Enhanced Provable Data Possession in Cloud Computing with Multiple Owners

Cheng-Yu Yang[1]

[1] Department of Communication Engineering National Central University, Taiwan;
995403005@cc.ncu.edu.tw

Cheng-Ta Huang[2]

[2] Department of Information Management, Oriental Institute of Technology, Taiwan;
cthuang@mail.oit.edu.tw

Ya-Ping Wang[3]

[3] Foresight and Innovation Section of Information Management Office, National Police Agency, Ministry of the Interior, Taiwan;
kate@npa.gov.tw

Yen-Wen Chen[4]

[4] Department of Communication Engineering, National Central University, Taiwan;
ywchen@ce.ncu.edu.tw

Shiuh-Jeng WANG [5,*]

[5] Department of Information Management, Central Police University, Taiwan;
***** Correspondence: sjwang@mail.cpu.edu.tw; Tel.: +886-3-3282321 ext. 4807

楊誠育
中央大學通訊工程所

黃正達
亞東技術學院資訊管理系

王雅平
警政署資訊室

陳彥文
中央大學通訊工程所

王旭正
中央警察大學資訊管理研究所

*****Correspondence: sjwang@mail.cpu.edu.tw

摘要

由於行動裝置對雲端資源存取的需求，因此雲端計算和雲端儲存服務日益重要。雲端環境提供的外部儲存服務已經成為一個常見並且隨處可取用的使用者資料存取平台。然而，使用者終究無法如同本機內部硬碟般存取外部雲端儲存資料，以致外部儲存服務無法得到使用者的完全信任。對於前述情形，本研究提出高效率的遠端資料稽核方法，讓使用者以較低計算成本，驗證雲端儲存資料的完整性。本研究基於橢圓曲線雙線性特點，發展雲端儲存的資料儲存證明協定，更易於進行批次的查驗。與其他研究相較，本研究具有高安全性和較高效能。

關鍵詞：雲端安全；資料儲存證明協定；公開稽核服務；密碼學

Abstract

Cloud computing and cloud storage are important developments because they can be accessed by mobile devices. The outsourced storage in cloud environment has become a stable, location-independent platform for managing user data. However, the outsourced data are not trustworthy because they cannot be accessed locally by users. This paper proposes an efficient remote data auditing method, which allows the client to check data possession in cloud storage at a lower computational cost. This study developed an improved provable data possession protocol technique based on the bilinear arithmetic of elliptic curves for cloud storage system, where batch and frequent integrity check is easy to perform completely. Comparisons with other state-of-the-art schemes show that the proposed scheme is highly secure and efficient.

Keywords: cloud security; provable data possession protocol; public audit service; cryptography

1. Introduction

Enterprises and academia have widely implemented cloud architecture. With the rise of personal mobile devices, the underlying infrastructure of information systems comprehensive in cloud computing go into all aspects of human life. Regarding data storage, software and hardware manufacturers have introduced various cloud-based data storage infrastructures. However, if data security and privacy protection are inadequate, cloud services cannot be delivered reliably.

Public cloud services are enabling users to control their data and applications indirectly. Even if the cloud service provider industry infrastructure is more robust and reliable than personal computing devices, users still face internal and external threats to security and privacy, including hardware failure, software error, malware,

operator error and malicious insiders. The cloud service provider (CSP) may discard data that are rarely accessed and it may reuse the same storage space for other customers.

Cloud services security issues are often hidden because the user cannot directly access the internal operational details of the CSP. Additionally, the CSP is also likely to abuse user data. The cloud services are essentially unsafe for users. Therefore, maintaining the security of stored data is a priority.

The main concern of users who store data in the cloud is whether the information is correct and complete. If a cloud storage service could ensure that the user can verify that the storage service provider holds its data correctly, users would consider cloud storage safe. To resolve this important issue, several researchers have proposed remote data auditing protocols (e.g., [1–16]) to verify the integrity of the data stored over a cloud securely and efficiently. Remote data auditing protocols challenge a server from an auditor to issue a provable data possession (PDP) to validate that the data originally stored by a client are on the server. A study of PDP schemes by Ateniese et al. formally defined protocols for PDP and presented two PDP schemes in [1]. Both schemes use homomorphic verifiable tags based on RSA algorithm. The server uses queried blocks and corresponding tags to implement a proof of possession. The client then checks whether the server possesses the file. Shacham and Waters [9] improved the security of the original PDP based on the data fragmentation concept but did not explore the protection of user data. The work by Erway et al. [6] presented the dynamicity in the PDP schemes by rank-based information and authentication dictionary. Ateniese et al. implemented a model for building public-key homomorphic linear authenticators (HLAs) in [2] and [4]. HLAs perform the auditing protocol and thus reduce the communication and computation overhead as compared to the trivial data auditing approaches. In [3], Ateniese et al. provided a PDP scheme by symmetric key cryptography. In [11], Wang Q. et al. proposed a dynamic protocol that supports dynamic operations in data stored on cloud servers. However, a limitation was that the protocol could leak the data content to the auditor because it transfers the combinations of data blocks between the auditor and server. In [12], Wang C. et al. extended their dynamic protocol to batch mode for multiple owners. Yang and Jia [15] used an efficient data auditing scheme to improve security in [11]. In [16, 17], Zhu et al. proposed provable data possession schemes that support the batch auditing for multiple clouds. However, their schemes did not support batch auditing for multiple owners since the data tags used by each owner are more complex. Recently, the authors used the property of the bilinear pairing and improved the dynamic data audit with index table but could not provide security against data leakage attack [18]. In [13], Wang B. et al. provide a privacy-preserving public auditing mechanism for shared data. In [14, 15], the authors propose their auditing scheme with key-exposure resilience to preserve privacy. Sookhak et al. implement a new data structure to decrease the computation overhead in remote data auditing scheme [10].

These properties are as follows.

(i) efficiency: data are verified with a minimum amount of computational cost, (ii) batch auditing: batch auditing is used to verify the integrity of data for the multiowner batch auditing in cloud computing framework.

This paper proposes an efficient remote data auditing method based on the bilinear arithmetic of elliptic curves, which allows the client to check data possession in cloud storage at a lower computational cost compared to homomorphic cryptosystem.

Table 1 lists the components of the scheme designed in this study. Performance comparison with PDP [1], CPDP [9], DPDP [6], relative methods in [7-11] clearly shows that the proposed scheme not only supports complete privacy protection and dynamic data operations, but also enables significant savings in computation costs.

The methods in [11, 12] support dynamic audit but has low efficiency. In the case of the third party auditor that does not actually own the file, privacy is protected. [15] supports dynamic auditing and batch audit function for multi-user, more efficient than methods in Zhu et al. [16, 17], but there may be leak user information such as security holes described in Zhu et al. [18].

The scheme developed by [16, 17] supports dynamic audit but not multi-users, so efficiency is not remarkable. Their scheme has good security and protects the privacy of users. Our approach provides more efficient dynamic auditing for multiple users and more efficient batch auditing compared to the schemes in [11], [16], [17] and has improved security holes in [18].

The features of our approach are summarized as follows.

(i) The proposed provable data possession protocol preserves privacy at a low computational cost to the verifier.

(ii) The proposed security scheme outperforms other state-of-the-art data auditing methods.

The rest of this paper is organized as follows. Section 2 describes the system model and defines the provable data possession protocol. Section 3 describes the properties of the framework. The dynamic and batch proof for the data possession protocol are discussed in section 4. Section 5 and 6 describe the results of security and performance tests of the proposed protocol. Section 7 concludes this paper.

2. Preliminary Remarks and Definitions

Table 1 Comparison of remote data auditing protocols

Scheme	Verifier Computation	Security	Dynamic Manner	Multiple Owner	Multiple Cloud
PDP [1]	O(*t*)	YES	N/A	N/A	N/A
CPDP [9]	O(*t*+*s*)	N/A	N/A	N/A	N/A
DPDP [6]	O(*t*log *n*)	N/A	N/A	N/A	N/A
Methods [11, 12]	O(*t*log *n*)	YES	YES	YES	N/A
Methods [10, 15]	O(*t*)	N/A	YES	YES	YES
Methods [16, 17]	O(*t*+*s*)	YES	YES	N/A	YES
Our Scheme	O(*t*)	YES	YES	YES	YES

n is the total number of data blocks of a file; *t* is the number of challenged data blocks in an auditing query; *s* is the number of sectors in each data block

This section first describes the system model and defines the provable data possession protocol. The three components of the provable data possession architecture for cloud storage are: CSP, its users, and the third-party auditor (TPA). As the users no longer possess the entire data locally, the role of TPA provides public audit service for user efficiently to check the integrity and correctness of their outsourced data.

2.1 Notation and Preliminaries

First, cryptographic primitives used in the scheme are defined follows.
$H(\cdot)$—cryptographic hash functions, defined as: $K \times \{0,1\}^* \to G_1$, where K denotes the key space.

Definition 2.1 (Bilinear Map).
Let G_1, G_2, and G_T be multiplicative cyclic groups of prime order p. Let g_1 and g_2 be generators of G_1 and G_2, respectively. A bilinear map is a map $e: G_1 \times G_2 \to G_T$ such that for all $u \in G_1$, $v \in G_2$ and $a, b \in Z_P$, $e(u^a, v^b) = e(u, v)^{ab}$.

This bilinearity implies that for any $u_1, u_2 \in G_1$, $v \in G_2$, $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$ [5]. In our scheme, this kind of homomorphism property is based on mathematic homomorphism. With bilinearity, homomorphic linear file tags can be combined into one value.

2.2 Definition of the System Model

Fig. 1 shows that the proposed provable data possession protocol for cloud storage involves data owners (user), CSP and TPA. After owners create data, the CSP stores the data and maintains. The TPA performs data storage auditing service for both the owners and CSP. Table 2 lists the notations of the proposed scheme.

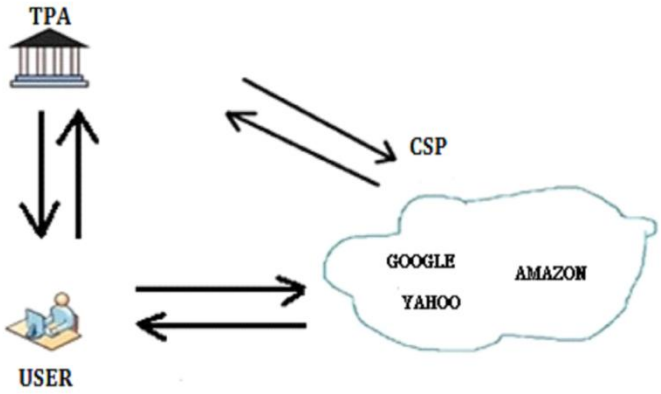

Fig. 1 The architecture of provable data possession protocol for cloud storage

Definition 2.2 (Provable Data Possession Protocol). A provable data possession protocol consists of the following five algorithms:

1. **Key.Gen(1^{λ}).** For this key generation algorithm, the input λ as security parameter for the length of key, and the outputs are a pair of secret-public keys (α, β).
2. **Pub.Setup(*F*, α)** is an algorithm run by the user to generate tags for a file. The inputs for the algorithm are a file *F* and the secret key α. For each data block m_i, it computes the set of tags *T* for file *F*.
3. **Challenge(*n*).** The challenge algorithm takes the number of blocks of the file *F* as input and returns the challenge.
4. **Pub.Proof(*F, T, Q*).** This algorithm takes the blocks of file *F*, the set of tags *T* and the challenge *Q* from the auditor as input and returns a proof of possession *P* for the challenged blocks in *F*.
5. **Pub.Verify(*P*).** The inputs for the verification algorithm are *P* from the CSP. The outputs are the auditing results, *i.e.*, whether the proof is a correct proof of possession.

3. Methods: Provable Data Possession Protocol

The properties of the provable data possession protocol can now be discussed. The performance of the scheme is improved by applying the data fragment technique and homomorphic verifiable tags. The data fragment technique can reduce number of data

tags, given the file *F*, then split *F* into *n* blocks (for some *n*), each *s* sectors long: $\{m_{i,j}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$, results less storage overhead.

Construction for Provable Data Possession Protocol

Table 2 Notations

Symbol	definition
α	**secret key**
β	**public key**
n	**the number of blocks in a file**
s	**the number of sectors in a block**
F	**the processed file with $n \times s$ sectors,** $F = \{m_{i,j}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$
T	**the set of tags**
Q	**the set of index-coefficient pairs**
P	**the response for the challenge Q**

For a file *F*, let G_1, G_2 and G_T be multiplicative cyclic groups of prime order *p*, and $e: G_1 \times G_2 \to G_T$ be a bilinear map. Let g_1 and g_2 be generators of G_1 and G_2, respectively. Let H:$\{0,1\}^* \to G_1$ be a secure map-to-point hash function that maps strings uniformly to G_1. The proposed scheme is composed of five major functions: Key.Gen, Pub.Setup, Challenge, Pub.Proof and Pub.Proof, shown in Fig. 2.

Key.Gen(1^λ)

For a cloud client user, choose a random number. Generate a random key pair (α, β). Choose a random $\alpha \in Z_P$ and compute $\beta \leftarrow g_2^{\propto} \epsilon\, G_2$. The secret key is α, the public key is β.

Pub.Setup(*F*, α)

Given a file *F*, then split *F* into n blocks (for some *n*), each *s* sectors long: $\{m_{i,j}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$. Choose *rv* random values $x_1, x_2, \cdots, x_{rv} \in Z_P$ from some sufficiently large domain Z_P, and compute $u_j = g_1^{x_j} \in G_1$, for all $j \in [1,s], u = (u_1, u_2, \cdots, u_s)$. To compute the file tag $\sigma_i{}'$, for each i, $1 \le i \le n$,

$$\sigma_i' = H(W_i) \cdot (\prod_{j=1}^{s} u_j^{m_{i,j}})^{\alpha} .$$

(1)

The processed file *F* is $\{m_{i,j}\}_{\substack{1\le i\le n \\ 1\le j\le s}}$, together with $\{\sigma_i'\}_{1\le i\le n}$, $W_i = (FID \,\|i\|\, O_i \,\|\, V_i)$, *FID* is file identifier, O_i is the original index of data block, and V_i is the current version of block, α is secret key.

Then, we have the public parameters ($\beta, \{H(W_i), W_i\}_{1\le i\le n}$, u, g_2) and the secret parameter ($\alpha, x_1, x_2, \cdots, x_s$), and the user saves the secret parameter. Finally, public parameters are declared to the public as TPA and the processed file *F* with file tags $T = \{\sigma_i'\}_{1\le i\le n}$ are transferred to the CSP.

Challenge(*n*)

The TPA picks some random data blocks, *i.e.*, subset $[1, i]$ of the set $[1, n]$, to construct the challenge set *Q*. For each i, generate a random element $v_i \in Z_p^*$. Let *Q* be the set $\{(i, v_i)_{i\in[1,n]}\}$ with the i's distinct. Finally, the TPA sends challenge *Q* to the prover.

Pub.Proof(*F, T, Q*)

The CSP parses the processed file *F* as $\{m_{i,j}\}_{\substack{1\le i\le n \\ 1\le j\le s}}$, along with $T = \{\sigma_i'\}_{1\le i\le n}$, which is come from user. The CSP also parses the message sent by the TPA as Q. Compute $\mu' = \{\mu_j'\}_{1\le j\le s'}$

$$\mu_j' = \sum_{(i, v_i)\in Q} v_i \cdot m_{i,j}.$$

(2)

For security concern, CSP chooses *s* as the number of random values $r_j \in Z_p$, where r_j is random value. In addition, we introduce $\rho \in G_1$is a random value, and *R* =$(r_1, r_2, \cdots, r_s)$. *R* and ρ are blind factors designed to prevent from data leakage attack. Then CSP computes$\mu = \{\mu_j\}_{1\le j\le s}$, γ and τ.

$$\gamma = \prod_{j=1}^{s} e(u_j^{-r_j}, \beta).$$

(3)

$$\tau = \prod_{j=1}^{s} e(u_j{}^{\mu_j}, \beta).$$

(4)

Next, σ and ψ are computed as follows:

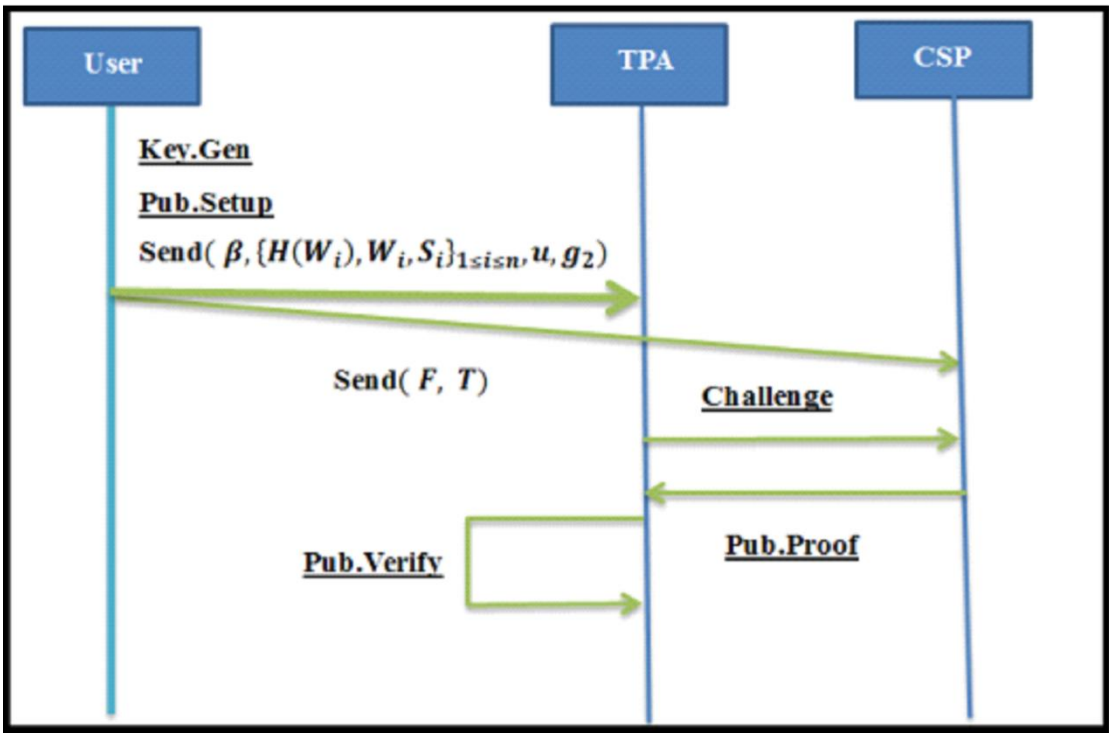

Fig. 2 The provable data possession protocol

$$\mu_j = r_j + \mu'_j.$$

(5)

$$\sigma_i = \rho \cdot \sigma'_i.$$

(6)

$$\sigma = \prod_{(i, v_i) \in Q} \sigma_i{}^{v_i} = \prod_{(i, v_i) \in Q} \{\rho \cdot \sigma'_i\}^{v_i}.$$

(7)

Then utilizes equation (1),

$$\sigma = \left(\prod_{(i, v_i) \in Q} \rho^{v_i} \right) \cdot \prod_{(i, v_i) \in Q} \left\{ H(W_i) \cdot (\prod_{j=1}^{s} u_j{}^{m_{i,j}})^{\alpha} \right\}^{v_i}$$

$$= \prod_{(i,\, v_i)\in Q} \left\{ \rho^{v_i} \cdot H(W_i)^{v_i} \cdot \left(\prod_{j=1}^{s} u_j^{\mu_j'} \right)^{\alpha} \right\}. \tag{8}$$

$$\psi = e\left(\prod_{(i,\, v_i)\in Q} \rho^{v_i}, g_2 \right). \tag{9}$$

Finally, the CSP sends response $P = (\tau, \gamma, \sigma, \psi)$ to the verifier.

Pub.Verify(*P*)

The TPA also parses the CSP's response P to obtain τ, γ, σ and ψ and checks whether the below equation is true.

$$e\left(\prod_{(i,\, v_i)\in Q} \sigma,\ g_2 \right) = \psi \cdot e\left(\prod_{(i,\, v_i)\in Q} H(W_i)^{v_i}, g_2 \right) \cdot \gamma \cdot \tau. \tag{10}$$

If so, output 1; otherwise, output 0. The left side of equation (10) shows the prover response σ in equation (8).

4. Secure Dynamic and Batch Provable Data Possession Protocol

Since the scheme supports the dynamic proof and batch provable data possession protocol, it supports data dynamics auditing in cloud. This section presents how the scheme performs data dynamics, including modification, and insertion besides the static archive data. The batch provable data possession is then discussed.

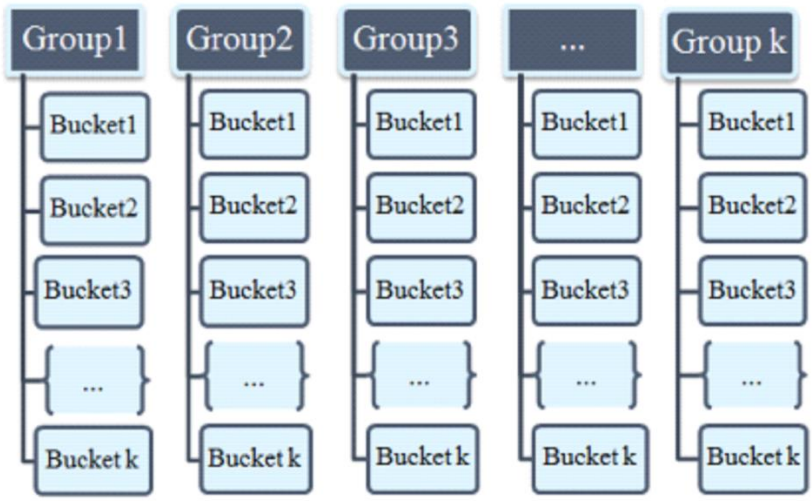

Fig. 3 The data structure of List table

4.1 Constructions for Dynamic Provable Data Possession Protocol

Dynamic data updates are supported by a data structure called List table (LT). The LT prevents the CSP from using the previous version of the stored data rather than the updated one pass successfully, *i.e.*, this study cannot suffer from tag forgery attack in [18]. The two elements of the LT are the original index O_i and version number V_i. The index of each block in LT is the actual position of the outsourced data block. We introduce a new data structure to decrease the computation overhead. First we group all *n* data block into *k* group, yield divide each group into *k* bucket as Fig. 3. This data structure must be managed by the TPA. The procedures for modification, insertion and deletion of data are as follows:

Data Modification

The user executes the modification algorithms as follows:

(i) User modifies the block of the file m_i to m_i^*, then updating $V_i^* = V_i + 1$.

(ii) User generates a new block tag $(\sigma_i')^*$ for modified data block m_i^*, then

$$(\sigma_i')^* = H(W_i^*) \cdot (\prod_{j=1}^{s} u_j^{m_{i,j}^*})^{\alpha} , \tag{11}$$

where $W_i^* = (FID \,\|i\|\, O_i \,\|\, V_i^*)$.

(iii) User sends the modification request message to the CSP, which includes $((\sigma_i')^*, m_i^*)$.

(iv) Upon receiving the modification request message, the CSP replaces the block m_i with m_i^*, and updates the tag $(\sigma_i')^*$.

(v) The user sends the modification request message $W_i^* = (FID \,\|i\|\, O_i \,\|\, V_i^*), H(W_i^*)$ to the TPA, and the TPA update the LT.

Data Insertion

The user must run the algorithm to perform the following modifications:

(i) User inserts a new data block m_i^* and sets the initial value for version of the block V_i^* and the original index of data block $O_i^* = O_i$+1.

(ii) User generates a block tag $(\sigma_i')^*$ for the new data block m_i^*, then

$$(\sigma_i')^* = H(W_i^*) \cdot (\prod_{j=1}^{s} u_j^{m_{i,j}^*})^{\alpha} ,$$

where $W_i^* = (FID \|i\| O_i^* \| V_i^*)$.

(iii) The User sends the insert request message to the CSP, which includes (σ_i^*, m_i^*). Upon receiving the insert request message, the CSP inserts m_i^*, and the tag σ_i^*.

(iv) The User sends the insert request message $W_i^* = (FID \|i\| O_i^* \| V_i^*)$, $H(W_i^*)$ to the TPA, and the TPA constructs a new row in the LT after *i*-th block and shifting the subsequent blocks one position down, *i.e.*, the value of index (i') of the subsequent blocks are plus 1, and $W_{i'} = (FID \|i'\| O_{i'} \| V_{i'})$.

As shown in Table 3, when *k*=5 the CSP insert a block after *i*, the verifier only moves less than $\frac{n}{5^2}$ blocks that spends $\frac{n}{5^2}$ computation overhead. For example while insert new block behind 553 block, which is in Group 3, Bucket 2, those blocks after block 553 in Group 3, Bucket 2 move downward 1 position as shown in Fig. 4, the verifier only moves less than $\frac{1250}{5^2}$ blocks that spends $\frac{1250}{5^2}$ computation overhead. Compare with Sookhak et al. [10] method as shown in Fig. 5, when insert new block behind the 553 block, the verifier moves the subsequent $\frac{1250}{5} - i$ blocks one position down that spends $\frac{1250}{5}$ computation overhead.

Table 3 The data structure of List table when *k*=5

Group 1	Group 2	Group 3	Group 4	Group 5
Bucket 1 Block 1-50	Bucket 1 Block 251-300	Bucket 1 Block 501-550	Bucket 1 Block 751-800	Bucket 1 Block 1001-1050
Bucket 2 Block 51-100	Bucket 2 Block 300-350	Bucket 2 Block 551-600	Bucket 2 Block 801-850	Bucket 2 Block 1051-1100
Bucket 3 Block 101-150	Bucket 3 Block 351-400	Bucket 3 Block 601-650	Bucket 3 Block 851-900	Bucket 3 Block 1101-1150
Bucket 4 Block 151-200	Bucket 4 Block 400-450	Bucket 4 Block 651-700	Bucket 4 Block 901-950	Bucket 4 Block 1151-1200
Bucket 5 Block 200-250	Bucket 5 Block 451-500	Bucket 5 Block 701-750	Bucket 5 Block 951-1000	Bucket 5 Block 1200-1250

Data Deletion

The user needs to run delete algorithm to perform the following steps:

(i) User deletes a data block m_i.

(ii) User sends the delete request message to the CSP, which includes (σ_i, m_i). Upon receiving the delete request message, the CSP delete m_i and the tag σ_i.

(iii) The user sends the delete request message $W_i = (FID \| i \| O_i \| V_i)$ to the TPA, then TPA delete the *i*-th row in the LT and shifting the all of subsequent blocks one position up, *i.e.*, the value of index (i') of the subsequent blocks are minus 1, and $W_{i'} = (FID \| i' \| O_{i'} \| V_{i'})$.

For example when *k*=5, the data structure of List table is same as shown in Table 3, the CSP delete a specific block *i*, the verifier only moves less than $\frac{n}{5^2}$ blocks that spends $\frac{n}{5^2}$ computation overhead. However, in Sookhak et al. method, data deletion is similar to data insertion will incur higher computational overhead $(\frac{n}{5})$ on the verifier. Follow the Yang et al. [15], *n* is infinity in cloud computing. Similarly, $\frac{n}{k^2}$ of group is infinity in cloud computing. Thus, there is no limit with $\frac{n}{k^2}$ blocks in the situation of a group with $\frac{n}{k^2}$+1 blocks while the process of data insertion is executed. Similarly, there is no limit with $\frac{n}{k^2}$-1 blocks while the process of data deletion is executed. For simplify, we indicate $\frac{n}{k^2}$ is the result of ceiling function of $\frac{n}{k^2}$, and $\frac{n}{k}$ is the same in this paper.

4.2 Algorithms for Batch Provable Data Possession Protocol

An end user of multiple clouds may use different account and access more than one CSP at the same time. The differences between a single user in single cloud and multiple users in multiple clouds are shown in the batch compution. Suppose there are w dependent users and k diferent CSPs in the system, denote each client C_w has a file F_{wk}, which means that this file is owned by the client C_w and stored on the CSP P_k.

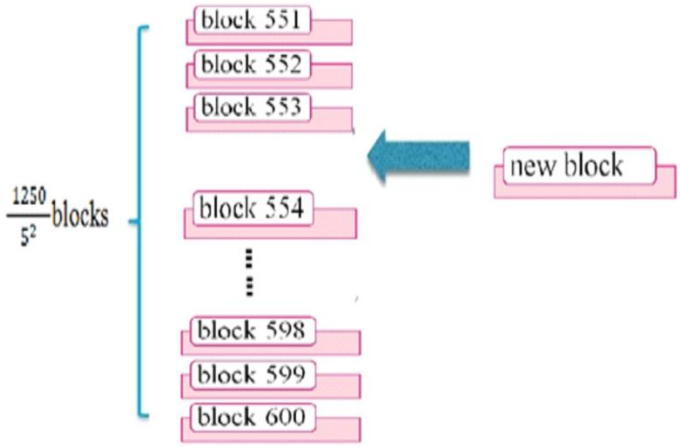

Fig. 4 Insert new block behind the 553 block in our method

Key.Gen(1^{λ})

First, each user C_w chooses a random number. Generate a random key pair (α_w, β_w). Choose a random $\alpha_w \in Z_P$ and compute $\beta_w = g_2^{\alpha_w} \epsilon\, G_2$.

Batch.Setup(F_{wk}, α_w)

For the sake of simplicity, we assume that each file F_{wk} has the same number of n blocks, and each data block is further split into *s* sectors: $\{m_{ij,wk}\}_{\substack{1\le i\le n \\ 1\le j\le s}}$. The user can compute tag $\sigma'_{i,wk}$,

$$\sigma'_{i,wk} = H(W_{i,wk}) \cdot (\prod_{j=1}^{s} u_{j,wk}^{m_{ij,wk}})^{\alpha_w}, \tag{12}$$

where $W_{i,wk} = (FID_{i,wk} \| i \| O_{i,wk} \| V_{i,wk})$. Then, the user has the public parameters $(\beta_w, \{H(W_{i,wk}), W_{i,wk}\}_{1\le i\le n}, u_{wk}, g_2)$ and the secret parameter $(\alpha_w, x_{w1}, x_{w2}, \cdots, x_{ws})$, and the user saves the secret parameter. Similar to the single user, each user C_w sends public parameters to TPA and generates his own file F_{wk} with the corresponding file tags $T_{wk} = \{\sigma'_{i,wk}\}_{1\le i\le n}$ to CSP.

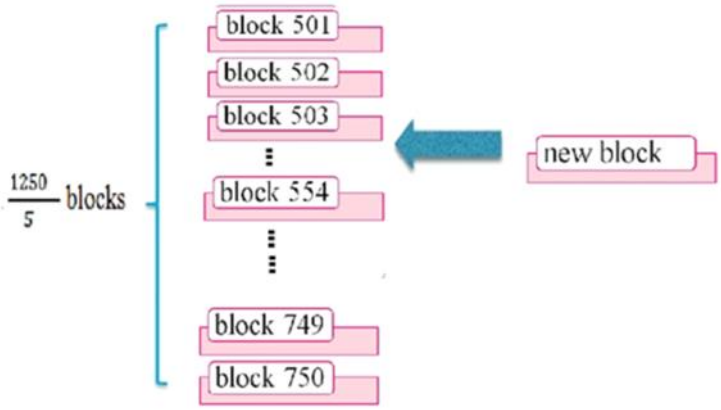

Fig. 5 Insert new block behind the 553 block in Sookhak et al. method

Batch.Challenge(*n*)

In batch challenge alogorithem for w independent users and k diferent CSPs, the TPA pick some random data blocks, subset $[1, i]$ of the set $[1, N]$, $N = w \cdot k \cdot n$, to construct the challenge set Q. For each i, generate a random element $v_i \in Z_p^*$. Let Q be the set $\{(i, v_i)_{i\in[1,N]}\}$ with the i's distinct. Finally, the TPA sends challenge Q to the provers.

Batch.Proof(F_{wk}, T_{wk}, Q)

The CSP P_k parse the processed file F_{wk} as $\{m_{ij,wk}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$, along with $T_{wk} = \{\sigma'_{i,wk}\}_{1\le i\le n}$. Parse the message Q sent by the verifier. Compute $\mu'_{wk} = \{\mu'_{j,wk}\}_{1\le j\le s}$,

$$\mu'_{j,wk} = \sum_{(i,\, v_i)\in Q} v_i \cdot m_{ij,wk}. \tag{13}$$

And the CSP P_k chooses s random $r_{j,wk} \in Z_p$, *i.e.* $R_{wk} = (r_{1,wk}, r_{2,wk}, \cdots, r_{s,wk},)$, and $\rho_{wk} \in G_1$ is a random value. Then the CSP computes $\mu_{wk} = \{\mu_{j,wk}\}_{1\le j\le s}$, γ_{wk} and τ_{wk}.

$$\mu_{j,wk} = r_{j,wk} + \mu'_{j,wk}. \tag{14}$$

$$\gamma_{wk} = \prod_{j=1}^{s} e({u_{j,wk}}^{-r_{j,wk}}, \beta_w). \tag{15}$$

$$\tau_{wk} = \prod_{j=1}^{s} e({u_{j,wk}}^{\mu_{j,wk}}, \beta_w). \tag{16}$$

Next, σ_k, ψ_{wk} are computed in the following:

$$\sigma_{i,wk} = \rho_{wk} \cdot \sigma'_{i,wk}. \tag{17}$$

$$\sigma_k = \prod_{w\in C_w} \sigma_{wk} = \prod_{w\in C_w} \prod_{(i,\, v_i)\in Q} {\sigma_{i,wk}}^{v_i}. \tag{18}$$

then

$$\psi_{wk} = e\left(\prod_{(i,\, v_i)\in Q} \rho_{wk}^{v_i}, g_2\right). \tag{19}$$

Finally, the CSP sends response $P = (\tau_{wk}, \gamma_{wk}, \sigma_k, \psi_{wk})$ to the verifier.

Batch.Verify(P, β).

After the TPA parses the CSPs batch responses to obtain $\tau_{wk}, \gamma_{wk}, \sigma_k$ and ψ_{wk}, confirm that

$$e\left(\prod_{k\in P_k}\sigma_k, g_2\right) = \prod_{w\in C_w}\prod_{k\in P_k}\left\{\psi_{wk}\cdot\gamma_{wk}\cdot\tau_{wk}\cdot e\left(\prod_{(i,v_i)\in Q}H(W_{i,wk})^{v_i}, g_2\right)\right\}. \tag{20}$$

If so, output 1; otherwise, output 0. The outputs are the auditing results, *i.e.*, whether the proof of multiple users in multiple clouds is a correct proof of possession.

5. Security Analysis

We first validate the correctness of our protocol in section 5.1. The security concern in our proposed scheme as follows, we start from where the data could be tampered or lost as in [18], Data Leakage Attack**,** then describe how to prevent from the attack to gain knowledge in section 5.2.

5.1 Correctness Proof of Our Enhanced Provable Data Possession Protocol

The correctness of our protocol is as the following theorem:

Theorem 5.1. In our proposed protocol, the TPA can check if all the chosen data blocks are correctly stored.

Proof. The verification equation (10) in section 3 can be rewritten in details as the following:

$$LEFT = e\left(\prod_{(i,v_i)\in Q}\sigma_i{}^{v_i}, g_2\right) = e\left(\prod_{(i,v_i)\in Q}\left\{\rho^{v_i}\cdot H(W_i)^{v_i}\cdot\left(\prod_{j=1}^{s}u_j{}^{\mu'_j}\right)^{\alpha}\right\}, g_2\right).$$

Additionally, the right side of equation (10) is the prover response τ, γ, ψ and challenge set (i, v_i), which can be expressed with equation (5),

$$RIGHT = e\left(\prod_{(i,v_i)\in Q} \rho^{v_i},\ g_2\right)\cdot e\left(\prod_{(i,v_i)\in Q} H(W_i)^{v_i}, g_2\right)\cdot \prod_{j=1}^{s} e({u_j}^{-r_j}, \beta)$$

$$\cdot \prod_{j=1}^{s} e({u_j}^{\mu_j}, \beta)$$

$$= e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}\cdot H(W_i)^{v_i},\ g_2\right)\cdot \prod_{j=1}^{s} e({u_j}^{\mu'_j}, g_2^{\propto}).$$

then we can get the result as the same as left-hand side,

$$RIGHT = e\left(\prod_{(i,v_i)\in Q} \rho^{v_i}\cdot H(W_i)^{v_i}\cdot \left\{\prod_{j=1}^{s} {u_j}^{\mu'_j}\right\}^{\alpha},\ g_2\right).$$

From equation (4), we can prove that the TPA can pass the correctness, if the data blocks are stored correctly in CSP. But if any of the challenged data block is corrupted, the TPA cannot pass the correctness.

5.2 Security Model: Against for Data leakage attack

We assume the adversary may launch data leakage attack in the auditing scheme. Some existing schemes in [2, 9] are not secure due to the leakage of file information and tags as described in **Theorem 5.2** such as [18]. Our protocol can resist the data leakage attack from the adversary as described in **Theorem 5.3**. These schemes in [2, 9] generally are composed of four phase:

(i) Setup phase:

Given a file *F*, split a given file F into *n* blocks $(m_1, m_2, \cdots, m_n) \in Z_P$ for some large prime p and each block m_i is also split into *s* sectors $(m_{1,s}, m_{2,s}, \cdots, m_{n,s}) \in Z_P$ for some large *p*. Let $e : G \times G \to G_T$ be a computable bilinear map, g is a generator in G and $H : \{0,1\}^* \to G$ be the hash function.

(ii) Generate tags phase:

Assume a client has a private key $x \in Z_P$, and a public key is $(g,\ g^x)$, The client chooses *s* random $u_1, u_2, \cdots, u_s \in G$ as the verification *t*= $(F_n, u_1, u_2, \cdots, u_s)$, where F_n is the file name. The tags on block *i* is

$$\sigma_i = \left(H((F_n \| i)) \cdot \prod_{j=1}^{s} u_j{}^{m_{i,j}} \right)^x . \tag{21}$$

(iii) Audit phase:

On receiving index-coefficient pair query $Q = \{(i, v_i)\}_{i \in I}$, the server responds with

$$\sigma' = \prod_{i \in Q} (\sigma_i)^{v_i}. \tag{22}$$

and $\mu = (\mu_1, \mu_2, \cdots, \mu_s)$,

$$\mu_j = \sum_{(i, v_i) \in Q} v_i \cdot m_{i,j}. \tag{23}$$

(iv) Verify phase:

$$e(\sigma', g) = e\left(\prod_{(i, v_i) \in Q} H(F_n \| i)^{v_i} \cdot \prod_{j=1}^{s} u_j{}^{\mu_j}, g^x \right). \tag{24}$$

Theorem 5.2. The adversary can get the file information and tags by collecting the *n*-times verification contents for a file with $n \times s$ sectors.

Proof. Let *s* be the number of sectors. Based on the results of *n* challenges $(Q^{(1)}, Q^{(2)}, \cdots, Q^{(n)})$, *i.e.*, $(\sigma'^{(1)}, \mu^{(1)})$, $(\sigma'^{(2)}, \mu^{(2)})$, . . . , $(\sigma'^{(n)}, \mu^{(n)})$, $\mu^{(k)} = (\mu_1^{(k)}, \mu_2^{(k)}, \cdots, \mu_s^{(k)})$ and $Q^{(k)} = \{(i, v_i)\}_{i \in I}$, the adversary can solve the system of equations, $\mu_i^{(k)} = m_{1,i} \cdot v_1^{(k)} + m_{2,i} \cdot v_2^{(1)} + \cdots + m_{n,i} \cdot v_n^{(k)}$ for $k \in [1, n]$, to acquire $\{m_{1,i}, m_{2,i}, \cdots, m_{n,i}\}$.

$$\mu_i^{(1)} = m_{1,i} \cdot v_1^{(1)} + m_{2,i} \cdot v_2^{(1)} + \cdots + m_{n,i} \cdot v_n^{(1)} . \tag{25}$$

$$\mu_i^{(2)} = m_{1,i} \cdot v_1^{(2)} + m_{2,i} \cdot v_2^{(2)} + \cdots + m_{n,i} \cdot v_n^{(2)} . \tag{26}$$

$$\vdots$$

$$\mu_i^{(n)} = m_{1,i} \cdot v_1^{(n)} + m_{2,i} \cdot v_2^{(n)} + \cdots + m_{n,i} \cdot v_n^{(n)}. \tag{27}$$

By solving these equations ($i \in [1, s]$) s times the adversary can learn the file contents, $F = \{m_{i,j}\}_{\substack{i\in[1,n] \\ j\in[1,s]}}$. Similarly, the adversary can get all tags $\sigma_1, \sigma_2, \cdots, \sigma_n$ with $\sigma'^{(1)}, \sigma'^{(2)}, \cdots, \sigma'^{(n)}$.

$$\sigma'^{(1)} = {\sigma_1}^{v_1(1)} \cdot {\sigma_2}^{v_2(1)} \cdot \ldots \cdot {\sigma_n}^{v_n(1)} \ . \tag{28}$$

$$\sigma'^{(2)} = {\sigma_1}^{v_1(2)} \cdot {\sigma_2}^{v_2(2)} \cdot \ldots \cdot {\sigma_n}^{v_n(2)} \ . \tag{29}$$

$$\vdots$$

$$\sigma'^{(n)} = {\sigma_1}^{v_1(n)} \cdot {\sigma_2}^{v_2(n)} \cdot \ldots \cdot {\sigma_n}^{v_n(n)}. \tag{30}$$

Theorem 5.3. Our protocol can resist the Data leakage attack from the adversary.

Proof. To solve this security problem, the proposed scheme provides the prover to prove to another that a statement is true, without revealing anything other than the veracity of the statement. These parameters are the same as presented in section 3. In Pub.Proof function, let $\mu'_j = \sum_{(i,\, v_i)\in Q} v_i \cdot m_{i,j}$ and $\sigma' = \prod_{i\in Q} (\sigma_i)^{v_i}$. In order to revealing nothing, the CSP chooses random r_j and ρ, then

$$\mu_j = r_j + \mu'_j = r_j + \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j}.$$

$$\sigma_i = \rho \cdot \sigma'_i.$$

Therefore, the cheating verifier cannot get $\{m_{i,j}\}_{\substack{1\le i\le n \\ 1\le j\le s}}$ from $\mu = \{\mu_j\}_{1\le j\le s}$ and similarly, the cheating verifier cannot get tags σ'_i such as **Theorem 5.2**. Due to the random masking r_j and ρ, the cheating verifier does not learn anything about μ'_j and σ'_i.

In order to protect the privacy of the checked data, we are more concerned about the leakage of personnel information in provable data possession protocol. Given no relative information, the proposed scheme provides better security compared to other data auditing scheme by Yang and Jia [15].

6. Results

This section compares computation cost between the proposed scheme and the PDP proposed by Zhu et al. [16]. The computational cost of the client user, the server, and the TPA is simulated on a Linux system with an Intel Core CPU at 3.10 GHz and 8.00-GB RAM. The code uses the pairing-based cryptography library version 0.5.14 [7] to simulate our auditing scheme and the method by Zhu et al. [16]. The security level is 160 bit (20 bytes), *i.e.*, the sector size is 20 bytes, and the file size is 1MB in our experiments, where the size of each block is 1K Byte and 50 sectors per block. The elliptic curve we used is a MNT d159 curve, where the base field size is 159 bit and the embedding degree is 6 which is the number of the subgroup of the curve [7]. The MNT d159 curve has a 160-bit group order, which means p is a 160-bit long prime. The same curve and base field size are then used in simulations of the proposed auditing scheme and the Zhu et al.'s scheme.

Fig. 6 describes the computation cost of the TPA of the multicloud batch auditing scheme versus the number of clouds. In the proposed scheme, the computational cost of the TPA is clearly lower when compared to the Zhu et al.'s scheme [16]. The figure shows that the time overheads of the Zhu et al.'s scheme are much larger than ours with the increasing numbers of owners. These results indicate an efficient provable data possession protocol, which allows the auditor to check data possession in cloud storage at a lower computational cost compared to the trivial homomorphic cryptosystem.

When numbers of users is 100, our scheme takes 0.33284 s, the method in Zhu et al. [16] takes 0.66502 s, however, when numbers of user is increasing to 700, our scheme takes 2.31103 s, it takes 4.59787 s in the operations in Zhu et al.'s scheme. Because it does not support the batch auditing for multiple owners, we compare, therefore, the computation cost of the TPA between our multi-owner batch auditing and the method in Zhu et al.'s scheme that is not support the multi-owner batch auditing. More importantly, the above experimental results show that the batch scheme is faster than the Zhu et al.'s scheme.

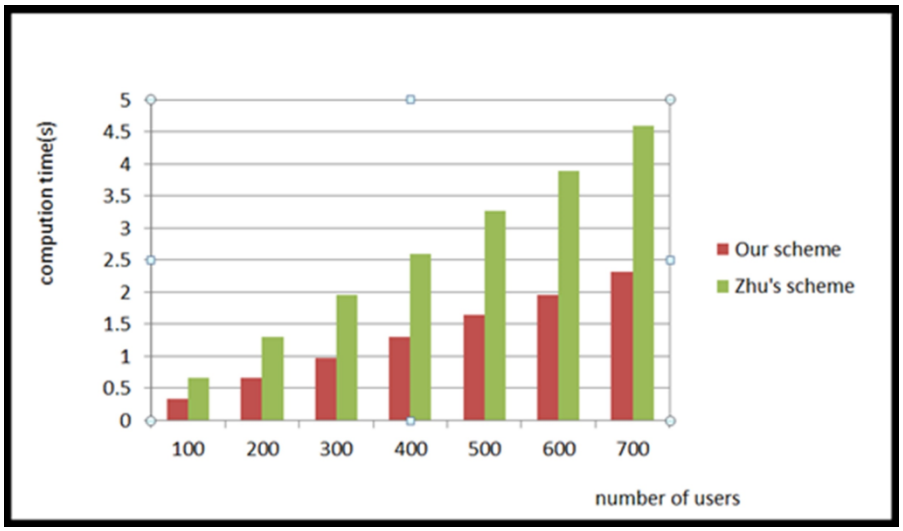

Fig. 6 The computation cost of the TPA of the multicloud batch auditing scheme versus the number of users

As shown in section 4.1, the CSP insert a block after i or delete, the verifier moves less than $\frac{n}{k^2}$ blocks that spends $\frac{n}{k^2}$ computation overhead. The method in Yang et al. [15], to insert a block after i or delete a specific block i, the verifier must shift $(n-i)$ entities in the data structure. Therefore, the computation overhead of their method for insert and delete operations is (n). Similarly, these operations in Sookhak et al. scheme [10], the verifier needs to move $\frac{n}{k}-i$ blocks that takes $\frac{n}{k}$ computation overhead on the verifier. In our method, the verifier only spends $\frac{n}{k^2}$ computation overhead at most. Thus, our method is faster than Sookhak et al. We compare the proposed method with Sookhak method especially in dynamic data update operations as listed in Table 4.

The same file with 125,000 data blocks within 1GB and the size of each block is 8k bytes are then used in simulations of the proposed auditing scheme and the Sookhak et al.'s scheme. The computational cost of dynamic data updates are simulated on a Linux system with mysql Ver 14.14 Distrib 5.5.52. The mathematical model of the proposed scheme are supported by empirical results for different file size from 1GB to 5 GB where the number of update request for insert or delete is 50 and the number of buckets or groups k is equal to 10 as shown in Fig. 7. In our study and Sookhak et al. [10], the computation overhead of finding the position of each block is negligible because the computation of finding the position of each block is shorter than those of data update operation. And each position can be precompute easily in list before data update operation, so computation overhead of finding the position is negligible.

7. Conclusions

This study developed an efficient and secure provable data possession protocol.

The contributions of this paper are summarized as follows.

Table 4 Comparison of remote data auditing protocols in dynamic data update operations.

Computation Cost	**scheme**		
	Yang et al. [15]	Sookhak et al. [10]	Our Scheme
Modification	O(t)	O(t)	O(t)
Insertion	O(n)	O($\frac{n}{k}$)	O($\frac{n}{k^2}$)
Deletion	O(n)	O($\frac{n}{k}$)	O($\frac{n}{k^2}$)

n is the total number of data blocks of a file; t is the number of challenged data blocks in an auditing query; k is the number of groups or buckets

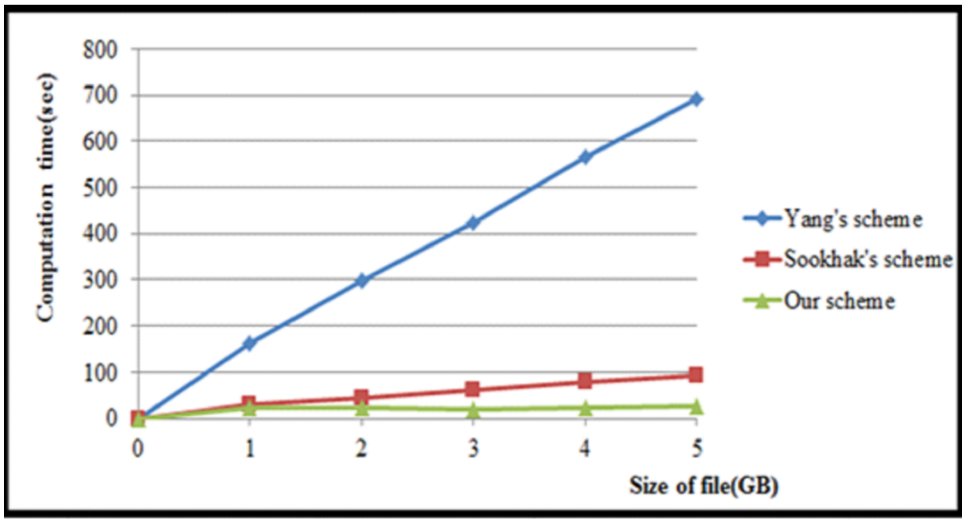

Fig. 7 Comparison of computation cost for different file size from 1GB to 5GB on insert or delete operation

(i) We propose an efficient provable data possession protocol for data storage in cloud computing. The proposed method preserves privacy and minimizes the computation cost incurred by TPA by applying bilinearity property of the bilinear pairing.

(ii) The scheme provides better security compared to other state of the art data auditing methods. It protects the data auditing system against security vulnerabilities by combining the masking method with the bilinearity property of bilinear paring.

(iii)To our best knowledge, we propose a more efficiency method than Sookhak et al. [10] method. We introduce a new data structure to decrease the computation overhead in dynamic data update operations. Additionally, the proof of our data possession protocol supports batch auditing for multiple owners. And, the computational cost of TPA, in this study, is lower enough to which significantly improves the performance

over the past studies.

Acknowledgements This research was partially supported by the Ministry of Science and Technology of the Republic of China under the Grant MOST 106-2221-E-015-001-.

References

[1] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., and Song, D., ”Provable data possession at untrusted stores,” Proceedings of the 14th ACM Conference on Computer and Communication Security, pp. 598–609, 2007.

[2] Ateniese, G., Kamara, S., and Katz. J., “Proofs of storage from homomorphic identification protocols,” Proceedings of the 15th International Conference Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 319-333, 2009.

[3] Ateniese, G., Pietro, R.D., Mancini, L.V., and Tsudik, G., “Scalable and efficient provable data possession,” Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, pp. 1-10, 2008.

[4] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., and Song, D., Remote data checking using provable data possession, ACM Transactions on Information and System Security, vol. 14, pp. 1–34, 2011.

[5] Boneh, D., Lynn, B., and Shacham, H., “Short Signatures from the Weil Pairing,” Journal of Cryptology, vol. 17, pp. 297-319, 2004.

[6] Erway, C., Kupcu, A., Papamanthou, C., and Tamassia. R., “Dynamic Provable Data Possession,” Proceedings of the 15th ACM Conference on Computer and Communication Security, pp. 213-222, 2009.

[7] Lynn, B., “PBC library–the pairing-based cryptography library,” http://crypto.stanford.edu/pbc/

[8] Liu, X., Deng, R. H., Choo, K.-K. R., and Weng, J., An Efficient Privacy-Preserving Outsourced Calculation Toolkit With Multiple Keys, IEEE Transactions on Information Forensics and Security, vol. 11, pp. 2401-2414, 2016.

[9] Shacham, H. and Waters, B., “Compact Proofs of Retrievability,” Proceedings of the 14th International Conference Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 90-107, 2008.

[10] Sookhak, M., Gani, A., Khan, M. K., and Buyya, R., Dynamic remote data auditing for securing big data storage in cloud computing, Information Science, vol. 380, pp. 101-116, 2017.

[11] Wang, Q., Wang, C., Ren, K., Lou, W., and Li, J., Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing, IEEE Transactions on Parallel Distributed Systems, vol. 22, pp. 847-859, 2011.

[12] Wang, C., Wang, Q., Ren, K., and Lou, W., "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," Proceedings of the 29th Conference on Information Communications, pp. 525-533, 2010.

[13] Wang, B., Li, B., and Li, H., "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," IEEE Transactions on Cloud Computing, vol. 2, pp. 43-56, 2014.

[14] Yu, J., Ren, K., Wang, C., and Varadharajan, V., Enabling Cloud Storage Auditing With Key-Exposure Resistance, IEEE Transactions on Information Forensic and Security, vol. 10, pp. 1167-1179, 2015.

[15] Yang, K. and Jia, X., An efficient and secure dynamic auditing protocol for data storage in cloud computing, IEEE Transactions on Parallel and Distributed Systems, vol. 24, pp. 1717–1726, 2012.

[16] Zhu, Y., Hu, H., Ahn, G.-J., and M., Yu., Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage, IEEE Transactions on Parallel and Distributed Systems, vol. 23, pp. 2231-2244, 2012.

[17] Zhu, Y., Wang, H., Hu, Z., Ahn, G.-J., Hu, H., and Yau, S.S., "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds," Proceedings of the 29th Annual ACM Symposium on Applied Computing, pp. 1550-1557, 2011.

[18] Zhu, Y., Wang, S., Hu, H., Ahn, G.-J., and Ma, D., "Secure Collaborative Integrity Verification for hybrid cloud environment," International Journal of Cooperative Information Systems, vol. 21, pp. 165–197, 2012.