i\in\mathbb{G}_1$, there exists $x_i,y_i\in\mathbb{Z}_q^*$ satisfying $\sigma_i=g^{x_i}$ and $\bar{\sigma}_i=g^{y_i}$. We get $g^{\sum_{i\in I}c_ix_i}=g^{\sum_{i\in I}c_iy_i}$, i.e., $\sum_{i\in I}c_ix_i=\sum_{i\in I}c_iy_i$, which means $\sum_{i\in I}c_i(x_i-y_i)=0$. Since $c_i\in\mathbb{Z}_q^*$, we get $x_i=y_i \bmod q$. This is contrary to the previous results. According to the Theorem 1 and Theorem 2, the probability of forging a single tag is negligible. Thus, the probability of forging a proof successfully is negligible if a file has been deleted or damaged.

Theorem 3 is proved.

References

[1] B. Latré, B. Braem, I. Moerman, *et al.*, "A survey on wireless body area networks," *Wireless Networks*, vol.17, no.1, pp.1–18, 2011.

[2] J. F. Wan, C. F. Zou, S. Ullah, *et al.*, "Cloud-enabled wireless body area networks for pervasive healthcare," *IEEE Network*, vol.27, no.5, pp.56–61, 2013.

[3] S. Ullah, A. V. Vasilakos, H. Chao, *et al.*, "Cloud-assisted wireless body area networks," *Information Sciences,*, vol.284, pp.81–83, 2014.

[4] Y. Deswarte, J. J. Quisquater, and A. Saïdane, "Remote integrity checking," in *Proceedings of the Sixth Working Conference on Integrity and Internal Control in Information Systems*, Lausanne, Switzerland, pp.1–11, 2004.

[5] G. Ateniese, R. Burns, R. Curtmola, *et al.*, "Provable data possession at untrusted stores," in *Proceedings of ACM Conference on Computer and Communications Security*, Alexandria, Virginia, USA, pp.598–09, 2007.

[6] Y. M. Li and F. T. Zhang, " An efficient certificate-based data integrity auditing protocol for cloud-assisted WBANs," *IEEE Internet of Things Journal*, vol.9, no.13, pp.11513–11523, 2022.

[7] C. Wang, S. S. M. Chow, Q. Wang, *et al.*, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol.62, no.2, pp.362–375, 2013.

[8] B. Wang, B. Li, H. Li, *et al.*, " Certificateless public auditing for data integrity in the cloud," in *Proceedings of IEEE Conference on Communications and Network Security*, National Harbor, MD, USA, pp.136–144, 2013.

[9] F. Armknecht, J. M. Bohli, G. O. Karame, *et al.*, "Outsourced proofs of retrievability," in *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, Arizona, USA, pp.831–843, 2014.

[10] S. K. Nayak and S. Tripathy, " SEPDP: Secure and efficient privacy preserving provable data possession in cloud storage," *IEEE Transactions on Services Computing*, vol.14, no.3, pp.876–888, 2021.

[11] Y. N. Li, Y. Yu, G. Min, *et al.*, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol.16, no.1, pp.72–83, 2019.

[12] Z. Yang, W. Y. Wang, Y. Huang, *et al.*, "Privacy-preserving public auditing scheme for data confidentiality and accountability in cloud storage," *Chinese Journal of Electronics*, vol.28, no.1, pp.179–187, 2019.

[13] M. Armbrust, A. Fox, R. Griffith, *et al.*, " A view of cloud computing," *Communications of the ACM* , vol.53, no.4, pp.50–58, 2010.

[14] T. Wu, G. M. Yang, Y. Mu, *et al.*, "Privacy-preserving proof of storage for the pay-as-you-go business model," *IEEE Transactions on Dependable and Secure Computing*, vol.18, no.2, pp.563–575, 2021.

[15] T. G. Zimmerman, " Personal area networks: near-field intrabody communication," *IBM Systems Journal* , vol.35, no.3.4, pp.609–617, 1996.

[16] G. Ateniese, S. Kamara, and J. Katz, " Proofs of storage from homomorphic identification protocols," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Tokyo, Japan, pp.319–333, 2009.

[17] A. Juels and B. S. Kaliski, " PORs: Proofs of retrievability for large files," in *Proceedings of ACM Conference on Computer and Communications Security*, Alexandria, Virginia, USA, pp.584–597, 2007.

[18] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, pp.90–107, 2008.

[19] D. Boneh, B. Lynn, and H. Shacham, " Short signatures from the weil pairing," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security*, Gold Coast, Australia, pp.514–532, 2001.

[20] H. Q. Wang, Q. H. Wu, B. Qin, *et al.*, "Identity-based remote data possession checking in public clouds," *IET Information Security*, vol.8, no.2, pp.114–121, 2014.

[21] Y. Yu, M. H. Au, G. Ateniese, *et al.*, " Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Transactions on Information Forensics and Security*, vol.12, no.4, pp.767–778, 2017.

[22] J. G. Li, H. Yan, and Y. C. Zhang, " Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, vol.14, no.1, pp.71–81, 2021.

[23] D. B. He, N. Kumar, S. Zeadally, *et al.*, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," *IEEE Transactions on Industrial Informatics*, vol.14, no.3, pp.1232–1241, 2018.

[24] Y. N. Qi, X. Tang, and Y. F. Huang, " Enabling efficient

batch updating verification for multi-versioned data in cloud storage," *Chinese Journal of Electronics* , vol.28, no.2, pp.377–385, 2019.

[25] G. Prakash, M. Prateek, and I. Singh, "Secure public auditing using batch processing for cloud data storage," in *Proceedings of International Conference on Smart System, Innovations and Computing*, Jaipur, India, pp.137–148, 2018.

[26] D. B. He, S. Zeadally, and L. B. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal* , vol.12, no.1, pp.64–73, 2018.

[27] C. M. Tang and X. J. Zhang, "A new publicly verifiable data possession on remote storage," *The Journal of Supercomputing*, vol.75, no.1, pp.77–91, 2019.

[28] X. J. Zhang, J. Zhao, C. X. Xu, *et al.*, "CIPPPA: Conditional identity privacy-preserving public auditing for cloud-based WBANs against malicious auditors," *IEEE Transactions on Cloud Computing*, vol.9, no.4, pp.1362–1375, 2021.

[29] Y. Zhang, J. Yu, R. Hao, *et al.*, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Transactions on Dependable and Secure Computing*, vol.17, no.3, pp.608–619, 2020.

[30] A. Rehman, L. Jian, M. Q. Yasin, *et al.*, "Securing cloud storage by remote data integrity check with secured key generation," *Chinese Journal of Electronics* , vol.30, no.3, pp.489–499, 2021.

[31] L. X. Huang, J. L. Zhou, G. X. Zhang, *et al.*, "Certificateless public verification for data storage and sharing in the cloud," *Chinese Journal of Electronics* , vol.29, no.4, pp.639–647, 2020.

[32] Y. J. Wang, Q. H. Wu, B. Qin, *et al.*, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. on Information Forensics and Security*, vol.12, no.4, pp.940–952, 2017.

[33] H. Yan, J. G. Li, J. G. Han, *et al.*, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Transactions on Information Forensics and Security*, vol.12, no.1, pp.78–88, 2017.

[34] S. Thokchom and D. K. Saikia, "Privacy preserving integrity checking of shared dynamic cloud data with user revocation," *Journal of Information Security and Applications*, vol.50, article no.102427, 2020.

[35] A. De Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proceedings of IEEE Symposium on Computers and Communications*, Kerkyra, Corfu, Greece, pp.850–855, 2011.

LI Yumei was born in Shandong Province, China. She received the Ph.D. degree in mathematical sciences from Nanjing Normal University. She is currently a Lecturer in the Hubei University of Technology. Her research interests include linearly homomorphic signature and cloud storage security. (Email: leamergo@163.com)

ZHANG Futai (corresponding author) received the B.S. and M.S. degrees in mathematics from Shaanxi Normal University, China, and Ph.D. degree from Xidian University, China. He is currently a Professor at Fujian Normal University. His main research interests include cryptography and applications of cryptography in cyberspace security. (Email: futai@fjnu.edu.cn)