Efficient Identity-based Provable Multi-Copy Data Possession in Multi-Cloud Storage

Jiguo Li, Hao Yan, and Yichen Zhang

Abstract—To increase the availability and durability of the outsourced data, many customers store multiple copies on multiple cloud servers. To guarantee the integrity of multi-copies, some provable data possession (PDP) protocols for multi-copy are presented. However, most of previous PDP protocols consider all copies to be stored on only one cloud storage server. In some degree, multi-copy makes little sense in such circumstance. Furthermore, many PDP protocols depend on the technique of public key infrastructure (PKI), which suffers many types of security vulnerabilities and also brings heavy communicational and computational cost. To increase the security and efficiency, we provide a novel identity-based PDP scheme of multi-copy on multiple cloud storage servers. In our scheme, all copies are delivered to different cloud storage servers, which work cooperatively to store the customer's data. By the homomorphic verifiable tags, the integrity of all copies can be checked simultaneously. The system model and security model of our scheme are provided in the paper. The security for our scheme is proved based on the computation Diffie-Hellman (CDH) hard problem. Analysis and experimental evaluation show that our scheme is efficient and practical. The proposed scheme is the first identity-based PDP scheme for multi-copy and multi-cloud servers.

Index Terms—Cloud computing, data integrity checking, multi-copy, multi-cloud servers, security, efficiency

1 Introduction

RECENTLY cloud computing as a new computation model [1] attracts lots of attention. It develops so fast that more and more IT corporations such as Microsoft, IBM, Amazon, Huawei, Alibaba etc. have built their own cloud computing service and opened the service to the world. To date, it is much expensive to maintain the data locally than outsourcing large amounts of data on cloud storage server. Because the later only requires paying the rent of service, the former not only needs to buy and maintain expensive IT infrastructure but also needs other corresponding investment. Therefore, it is a popular trend to store data on cloud storage server.

However, once the data is uploaded on cloud server, the data owner loses the direct control of the data. How to guarantee security and privacy is a challenging problem in cloud storage [2-12]. Although cloud server promises that the data is guaranteed well, some unreliable hardware or software may generate exceptions to destroy the data integrity [13,14]. Furthermore, the cloud server is not completely trusted either. To save storage resource, the cloud server may delete the data accessed rarely without notifying the data owner. It is very possible that the data is tampered but the data owner doesn't know that. So clients need an effective method to verify the integrity of their data.

Provable data possession (PDP) [15] is an important technique for the data owner to check whether the data is correctly maintained on remote cloud server without downloading the data. Although some different PDP schemes [15-42] for different circumstance have been proposed, the core idea of them is almost the same. Specifically, to verify data integrity, the verifier launches a challenge to cloud server. Upon receiving the challenge information, the cloud server calculates an integrity proof by the data and the corresponding metadata. If the proof passes the verifier's verification, the data is proved to be intact. To make the checking result fair, the third party auditor (TPA) trusted by both the cloud server and the data owner often conducts the data integrity auditing.

PDP can help the data owner check the data integrity. Once the data is destroyed, the data owner also loses the data forever although he knows the truth. To improve the durability and availability of the data, data owners can generate multiple copies and store all copies on cloud server. If one copy is tampered, the data owner needs to recover the data from other copies. To further decrease risk, the data owner delivers multiple copies to different cloud storage servers. Even if all copies on one cloud storage server are broken, the data owner can get the data from other cloud storage servers. In this case, the PDP scheme must be extended so that all copies on distributed cloud servers can be checked together. To achieve this goal, special attentions should be paid to four challenges carefully. (1) All copies should be checked by one challenge-response interaction. If the copy is verified one by one, the efficiency is very low and it is meaningless to construct PDP scheme for multi-copy. (2) Data copies should be different from each other. If all copies are the same, the cloud servers can collude to cheat the data owner as they share only one copy but claim all copies are

- *J. Li is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China 350117,E-mail: ljg1688@163.com*
- *J. Li is also with Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, China and the College of Computer and Information, Hohai University, Nanjing, China 211100,E-mail: ljg1688@163.com.*
- *H. Yan is with College of Cyber Security, Jingling Institute of Technology, Nanjing, China 211169, E-mail: pxy_hao@163.com.*
- *Y. Zhang is with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China 350117, E-mail: zyc_718@163.com.*

stored. (3) The scheme should support public verification. Currently, the network and society environment is becoming open, so public verification is an important and attractive feature of PDP, which needs to be realized. (4) All cloud servers can work cooperatively to store and maintain all data copies for the data owner. In fact, it is an easy thing, because cloud computing is designed with open architectures and interfaces.

Unfortunately, most of existing schemes [15-34] only consider checking single data's integrity. For multiple copies, they must be run N times to check the integrity of N copies. It is very inefficient. To overcome this problem, some PDP schemes [35-42] for multi-copy have been presented, most of which can verify all copies by only one challenge-response interaction. Although these protocols improve the efficiency greatly, they just deal with the simple case that all copies are stored on one cloud storage server. They are not suitable for the setting of multiple cloud servers. Furthermore, all of them are based on the traditional PKI technique. As we known, PKI is widely used in many applications but it brings big burden for certificate management. Compared with PKI, identity-based cryptography (IBC) [43] doesn't have such problems. IBC uses the user's unique identity as the public key, it doesn't need the certificate to ensure the validity of the public key. As a result, IBC avoids the heavy cost of certificate management. Therefore, it is important to design an identity-based PDP scheme of multi-copy and multi-cloud storage servers.

1.1 Related Works

Ateniese et al. [15] first proposed the concept of PDP, which realizes the "spot-checking" technique to efficiently check the data on cloud server. In PDP, all data is split into numbers of blocks. By randomly checking parts of data, it can get the integrity status of the entire data with high probability. To support data dynamic, Ateniese et al. [18] further presented a scalable PDP scheme, which realizes the data operations of block appending, updating and deleting. To improve efficiency and flexibility, Sebé et al. [17] and Erway et al. [16] presented their PDP schemes based on authenticated skip list and the hardness of large integers factoring respectively. All these schemes are private verification, namely, only the data owner can check the data integrity. To achieve public verification, Wang et al. [19] provided a dynamic PDP scheme with public audition. The advantage of public verification is that anyone can check the data on cloud server, which improves the flexibility for the scheme greatly. Zhu et al. [20] presented a dynamic audit service for data in cloud storage. Yang and Jia [21] provided a new public verification PDP scheme, which supports data dynamic by index table. Ren et al. [22] presented an improved PDP scheme with data dynamic and public verification. Yan et al. [23] presented an efficient RDPC scheme to improve the data dynamic efficiency by an optimized linear index table. In order to solve key escrow issue, Li et al. [24] provided a certificate-less public integrity checking protocol.

To ensure the group data integrity, Wang et al. [25] proposed an integrity checking protocol (Oruta) for remote data shared in a group. In Oruta [25], each block is signed by a ring signature technique to preserve the user's privacy. Unfortunately, Yu et al. [26] proved that Oruta [25] does not resist the attack launched by active adversary. To enhance the security and address the issue of user revocation from the group, Wang et al. [27] presented a new PDP scheme by using proxy re-signature. To protect the data privacy, Hao et al. [28] provided a PDP protocol with data dynamics and public verifiability. Wang et al. [29] presented a public auditing scheme, which the user's data is guaranteed "zero-knowledge" to TPA. To remove the certificate management, Yu et al. [30] provided an identity-based public PDP protocol with data privacy preserving. Zhang et al. [31] presented a PDP scheme, which realized data privacy preserving by the technique of indistinguishability obfuscation. Zhu et al. [32] provided a PDP scheme which aims to cooperatively verify the data stored in the multi-cloud setting. To improve efficiency, Bharati and Patil [33] presented an advanced cooperative PDP scheme in multicloud storage. Wang [34] presented an identity-based PDP scheme applied to multi-cloud setting. These schemes [32-34] considered the situation of multiple cloud servers, but they don't support multiple copies.

In 2008, Curtmola et al. [35] first presented a PDP scheme (MR-PDP) for integrity checking of multiple replicas in cloud servers. However, MR-PDP scheme [35] is not efficient because the replicas need to be checked one by one. Moreover, MR-PDP scheme [35] only supports private verification. To overcome these problems, Hao et al. [36] provided a new public PDP protocol of the multiple replicas with privacy preservation. Following these works, many PDP schemes [37-43] for multiple copies are presented. For example, Barsoum et al. [37] presented a map-based multi-copy PDP scheme based on map-version table. Zhang et al. [41] presented a new dynamic multi-copy PDP scheme by ranked Merkle hash tree. Yi et al. [42] proposed a dynamic distributed PDP scheme with multi-copies in CSP, which designs a algorithm like 'binary search' to retrieve the corrupted data block. In these multi-copy schemes, all copies are stored on one cloud server. They can not solve the integrity checking problem when data copies are distributed on multiple cloud servers. Furthermore, most of them are based on PKI technique. They need to bear the heavy cost of certificate management. In order to improve efficiency, Li et al. [44,45] presented hierarchy access control scheme in cloud computing.

1.2 Motivation and Contributions

Many corporations and organizations use the technique of remote backup to ensure the data availability and durability. They build multiple geographically dispersed data centers to store their important data and enforce the data consistency. If one data center is destroyed, the data can be recovered from other data centers. To reduce the cost of data center building and maintaining, some organizations rent multi public cloud storage services to store different data copies, which achieves almost the same result as remote backup but with much lower investment.

Because the cloud storage service provider is not fully trusted, the data owner has to consider how to verify the integrity of all data copies.

This article aims to solve the problem of data integrity checking for multiple copies on multiple cloud storage servers. To achieve this goal, we provide the first identity-based PDP protocol for multiple copies on distributed cloud storage servers. Our protocol is designed to verify all copies on different cloud servers at one time so as to improve the protocol's efficiency. In our protocol, a homomorphic identity-based tag is generated for each data block of each copy. Once receiving the integrity challenge from TPA, each cloud storage server generates its own proof and all these distinct proofs are aggregated to be the final proof to decrease the communication cost. We give the framework of our scheme and further propose our new construction. According to the formalized security model, we give the detailed security proof of the scheme under random oracle model. The security of the scheme is reduced to the CDH hard problem. Performance evaluation results show that our scheme is efficient and feasible.

1.3 Organization

The rest of article is organized as follows. In Section 2, preliminaries are introduced. In Section 3, our scheme is constructed. The security proof and performance analysis are given in Section 4 and Section 5. We conclude the paper in Section 6.

2 PRELIMINARIES

The preliminary knowledge used in this paper is introduced in this section.

2.1 Bilinear Maps

$\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups with large prime order q. $g \in \mathbb{G}_1$ is a generator. $e: \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map, which has the below properties.

(1) Computability: for $\forall u, v \in \mathbb{G}_1$, there exists an efficient algorithm to calculate the map $e(u,v)$.

(2) Bilinearity: for $\forall a, b \in Z_q^*$, $\forall u, v \in \mathbb{G}_1$, it has $e(u^a, v^b) = e(u,v)^{ab}$.

(3) Non-degeneracy: $\exists u, v \in \mathbb{G}_1$ such that $e(u,v) \neq 1_{\mathbb{G}_2}$.

2.2 Complexity Assumption

Definition 1(Computation Diffie-Hellman (CDH) problem). *Assume that $\mathbb{G}_1$ is multiplicative cyclic groups. Let g be a generator in $\mathbb{G}_1$. Given the tuple (g, g^a, g^b) for the unknown values $a, b \in Z_q^*$, the CDH problem is to calculate g^{ab}.*

Definition 2(CDH assumption). *For any probabilistic polynomial time (PPT) algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ to address the CDH problem of $\mathbb{G}_1$ is negligible, which is defined as below.*

$Adv_{\mathbb{G}_1,\mathcal{A}}^{CDH} = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \xleftarrow{R} Z_q^*] \leq \varepsilon$. Here, ε denotes a negligible value.

2.3 System Model

Referring to [19, 21], the system of our PDP protocol consists of five entities: Key generation center (KGC), data owner, cloud storage server, cloud organizer and TPA.

(1) KGC generates private keys for users. It uses the user's identity to calculate the private key and returns it to the user by secure channel.

(2) Data owner rents the cloud storage service and stores massive data on multiple cloud servers. It generates many different copies of the outsourced file and stores the file copies to different cloud servers. It can be the organization consumer or the individual consumer.

(3) Cloud storage server (CSS) supplies storage service to the users. It owns significant storage capacity and powerful computation ability to maintain user's data.

(4) Cloud organizer (CO) is the combiner of CSS. When storing file copies, the data owner first sends copies to CO. CO distributes different copies to the target CSS according to user's request. When challenging the file integrity, TPA first sends the challenge request to CO. CO distributes the challenge to the corresponding CSS. Upon receiving all the distinct proofs from CSS, CO aggregates them to be the complete proof and sends it to TPA. In real-life, CO is supported by TPA, they can be bound as one service.

(5) TPA verifies the integrity of all outsourced copies on behalf of the data owner. Both the data owner and CSS trust that the TPA has capability and knowledge to honestly perform the verification work.

The system model is demonstrated in Fig. 1. We assume that the CSS and CO are semi-trusted. They follow the protocol rightly, but may lie to the TPA about the data broken. TPA is assumed to be trustful, who is able to honestly perform the data integrity verification and returns the real result to the data owner.

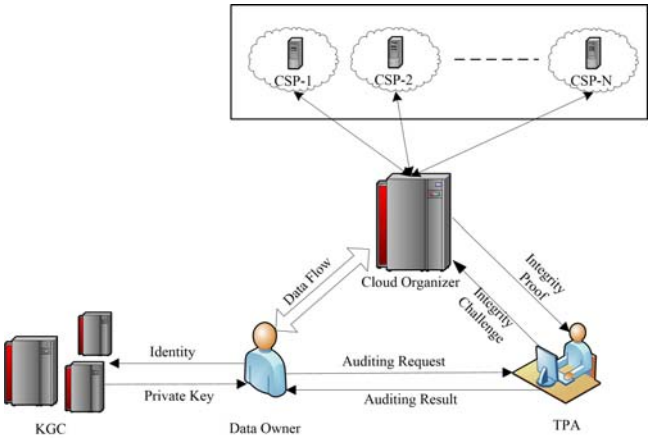

Fig. 1. The system model of our scheme.

The Fig. 2 shows the steps of the file processing and storage. Notably, the dash line between 'copy and tag' and ' CSS ' only means the file copy and its tags are stored together on one CSS, it is not required that the first file copy is stored on the first CSS. The data owner can randomly choose the CSS to store any number of file copies and its tags.

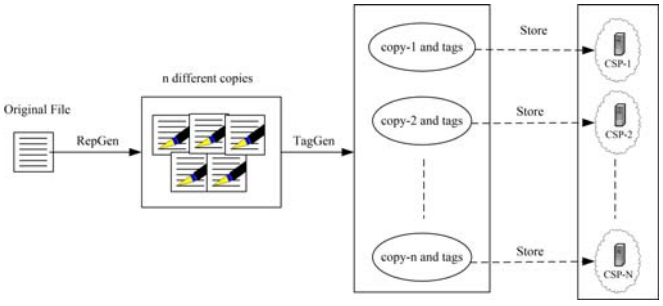

Fig. 2. The process of file storage.

2.4 Scheme Outline

The proposed scheme includes eight polynomial time algorithms: *Setup* , *KeyGen* , *RepGen* , *TagGen* , *Challenge* , *ProofGen* , *ProofAggre* and *Verify* .

$Setup(1^k) \rightarrow (params, msk)$: It is executed by KGC. It inputs a security parameter 1^k , outputs the system public parameter $params$ and the master secret key msk .

$KeyGen(ID, msk) \rightarrow sk_{ID}$: It is run by KGC to produce private keys of the users. It inputs the user's identity $ID \in \{0,1\}^*$ and the master secret key msk , outputs the private key sk_{ID} of the identity ID .

$RepGen(F, N) \rightarrow \mathbb{F}$: The data owner executes this algorithm to produce file copies. It inputs two parameters F and N , where N is the count of the copies and F is the original file to be stored. It outputs N copies of F , namely $\mathbb{F} = \{F_i\}_{1 \le i \le N}$. The data owner uploads the file copies to different CSS, then publishes file name Fid and the number of all copies N .

$TagGen(sk_{ID}, F_i, Cid_i) \rightarrow T_i$. It is run by the data owner to generate block tags for $\mathbb{F}$. It takes data owner's private key sk_{ID} , the file copy F_i and the identity Cid_i of the CSS that F_i will be stored as input, outputs the tag set T_i of F_i . T_i and F_i are sent to the CSS of Cid_i .

$Challenge(Fid) \rightarrow chal$. It is a randomized algorithm executed by the TPA. It inputs the unique file name Fid , outputs challenge $chal$ of the file name Fid .

$ProofGen(FS_i, TS_i, chal) \rightarrow P_i$. Each CSS runs this algorithm to produce the integrity proof for all copies it stored. It takes the file copy set FS_i , the challenge $chal$, and the corresponding tag set TS_i as input, outputs the data integrity proof P_i .

$ProofAggre(\{P_i\}_{(1 \le i \le \xi)}) \rightarrow P$. It is run by CO to aggregate all the integrity proofs from the CSS storing the challenged data (assume ξ CSS in total). The input $\{P_i\}_{(1 \le i \le \xi)}$ is the set of integrity proof from CSS and the output P is the final integrity proof to TPA.

$Verify(ID, chal, P, \{Cid_i\}_{1 \le i \le \xi}) \rightarrow \{1, 0\}$. TPA executes this algorithm to check data integrity by the proof P . It inputs the challenge $chal$, the data owner's identity ID , the integrity proof P and the identity set of CSS, outputs 1 if P passed the verification, otherwise 0 .

2.5 Security Model

To make protocol practical, the scheme must satisfy the following security requirement. (1) All copies stored on cloud servers are verified by TPA without local copy. Specifically, if all CSS, CO and TPA honestly perform the protocol, the integrity of all file copies are verified correctly. (2) If any copy is modified or lost, the CSS can not deceive the TPA. That is, CSS can not forge the valid proof with the broken data.

The security game can capture above security requirements. The game between the challenger $\mathcal{C}$ and the adversary $\mathcal{A}$ is defined as follow.

Setup: The challenger $\mathcal{C}$ executes the *Setup* algorithm to obtain the $params$ and msk . $\mathcal{C}$ keeps msk confidential and transfers $params$ to $\mathcal{A}$.

RepGen: $\mathcal{C}$ executes the algorithm *RepGen* to obtain all copies for the original file.

Queries: $\mathcal{A}$ can make polynomial times queries to $\mathcal{C}$. $\mathcal{C}$ answers the queries as below.

(a) Key Query. $\mathcal{A}$ sends any identity ID_i to query its private key. $\mathcal{C}$ executes the algorithm *KeyGen* to compute the private key sk_i and returns the key to $\mathcal{A}$.

(b) Hash Query. $\mathcal{A}$ sends hash queries to $\mathcal{C}$. $\mathcal{C}$ returns the hash results to $\mathcal{A}$.

(c) Tag Query. $\mathcal{A}$ adaptively chooses any block of any copy to query the corresponding tag under any identity ID_i . $\mathcal{C}$ executes the *TagGen* algorithm to obtain the tag and returns it to $\mathcal{A}$.

ProofCheck: $\mathcal{A}$ executes the algorithm *ProofGen* and *ProofAggre* to generate integrity proof for any block of the file with the identity ID_i . All these blocks have been made 'Tag Query' by $\mathcal{A}$. $\mathcal{A}$ sends the proofs to $\mathcal{C}$. $\mathcal{C}$ executes the *Verify* algorithm to check these proofs and returns the results to $\mathcal{A}$.

Output: Finally, $\mathcal{A}$ outputs a proof P on some blocks of file with the identity ID' . If P passes the integrity verification and is not equal to the real proof, $\mathcal{A}$ wins the game.

Definition 3: An identity-based PDP scheme of multi-copy and multi-cloud servers is secure against untrusted cloud service provider, if for any PPT adversary $\mathcal{A}$, the probability that $\mathcal{A}$ wins the security game mentioned above is negligible.

3 Our Construction

In this section, we give the concrete construction of our efficient identity-based PDP protocol for multi-copy on multi-cloud servers, which supports public verification.

Without loss of generality, we suppose the data owner plans to outsource the file F to cloud server. According to the file size, the data owner splits F into n blocks, i.e., $F = \{m_1, m_2, \cdots, m_n\}$, in which m_i denotes the i-th block of F . The algorithms of our construction are described in detail as follows.

$Setup(1^k) \rightarrow (params, msk)$: On input a security parameter 1^k, KGC selects two cyclic multiplicative groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with prime order q. Let g be a generator of $\mathbb{G}_1$. $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is bilinear map. KGC chooses two different cryptographic hash functions $H_1: \{0,1\}^* \rightarrow \mathbb{G}_1^*$ and $H_2: \{0,1\}^* \rightarrow \mathbb{G}_1^*$, a pseudo-random permutation π: $Z_q^* \times \{1, \cdots, n\} \rightarrow \{1, \cdots, n\}$ and a pseudo-random function (PRF) ϕ: $Z_q^* \times Z_q^* \rightarrow Z_q^*$. Finally, KGC randomly selects $x \in Z_q^*$ as the master secret key and computes the master public key $P_0 = g^x$. KGC publishes the system parameters $params = (q, g, \mathbb{G}_1, \mathbb{G}_2, e, P_0, H_1, H_2, \phi, \pi)$.

$KeyGen(ID, msk) \rightarrow sk_{ID}$: When receiving the user's identity ID, KGC calculates $sk_{ID} = H_1(ID)^x$ as the private key of the user.

$RepGen(F, N) \rightarrow \mathbb{F}$: For any file F, the data owner generates N distinct copies. These copies are different from each other, because if all copies are same, the cloud servers can collude to cheat the data owner as they share only one copy but claim all copies are stored. Thus, we utilize an encryption algorithm $E_K(\cdot)$ with the diffusion property to get different copies. The diffusion property ensures that even a single bit change in plaintext makes completely unpredictable change in ciphertext. Lots of encryption algorithms have such diffusion property such as DES, AES etc. Assume the original file F is split into n blocks, i.e., $F = \{m_1, m_2, \cdots, m_n\}$. The i-th copy is represented as $F_i = \{m_{ij}\}_{(1 \le i \le N, 1 \le j \le n)}$, in which m_{ij} is computed by $m_{ij} = E_K(i \| m_j)$. The encrypted block m_{ij} is further split into s sectors $m_{ij} = \{m_{ijk}\}_{1 \le k \le s}$, so it has $F_i = \{m_{ijk}\}_{1 \le j \le n, 1 \le k \le s}$ where each sector is an element of Z_q.

$TagGen(sk_{ID}, F_i, Cid_i) \rightarrow T_i$: The data owner selects s random values $(\alpha_1, \cdots, \alpha_s) \in Z_q^*$ and calculates $u_k = g^{\alpha_k}$ $(1 \le k \le s)$. The data owner chooses a random value $\lambda \in Z_q^*$ and computes the tag for each block m_{ij} of the copy F_i by the equation:

$$T_{ij} = sk_{ID} \cdot (H_2(Fid \| Cid_i \| i \| j) \cdot g^{\sum_{k=1}^{s} \alpha_k m_{ijk}})^{\lambda} \quad (1).$$

The symbol Cid_i is the identity of the CSS that the i-th copy is stored on. The data owner repeats the equation (1) for $N \times n$ times to get the tag set T of all copies and creates a table as Table I to record the relationships of Cid_i and the corresponding copy indexes. For example:

TABLE I Record Table

ID of CSS	Index of Copy
Cid_1	1,8,12
Cid_2	3,5
……	……

The data owner computes $R = g^{\lambda}$ and selects a signing algorithm Sig such as BLS [46] to compute tag of the whole file as $T_{FID} = Sig(R \| u_1 \| \cdots \| u_s \| Fid)$. Finally, the data owner sends the record table and $(R, \{u_i\}_{1 \le i \le s}, T_{FID})$ to the CO. The CO sends the file copies and their tags to the corresponding CSS. Besides, each tag is verified by the following equation:

$$e(T_{ij}, g) = e(H_1(ID), P_0) \cdot e(H_2(Fid \| Cid_i \| i \| j) \cdot \prod_{k=1}^{s} u_k^{m_{ijk}}, R) \quad (2)$$

$Challenge(Fid) \rightarrow chal$: To check the integrity of all copies named Fid. TPA randomly picks the values (k_1, k_2), where $k_1, k_2 \in Z_q^*$ are two random seeds of pseudo-random permutation and pseudo-random function respectively. Together with the challenged block count $c \in [1, n]$, the TPA sends $chal = (c, k_1, k_2)$ and the file name Fid to CO. CO searches the record table and sends the challenge to every CSS which stores at least one copy of the challenged file. Assume there are ξ CSS who stores the challenged copies. Then CO sends $chal$ to the ξ CSS.

$ProofGen(FS_i, TS_i, chal) \rightarrow P_i$: Upon receiving the $chal = (c, k_1, k_2)$ from CO, the CSS identified by Cid_i calculates his own proof. Assume the index set of file copies FS_i on Cid_i is CT_i. Note that $\sum_{l=1}^{\xi} |CT_l| = N$. Cid_i first gets the challenge set $C = \{(v_j, a_j)\}$, where $v_j = \pi(k_1, j)$, $a_j = \phi(k_2, j)$ for $1 \le j \le c$. For $\forall \beta \in CT_i$, the CSS with Cid_i computes $\sigma_\beta = \prod_{(v_t, a_t) \in C} T_{\beta v_t}^{a_t}$, $M'_{\beta k} = \sum_{(v_t, a_t \in C)} a_t m_{\beta v_t k}$ $(1 \le k \le s)$. Then, the CSS with Cid_i computes his proof: $\sigma_i = \prod_{\beta \in CT_i} \sigma_\beta$, $M_{i,k} = \sum_{\beta \in CT_i} M'_{\beta k}$ and sends the proof $P_i = (\sigma_i, \{M_{i,k}\}_{(1 \le k \le s)})$ to CO.

$ProofAggre(\{P_i\}_{(1 \le i \le \xi)}) \rightarrow P$: Upon receiving all proofs from ξ CSS, CO aggregates these distinct proofs and outputs the final proof. CO computes $\sigma = \prod_{i=1}^{\xi} \sigma_i$ and $M_k = \sum_{i=1}^{\xi} M_{i,k}$. CO returns the proof $P = (\sigma, \{M_k\}_{1 \le k \le s}, R, \{u_k\}_{1 \le k \le s}, T_{FID})$ to TPA. The detailed process of challenge is shown in Fig. 3.

$Verify(ID, chal, P, \{Cid_i\}_{1 \le i \le \xi}) \rightarrow \{1, 0\}$: Upon receiving P from the CO, TPA first checks if the T_{FID} is a valid signature of the data owner for the message $R \| u_1 \| \cdots \| u_s \| Fid$. If not, the TPA refuses the proof and outputs '0'. Otherwise, TPA computes the set $C = \{(v_i, a_i)\}$, in which $v_i = \pi(k_1, i)$, $a_i = \phi(k_2, i)$ $(1 \le i \le c)$. TPA verifies whether the following equation holds.

$$e(\sigma, g) = e(H_1(ID)^{\sum_{(v_t, a_t) \in C} a_t}, P_0^N) \cdot e(\prod_{i=1}^{N} \prod_{(v_t, a_t) \in C} H_2(Fid \| Cid_i \| i \| v_t \|)^{a_t} \cdot \prod_{k=1}^{s} u_k^{M_k}, R) \quad (3)$$

If it holds, it outputs '1', otherwise outputs '0'.

If CSS, CO and TPA honestly execute the protocol, the integrity of file copies is verified by the equation (3). The correctness of the equation (3) is shown as follow.

$$e(\sigma,g)=e(\prod_{l=1}^{\xi}\sigma_l,g)=e(\prod_{l=1}^{\xi}\prod_{\beta\in CT_l}\sigma_\beta,g)$$

$$=e(\prod_{i=1}^{N}\sigma_i,g)=e(\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}T_{iv_t}^{a_t},g)$$

$$=e(\prod_{i=1}^{N}(\prod_{(v_t,a_t)\in C}(H_1(ID)^x\cdot(H_2(Fid\|Cid_i\|i\|v_t)\cdot g^{\sum_{k=1}^{s}\alpha_k m_{iv_tk}})^{\lambda})^{a_t},g)$$

$$=e(\prod_{i=1}^{N}H_1(ID)^{x\sum_{(v_t,a_t)\in C}a_t},g)\cdot$$

$$e(\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}(H_2(Fid\|Cid_i\|i\|v_t)\cdot g^{\sum_{k=1}^{s}\alpha_k m_{iv_tk}})^{a_t\lambda}),g)$$

$$=e(H_1(ID)^{\sum_{(v_t,a_t)\in C}a_t},g^{xN})\cdot$$

$$e(\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}H_2(Fid\|Cid_i\|i\|v_t)^{a_t}\cdot\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}(g^{\sum_{k=1}^{s}\alpha_k m_{iv_tk}})^{a_t},g^{\lambda})$$

$$=e(H_1(ID)^{\sum_{(v_t,a_t)\in C}a_t},P_0^N)\cdot$$

$$e(\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}H_2(Fid\|Cid_i\|i\|v_t)^{a_t}\cdot\prod_{i=1}^{N}\prod_{k=1}^{s}u_k^{\sum_{(v_t,a_t)\in C}a_t m_{iv_tk}},g^{\lambda})$$

$$=e(H_1(ID)^{\sum_{(v_t,a_t)\in C}a_t},P_0^N)\cdot$$

$$e(\prod_{i=1}^{N}\prod_{(v_t,a_t)\in C}H_2(Fid\|Cid_i\|i\|v_t)^{a_t}\cdot\prod_{k=1}^{s}u_k^{M_k},R)$$

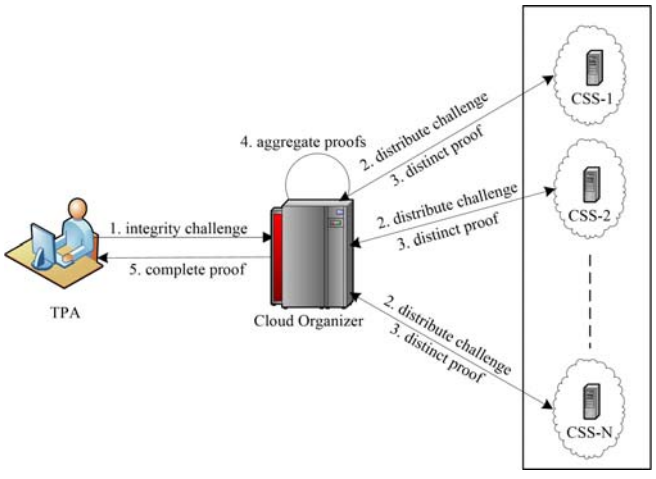

Fig. 3 The challenge process

4 Security Analysis

We prove the security of the protocol in this section. We first show that the single tag is unforgeable. Then we show the complete proof is unforgeable, which covers the resistance for the collusion of CSS and CO.

Theorem 1: If there exists a PPT adversary $\mathcal{A}$, who makes H_1-Query, Key-Query, H_2-Query and Tag-Query at most q_{H_1}, q_k, q_{H_2}, q_T times respectively and wins the security game described in section 2.5 at the advantage ε within time t, then a simulator $\mathcal{B}$ breaks the CDH assumption at the probability $\varepsilon'\geq\varepsilon/((q_k+q_T)\cdot 2e)$ within the time $t'\leq t+O(q_{H_1}+q_k+q_{H_2}+q_T)$.

Proof: Given CDH instance $(g,\mathbb{G}_1,g^a,g^b)$. If the adversary $\mathcal{A}$ wins the security game in section 2.5 at non-negligible probability, $\mathcal{B}$ is able to compute g^{ab} with non-negligible probability using the capability of $\mathcal{A}$. $\mathcal{B}$ simulates each interaction step with $\mathcal{A}$ as follow.

Setup: $\mathcal{B}$ generates the public parameters and sets $P_0=g^a$ in which the master key is the unknown value a implicitly. Furthermore, $\mathcal{B}$ randomly selects $\lambda\in Z_q^*$ and s random values $(\alpha_1,\cdots,\alpha_s)\in Z_q^*$. $\mathcal{B}$ computes $R=g^{\lambda}$ and $u_k=g^{\alpha_k}$ $(1\leq k\leq s)$. $\mathcal{B}$ sends the public parameters, R and $\{u_k\}_{1\leq k\leq s}$ to $\mathcal{A}$.

RepGen: $\mathcal{B}$ executes the algorithm *RepGen* to obtain all copies of the original file. $\mathcal{B}$ randomly sets the storage place of these copies and gives the information to $\mathcal{A}$.

H_1-Query: $\mathcal{A}$ adaptively carries out the H_1-Query with identity ID. $\mathcal{B}$ keeps a list $L_1=\{(ID,h_1,Q_1,\tau)\}$ to store query results. If L_1 contains the ID, $\mathcal{B}$ finds the tuple (ID,h_1,Q_1,τ) and returns Q_1 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ selects a random value $h_1\in Z_q^*$ and flips a coin $\tau\in\{0,1\}$ that $\tau=1$ with the probability γ and $\tau=0$ with the probability $1-\gamma$. If $\tau=0$, $\mathcal{B}$ computes $Q_1=g^{h_1}$. If $\tau=1$, $\mathcal{B}$ sets $Q_1=(g^b)^{h_1}$. $\mathcal{B}$ responses Q_1 to $\mathcal{A}$ and adds the new tuple (ID,h_1,Q_1,τ) to L_1.

Key-Query: For identity ID, $\mathcal{A}$ adaptively executes the Key-Query. $\mathcal{B}$ checks whether (ID,h_1,Q_1,τ) exists in L_1. If not, $\mathcal{B}$ makes the H_1-Query for ID itself. After retrieving the corresponding (ID,h_1,Q_1,τ) from L_1, $\mathcal{B}$ checks the value of τ. If $\tau=0$, $\mathcal{B}$ computes and returns $(Q_1)^a=(g^{h_1})^a=(g^a)^{h_1}$ to $\mathcal{A}$. Otherwise, $\mathcal{B}$ aborts.

H_2-Query: At any time, $\mathcal{A}$ submits (Fid,Cid_i,i,j) to $\mathcal{B}$ for the H_2-Query. To response the query, $\mathcal{B}$ keeps a H_2-list with tuples (Fid,Cid_i,i,j,Q_2). If $(Fid,Cid_i,i,j)\in H_2$-list, $\mathcal{B}$ finds the tuple and returns Q_2 to $\mathcal{A}$. Otherwise, $\mathcal{B}$ randomly chooses $Q_2'\in\mathbb{G}_1$ and sends Q_2' to $\mathcal{A}$. Then $\mathcal{B}$ adds the new tuple (Fid,Cid_i,i,j,Q_2') to the H_2-list.

Tag-Query: At any time, $\mathcal{A}$ is able to query the tags of the blocks in any copy for any identity ID. $\mathcal{A}$ sends the tuple (Fid,Cid_i,i,j,ID) to $\mathcal{B}$ for querying the tag of m_{ij}. On receiving the tag query, $\mathcal{B}$ first searches the tuple (ID,h_1,Q_1,τ) from H_1-list and (Fid,Cid_i,i,j,Q_2) from H_2-

list. If they do not exist, $\mathcal{B}$ executes H_1-Query and H_2-Query to get them. If $\tau = 0$, $\mathcal{B}$ calculates the tag by $T_{ij} = (g^a)^{h_1} \cdot (Q_2 \cdot g^{\sum_{k=1}^{s} m_{ijk}})^r$. Otherwise, $\mathcal{B}$ aborts.

Output: Finally, the adversary $\mathcal{A}$ outputs a forged tag T_{ij}' of block m_{ij}' for the identity ID'. It is required that the block m_{ij}' should not be executed the Tag-Query with ID'.

Analysis: If $\mathcal{A}$ succeeds in winning the game, $\mathcal{B}$ can obtain

$$e(T_{ij}', g) = e(H_1(ID), P_0) \cdot e(H_2(Fid \| Cid_i \| i \| j) \cdot \prod_{k=1}^{s} u_k^{m_{ijk}'}, R)$$

according to the verification equation (2). $\mathcal{B}$ recovers the (ID', h_1', Q_1', τ') from L_1 and checks the value of τ'. If $\tau' = 0$, $\mathcal{B}$ aborts. Otherwise, $\mathcal{B}$ searches the tuple $(Fid', Cid_{i'}, i', j', Q_2')$ from L_2. By the verification equation mentioned above, $\mathcal{B}$ can obtain $e(T_{ij}', g) = e(g^{bh_1'}, g^a) \cdot e(Q_2' \cdot \prod_{k=1}^{s} u_k^{m_{ijk}'}, g^r)$. Thus, the result for the given CDH instance is $g^{ab} = \left(\frac{T_{ij}'}{(Q_2' \prod_{k=1}^{s} u_k^{m_{ijk}'})^r} \right)^{1/h_1'}$.

We can see that if $\mathcal{B}$ does not abort in the process above, $\mathcal{A}$ can not distinguish $\mathcal{B}$ or $\mathcal{C}$ in the game. It is easy to see only in the phase of Key-Query and Tag-Query, $\mathcal{B}$ may abort. Thus, the probability that $\mathcal{B}$ doesn't abort in the game is beyond $(1-\gamma)^{q_k+q_T}$. Then $\mathcal{B}$ computes the value g^{ab} with at least probability of $\varepsilon' \geq \varepsilon \cdot \gamma \cdot (1-\gamma)^{q_k+q_T} \geq \varepsilon / ((q_k+q_T) \cdot 2e)$. The required time is $t' \leq t + O(q_{H_1} + q_k + q_{H_2} + q_T)$.

Next, we show the aggregated proof can not be forged to deceive the TPA.

Theorem 2: If hash function is collision-resistance and the selected signing algorithm is secure, the probability of forging complete proof to pass the verification without the real data is negligible.

Proof: From the complete proof $P = (\sigma, \{M_k\}_{1 \leq k \leq s}, R, \{u_k\}_{1 \leq k \leq s}, T_{FID})$, we see that if the signing algorithm selected in tag generation is secure, the signature T_{FID} can ensure the correctness of R, $\{u_k\}_{1 \leq k \leq s}$. That's to say T_{FID}, R and $\{u_k\}_{1 \leq k \leq s}$ in the proof can not be forged. Thus, the forged parts of the proof only happen in $(\sigma, \{M_k\}_{1 \leq k \leq s})$. Let $chal = (c, k_1, k_2)$ be the challenge information, we compute the challenged block indices by $v_i = \pi(k_1, i)$ and the parameters by $a_i = \phi(k_2, i)$. Suppose the forged aggregate proof is $P' = (\sigma', \{M'_k\}_{1 \leq k \leq s})$. Since the forged proof can pass the verification, we get the verification equation $e(\sigma', g) = e(H_1(ID)^{\sum_{(v_t, a_t) \in C} a_t}, P_0^N) \cdot e(\prod_{i=1}^{N} \prod_{(v_t, a_t) \in C} H_2(Fid \| Cid_i \| i \| v_t \|)^{a_t} \cdot \prod_{k=1}^{s} u_k^{M'_k}, R)$. Assume the real proof for the $chal = (c, k_1, k_2)$ is $P = (\sigma, \{M_k\}_{1 \leq k \leq s})$, so we can also get the equation

$$e(\sigma, g) = e(H_1(ID)^{\sum_{(v_t, a_t) \in C} a_t}, P_0^N) \cdot e(\prod_{i=1}^{N} \prod_{(v_t, a_t) \in C} H_2(Fid \| Cid_i \| i \| v_t \|)^{a_t} \cdot \prod_{k=1}^{s} u_k^{M_k}, R)$$

Compared the two equations above, it is easy to get $\sigma' = \sigma$ if each $M'_k = M_k$, which is contrast to the assumption. Thus, there is at least one $M'_k \neq M_k$. In this case, according to the Theorem 1, single tag can't be forged, so it must have $\sigma' = \sigma$. Otherwise, we get the forged tag by challenging only one block. Therefore, to make the two equations hold, it has $\sigma' = \sigma$ and at least one $M'_k \neq M_k$. Compared with the above two equations, we see that $\prod_{k=1}^{s} u_k^{M_k}$ is equal to $\prod_{k=1}^{s} u_k^{M'_k}$. So it has $g^{\sum_{k=1}^{s} \alpha_k M_k} = g^{\sum_{k=1}^{s} \alpha_k M'_k}$, i.e., $\sum_{k=1}^{s} \alpha_k M_k = \sum_{k=1}^{s} \alpha_k M'_k$, i.e., $\sum_{k=1}^{s} \alpha_k (M_k - M'_k) = 0$. Without loss of generality, we suppose there are β $(1 \leq \beta \leq s)$ different $M'_k \neq M_k$. Then, the count of tuples $(\alpha_1, \cdots, \alpha_s)$ satisfying the property is at most $q^{\beta-1}$. Since $(\alpha_1, \cdots, \alpha_s)$ are random values and unknown to CSS, the probability of the equation $\sum_{k=1}^{s} \alpha_k (M_k - M'_k) = 0$ holding is less than $\frac{q^{\beta-1}}{q^s} \leq \frac{q^{\beta-1}}{q^\beta} = \frac{1}{q}$, which is negligible.

5 Performance Analysis

The performance analysis and experiment results of the proposed scheme are presented in this section.

5.1 Performance Evaluation

From the aspects of computation cost and communication cost, we summary the performance of our scheme as follows.

Computation cost: Let T_p, $T_{\exp}$, T_{mul} represent the operations of pairing, exponentiation and multiplication on group $\mathbb{G}_1$ respectively. Other operations such as hash, addition and multiplication in Z_q are omitted, since the cost of them is nearly negligible. Assume that the data owner stores N copies in total on ξ cloud servers, and the copy number of each CSS is $|CT_i|$. Each copy has n blocks and each block is split into s sectors. TPA challenges c blocks. The algorithm *KeyGen* costs one $T_{\exp}$ to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCC.2019.2929045, IEEE Transactions on Cloud Computing

generate user's private key. To generate N copies, the algorithm *RepGen* needs to run encryption algorithm E_K for N times. Assume the computational cost of E_K is T_E, so the cost of *RepGen* is $N \cdot T_E$. To generate all the tags for all copies, the algorithm *TagGen* costs $N \cdot n \cdot (2T_{\exp} + 2T_{mul}) + (s+1) \cdot T_{exp}$. The algorithm *Challenge* only causes negligible cost. Each CSS runs the *ProofGen* to generate the integrity proof. Assume the CSS stores L copies, so the *ProofGen* needs $L \cdot c \cdot (T_{\exp} + T_{mul})$ computation cost. CO runs the algorithm *ProofAggre* to aggregate all the distinct proofs, which costs $\xi \cdot T_{mul}$. To check the file integrity, TPA runs the algorithm *Verify* which needs the computation cost is $3T_p + (c \cdot N + s + 3)T_{exp} + (c \cdot N + s)T_{mul}$. Moreover, we compare the scheme CPDP [32] with our scheme from different aspects and the results are shown in the table II. For simplicity, we list the computational cost of only one operation in each phase, for instance generating one tag, performing challenge and verification only once.

Communication Cost: To audit the integrity of the file on cloud server, TPA submits the challenge information $chal = (c, k_1, k_2)$ and the file name Fid to CO. The communication cost of the challenge request is $(logc + 2 | q | + | Fid |)$ bits. The CO returns the integrity proof $P = (\sigma, \{M_k\}_{1 \le k \le s}, R, \{u_k\}_{1 \le k \le s}, T_{FID})$ to TPA. So the communication cost of challenge response is $(s | q | + (2+s) | \mathbb{G}_1 | + | T_{FID} |)$ bits.

TABLE II Comparison with the scheme CPDP

Schemes	CPDP [32]	Our scheme
Tag-Gen	$(s+2)T_{\exp} + sT_{mul}$	$2T_{\exp} + 2T_{mul}$
Proof-Gen	$(c+2)T_{\exp} + T_p + c \cdot T_{mul}$	$c \cdot (T_{\exp} + T_{mul})$
Proof-Aggr	$(\xi+2)T_{\exp} + 3\xi \cdot T_{mul}$	$\xi \cdot T_{mul}$
Verify	$3T_p + (c+s)T_{exp} + (c+s)T_{mul}$	$3T_p + (c \cdot N + s + 3)T_{exp} + (c \cdot N + s)T_{mul}$
Multi-copy	No	Yes
Multi-CSS	Yes	Yes
ID-Based	No	Yes

5.2 Experimental Results

Based on GMP Library [47] and PBC Library [48], we implement a prototype of our scheme. We perform lots of experiments on the VMware workstation 10 with 1 CPU, 20G ROM and 4G RAM. We choose the operating system Ubuntukylin-15.10-desktop-i386. The VMware workstation is hosted on the Lenovo Laptop L440, the configuration of which is Core i7-4712MQ@2.3GHz CPU, 8G Ram and Win7 operation system. In our experiments, we choose the file "a.param" as the parameters of PBC library.

We evaluate the computation cost for tag generation. In this experiment, we create a 4M file and generate 20 copies. Each copy has 1000 blocks and each block has 200 sectors. Without loss of generality, we change the block count from 100-600 with an increment of 100 in each test. The results are shown in Fig.4. As observed, our scheme has large advantages on generating block tags compared with CPDP. To generate 600 tags with 200 sectors only needs about 3.6 seconds by our scheme, but costs about 350 seconds by CPDP.

The algorithms *ProofGen* and *Verify* are executed by CSS and TPA respectively, and also are two main parts in such a challenge-response model protocol. We perform the second experiment to evaluate the computational cost

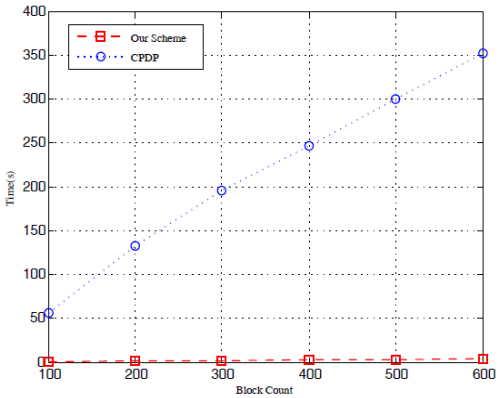

Fig. 4 Computation cost of tag generation

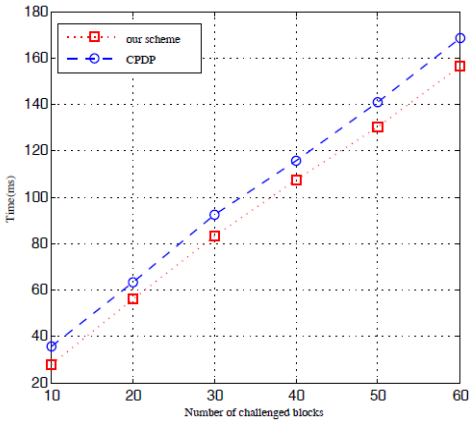

Fig. 5 Computation cost of proof generation

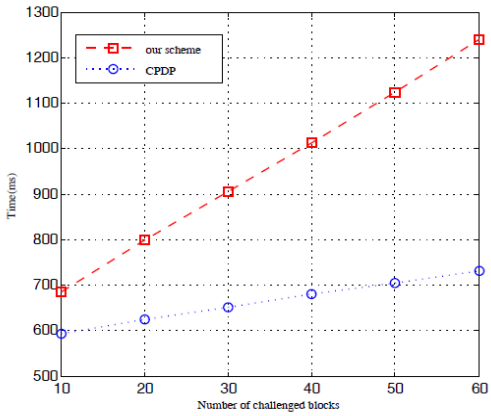

Fig. 6 Computation cost of verification

of them. Here, we distribute the 20 file copies to 5 CSS, each CSS has averagely 4 copies. Like the first experiment, we also implement the CPDP scheme under the same experiment setting. We challenge different blocks from 10-100 with an increment of 10. The results are shown in Fig.5 and Fig.6. From Fig. 5, we see that our scheme and CPDP have the similar computation cost for tag generation. Both of them are linearly with the number of challenged blocks. However, our scheme is a little more efficient than CPDP. The Fig.6 shows that our scheme needs bigger cost in verification than CPDP. That is because our scheme is multi-copy model, but CPDP is not. Theoretically, to challenge one block in our scheme is similar to challenge N blocks in CPDP. The efficiency of CPDP should be N times better than our scheme. However, the experiment result shows that our scheme is far better than theoretical analysis. Overall, our scheme is efficient and practical for real applications.

6 Conclusions

In this article, we present a new identity-based PDP scheme for checking multi-copy on distributed cloud servers. The proposed scheme is proved secure under the CDH assumption. Our scheme is designed to verify the integrity of multiple copies on multiple cloud servers within one challenge-response interaction. Moreover, the structural advantage of identity-based cryptography makes our scheme more efficient and secure due to removing certificate management problem in PKI. In addition, our scheme supports the feature of public verification. Theoretical analysis and experiment results show that the new scheme is very efficient and practical. Our future work is to upgrade the scheme to support data dynamic and batch verification.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (U1736112, 61772009, 61672207), Jiangsu Provincial Natural Science Foundation of China (BK20161511), Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, NJUPT.

References

[1] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,'' *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599-616, 2009.

[2] J. Li, W. Yao, Y. Zhang, H. Qian and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Service Comput.*, 2017, 10(5): 785-796.

[3] C. Zuo, J. Shao, J. K. Liu, G. Wei and Y. Ling, "Fine-grained two-factor protection mechanism for data sharing in cloud storage," *IEEE Trans. Information Forensics and Security*, vol. 13, no 1, pp. 186-196, 2018.

[4] J. Li, W. Yao, J. Han, Y. Zhang and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage, " *IEEE Systems Journal*, 2017, DOI: 10.1109/JSYST.2017.2667679.

[5] J. Ning, X. Dong, Z. Cao, L. Wei and X. Lin, "White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes," *IEEE Transactions on Information Forensics and Security*, vol.10, no 6, pp. 1274-1288, 2015.

[6] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Service Comput.*, vol. 10, no. 5, pp. 715-725, Sept.-Oct. 2017.

[7] Z. Liu, Z. Cao and Duncan S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76-88, 2013

[8] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *International Journal of Communication Systems*, vol. 30, no. 1, Art. no. e2942, Jan. 2017.

[9] J. Han, W. Susilo, Y. Mu, J. Zhou, M. Au, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Transactions on Information Forensics and Security*, 2015, 10(3): 665-678

[10] J. Li, Q. Yu and Y. Zhang, "Key-policy attribute-based encryption against continual auxiliary input leakage," *Information Sciences*, 2019, 470: 175–188

[11] H. Qian, J. Li, Y. Zhang and J. Han, "Privacy preserving personal health record using multi-authority attribute-based encryption with revocation," *Int. J. Inf. Secur.*, vol. 14, no. 6, pp. 487-497, 2015.

[12] H. Li, H. Zhu, S. Du, X. Liang, X. Shen, "Privacy leakage of location sharing in mobile social networks: attacks and defense," *IEEE Trans. on Dependable and Secure Computing*, 2016, DOI: 10.1109/TDSC.2016.2604383

[13] M. Ali, S.U. Khan and A.V. Vasilakos, "Security in cloud computing: Opportunities and challenges,'' *Inf. Sci.* vol. 305, no. 1, pp. 357–383, 2015.

[14] J. Li, Y. Wang, Y. Zhang and J. Han. "Full verifiability for outsourced decryption in attribute based encryption," IEEE Transactions on Services Computing, 2017, DOI: 10.1109/TSC.2017.2710190

[15] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, ''Provable data possession at untrusted stores,'' in *Proc. 14th ACM Conf. on Comput. and Commun. Security (CCS)*, 2007, pp. 598-609.

[16] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession,'' in *Proc. 16th ACM Conf. on Comput. and Commun. Security (CCS), 2009*, pp. 213-222.

[17] F. Sebé, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Trans. Knowledge and Data Eng.*, vol. 20, no. 8, pp. 1034-1038, Aug. 2008.

[18] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, ''Scalable and efficient provable data possession,'' in *Proc. 4th Int'l Conf. Security and Privacy in Commun. Netw. (SecureComm)*, 2008, pp. 1-10.

[19] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, ''Enabling public auditability and data dynamics for storage security in cloud computing,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847-859, May, 2011.

[20] Y. Zhu, G.J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, ''Dynamic audit services for outsourced storage in clouds,'' *IEEE Trans. on Services Computing*, vol. 6, no. 2, pp. 227-238, 2013.

[21] K. Yang and X. Jia, ''An efficient and secure dynamic auditing protocol for data storage in cloud computing,'' *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717-1726, 2013.

[22] Z. Ren, L. Wang, R. Deng and R. Yu, ''Improved fair and dynamic provable data possession supporting public verification,'' *Wuhan University Journal of Natural Sciences,* vol. 18, no. 4, pp. 248-354, 2013.

[23] H. Yan, J. Li, J. Han and Y. Zhang, ''A novel efficient remote

data possession checking protocol in cloud storage, '' *IEEE Trans. Inf. Foren. and Sec.*, vol. 12, no. 1, pp. 78-88, 2017.

[24] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Transactions on Services Computing*, 2017, 10.1109/TSC.2018.2789893.

[25] B. Wang, B. Li, and H. Li, ''Oruta: privacy-preserving public auditing for shared data in the cloud, '' *IEEE Trans. Cloud Computing*, vol. 2, no. 1, pp. 43–56, 2014.

[26] Y. Yu, L. Niu, G. Yang, Y. Mu and W. Susilo, ''On the security of auditing mechanisms for secure cloud storage, '' *Future Gener. Comp. Sys.*, no. 30, pp. 127-132, 2014.

[27] B. Wang, B. Li, and H. Li, ''Panda: public auditing for shared data with efficient user revocation in the cloud,'' *IEEE Trans. Service Comput.*, vol. 8, no. 1, 2015.

[28] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 9, pp. 1432–1437, Sep. 2011.

[29] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[30] Y. Yu, M.H Au, G. Ateniese, X. Huang, W. Susilo, Y. D and G. Min, ''Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage,'' *IEEE Trans. Inf. Foren. and Sec.*, vol. 12, no 4, pp. 767-778, APRIL 2017.

[31] Y. Zhang, C. Xu, X. Liang, H. Li, Y. Mu and X. Zhang, ''Efficient public verification of data integrity for cloud storage systems from indistinguishability Obfuscation,'' *IEEE Trans. Inf. Foren. and Sec.*, vol. 12, no. 3, pp. 676-688, MARCH, 2017.

[32] Y. Zhu, H. Hu, G. J. Ahn and M. Yu, ''Cooperative provable data possession for integrity verification in multicloud storage.'' *IEEE Trans Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231-2244, 2012.

[33] V. Bharati and M. R. Patil. ''Advanced cooperative provable data possession based data integrity verification for multicloud storage,'' *International Journal of Computer Applications*, vol. 81, no. 13, pp. 25-28, 2013.

[34] H. Wang. ''Identity-Based distributed provable data possession in Multicloud storage,'' *IEEE Trans Service Comput.*, vol. 8, no. 2, pp. 328-340, 2015.

[35] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, ''MR-PDP: Multiple-replica provable data possession,'' in *Proc. 28th IEEE Conf. on Distrib. Comput. Syst. (ICDCS)*, 2008, pp. 411-420.

[36] Z. Hao and N. Yu, ''A multiple-replica remote data possession checking protocol with public verifiability,'' in *Proc. 2th Int'l Symp. Data, Privacy, E-Comm. (ISDPE)*, 2010, pp. 84-89.

[37] A.F. Barsoum and M.A. Hasan, ''Integrity verification of multiple data copies over untrusted cloud servers,'' in *Proc. 12th IEEE/ACM Int. Symp. CCGRID*, 2012, pp. 829-834.

[38] J. He, Y. Zhang, G. Huang, Y. Shi and J. Cao, ''Distributed data possession checking for securing multiple replicas in geographically-dispersed clouds,'' *Journal of Comput. and Syst. Sci.*, vol. 78, pp. 1345-1358, 2012.

[39] F. Zhou, S. Peng, J. Xu and Z. Zhou, ''Identity-Based batch provable data possession,'' in *Proc. 10th Int'l Conf. on Provable Security*. Springer International Publishing, 2016, pp. 112-129.

[40] A. F. Barsoum and M. A. Hasan, ''Provable multicopy dynamic data possession in cloud computing systems,'' *IEEE Trans. Inf. Foren. Sec.*, vol. 10, no. 3, pp. 485-497, 2015.

[41] Y. Zhang, J. Ni, X. Tao, Y. Wang and Y. Yu, ''Provable multiple replication data possession with full dynamics for secure cloud storage,'' *Concurrency and Computation: Practice and Experience*, vol. 28, no. 4, pp. 1161-1173, 2016.

[42] M. Yi, L. Wang and J. Wei, ''Distributed data possession provable in cloud,'' *Distributed and Parallel Databases*, vol. 35, no. 1, pp. 1-21, 2017.

[43] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proc. CRYPTO*, vol. 2139. 2001, pp. 213–229.

[44] J. Li, Q. Yu and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience,'' *Information Sciences*, Vol. 484, 113–134, 2019.

[45] J. Li, N. Chen, and Y. Zhang, "Extended file hierarchy access control scheme with attribute based encryption in cloud computing,'' *IEEE Transactions on Emerging Topics in Computing*, DOI: 10.1109/TETC.2019.2904637

[46] D. Boneh, H. Shacham, and B. Lynn, ''Short signatures from the weil pairing,'' *J. Cryptol.*, vol. 17, no. 4, pp. 297-319, Sept. 2004.

[47] The GNU Multiple Precision Arithmetic Library (GMP). [Online]. Available: http://gmplib.org/, accessed Sep. 16, 2016.

[48] The Pairing-based Cryptography Library (PBC). [Online]. Available: https://crpto.stanford.edu/pbc/download.html, accessed Sep. 16, 2016.

Jiguo Li received his B.S. degree in mathematics from Heilongjiang University, Harbin, China in 1996, M.S. degree in mathematics and Ph.D. degree in computer science from Harbin Institute of Technology, Harbin, China in 2000 and 2003, respectively. During 2006.9-2007.3, he was a visiting scholar at Centre for Computer and Information Security Research, School of Computer Science & Software Engineering, University of Wollongong, Australia. During 2013.2-2014.1, he was a visiting scholar in Institute for Cyber Security in the University of Texas at San Antonio. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. He had been a professor with College of Computer and Information, Hohai University, Nanjing, China during 2003.11-2018.6. His research interests include cryptography and information security, cloud computing, wireless security and trusted computing etc. He has published over 150 research papers in refereed international conferences and journals. His work has been cited more than 3000 times at Google Scholar. He has served as program committee member in over 20 international conferences and served as the reviewers in over 90 international journals and conferences.

Hao Yan received the B.S. and M.S. degrees in computer science & technology from the Nanjing University of Science and Technology, China, in 2003 and 2006, respectively. He is currently a Ph.D student of Hohai University, China. His research interests include cloud computing security and applied cryptography.

Yichen Zhang received the Ph.D. degree in the College of Computer and Information, Hohai University, Nanjing, China in 2015. She is currently an associate professor with the College of Mathematics and Informatics, Fujian Normal University, Fuzhou, China. Her research interests include cryptography, network security. She has published over 30 research papers in refereed international conferences and journals.