International Journal of Security and Its Applications
Vol. 10, No. 12 (2016), pp.81-96
http://dx.doi.org/10.14257/ijsia.2016.10.12.08

An Efficient Public Auditing Scheme for Multi-Cloud Storage

Haiyang Yu[1,*], Yongquan Cai[1], Shanshan Kong[1], Fei Xue[1,2] and Ditta Allah[1]

[1]*College of Computer Science, Beijing University of Technology, Beijing, 100022, China*
[2]*College of Information, Beijing Wuzi University, Beijing, 100149, China,*
[*]*billyukiwi@163.com*

Abstract

In cloud computing, cloud users can upload their data to cloud storage server in order to save local storage and access their data from anywhere. However, cloud storage service also brings serious security issues. Cloud users should be convinced of the correctness of their data stored remotely in the cloud. Thus, a reliable auditing scheme is desired to help cloud users check the integrity of their remote data. In this paper, we first propose an efficient cloud auditing scheme for multi-cloud storage systems, which can also preserve the privacy. Then, we extend our auditing scheme to support dynamic auditing and batch auditing for both multiple cloud users and multiple cloud service providers (CSPs), which makes the scheme more practical and efficient. Security analysis shows that our auditing scheme is provably secure. Our experiments indicate that our solution is efficient and significantly relieves the computation burden of both third party auditor (TPA) and CSP.

Keywords: *Cloud computing security; Cloud storage auditing; Multi-cloud; Batch auditing; Privacy-preserving auditing*

1. Introduction

With the development of IT technology and the extensively application of computers, the amount of data created by computers has exploded in modern society. The quantity of data managed by individual users or enterprises is getting increasingly large. Storing all of the data in local storage system limits system performance. Thus, moving data to cloud storage servers becomes a popular choice [1]. Cloud storage is a crucial service of cloud computing [2], which stores data remotely to the cloud. Compared with local storage, cloud storage offers flexible on-demand outsourcing service for both individuals and IT enterprises, which brings appealing benefits: avoidance of cost on software and hardware, convenient data access on any devices and relief of local storage burden. However, with all these advantages, cloud storage also brings serious security threats [3].

Many security issues of cloud storage technology have exposed to the public in recent years. System failures caused by famous Cloud Service Providers (CSPs) such as Google, Microsoft and Amazon [4] make the public worried about the security of their data. Security of data in the cloud has become the key issue of limiting the use of cloud storage service [5]-[7]. After moving data to CSP, cloud users then lose control of their data which may be modified or deleted by CSP and malicious attackers. Thus, cloud auditing scheme for checking the integrity of cloud data is necessary for cloud users. Besides, in real cloud storage system, one cloud user can use cloud service from multiple CSPs and one CSP can provide storage service for

[*]Corresponding Author

ISSN: 1738-9976 IJSIA

multiple cloud users. Therefore, researches of cloud data auditing in multi-cloud environment become a hot area in both academia and industry.

Related works. Provable Data Possession (PDP) is an essential technology in cloud auditing. It checks the integrity of remote data with high probabilistic possession guarantee. The key component of PDP is homomorphic verifiable tags. The classic model of PDP was proposed by Ateniese *et al.* [8] in 2007. They utilize RSA based homomorphic verifiable tag to check data at untrusted servers. By using sampling strategy, PDP could verify the integrity of remote data with high probability. However, it can't support the verification of user's data with dynamic operations. Subsequently, Ateniese *et al.* [9] came up with a scalable and efficient PDP scheme which can deal with user's data deletion, addition and update except data insertion. Erway *et al.* [10] proposed an extended dynamic PDP scheme by utilizing skip-list. Wang *et al.* [11] utilized Merkle Hash Tree to realize fully dynamic auditing. However, all schemes mentioned above are not fit for auditing multiple CSPs or cloud users.

Curtmola *et al.* [12] first proposed a multi-replica provable data possession which could verify the integrity of multiple copies of user's data stored on multiple servers. Barsoum and Hasan [13], [14] further extend MR-PDP to support dynamic and public auditing. But all these schemes are only suitable for verifying multiple copies of user's data and cannot be applied for cloud user's data which are divided and separately stored at different cloud storages.

Wang *et al.* [15] extended their original scheme [11] to support batch auditing for multiple users by utilizing BLS signature. They also introduce a third party auditor (TPA) to perform public auditing and relieve the computation burden of cloud users. However, their scheme couldn't perform batch auditing for multiple CSPs. Zhu *et al.* [16] proposed a cooperative provable data possession scheme that supports the batch auditing for multiple CSPs by introducing index hash hierarchy. The drawback of their scheme is that it cannot support batch auditing for multiple users. Yang and Jia [17] presented an efficient auditing protocol which support batch auditing for both multiple users and multiple CSPs. However, they couldn't balance the computation complexity between the TPA and the CSP because they move the computation loads from the former to the latter. Liu and Jiang [18] proposed an auditing protocol to support batch auditing for multi-cloud environment, however, like [16], their scheme requires an organizer, which is impractical in reality. Sookhak *et al.* [19] proposed an auditing scheme which has better efficiency than Yang's scheme on both TPA and CSP side, but their scheme couldn't support batch auditing. Yang *et al.* [20] proposed an auditing scheme to prevent the cloud servers from suffering DDOS attack, but it only supports constrained auditing number.

Our contributions. In this paper, we focus on the problem of integrity checking in multi-cloud storage. We propose a public auditing protocol, which can efficiently verify the data integrity without leaking any privacy. It can also support dynamic auditing and batch auditing.

Our contribution can be summarized as follows.

1. We design a new auditing framework for multi-cloud storage systems and propose an efficient and privacy-preserving public auditing protocol. Our auditing scheme takes the advantage of both the random mask technique and bilinear map to ensure the data privacy and improve the performance of TPA and CSP.

2. We expand our auditing scheme to support dynamic data operations.

3. We further expand our scheme to support batch auditing for both multiple users and CSPs, which is efficient in large scale cloud storage environment.

4. We prove the security of our auditing scheme and evaluate the performance of our proposed scheme with comparison experiments.

Organization. The remaining of this paper is organized as follows.

In Section 1, we discuss the related work of the proposed scheme. Section 2 introduces the framework, definitions, threat model and design goals. In Section 3, we describe our detailed auditing scheme. Section 4 gives the security analysis of the proposed auditing scheme. Section 5 presents the performance analysis of our auditing scheme. In the end, we give the conclusion of the whole paper in Section 6.

2. Problem Statement and Definitions

2.1. Definition of System Model

The framework of cloud storage auditing system involves three entities, as shown in Fig 1: the cloud user (U), who uses cloud storage service and stores large amount of data in the cloud; the cloud service provider (CSP), which provides the cloud storage service and has huge storage and computation capability; the third party auditor (TPA), who helps the cloud user to check the integrity of its data in CSP and has more storage space and computation capability than cloud user.

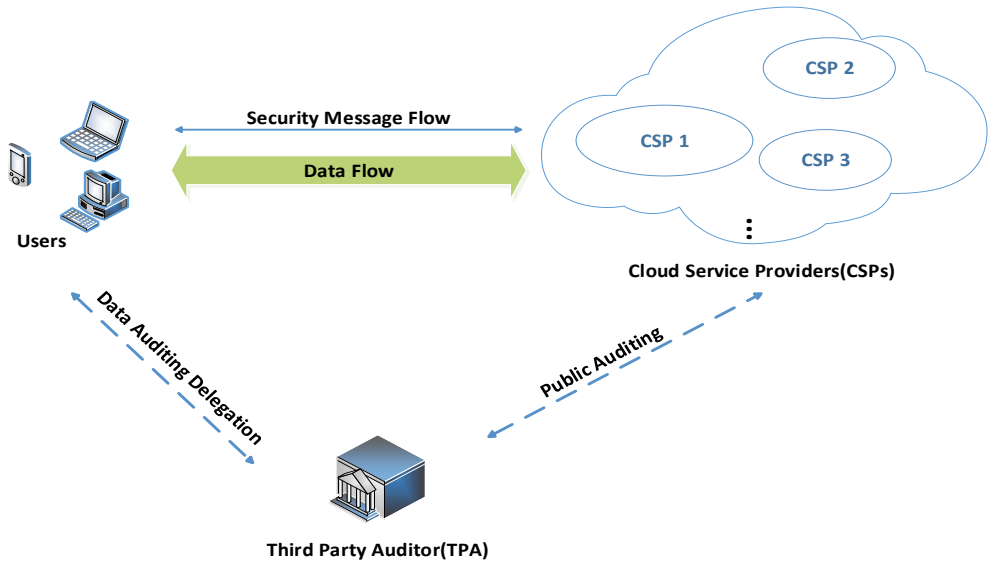


Figure 1. Framework of the Auditing System

Cloud users store their data in CSP and may frequently access or update data in CSP. CSP manages data in its cloud storage servers. To ensure the integrity of cloud users' data, they ask TPA who has more computation resources to help them check the integrity of their outsourced data.

Definition 1 (Cloud Storage Auditing Scheme). Cloud storage auditing scheme consists of five certain algorithms: KeyGen, TagGen, Challenge, ProofGen and Verify.

KeyGen$(\lambda) \rightarrow (pk_u, sk_u, sk_h)$. This key generation algorithm takes a secret parameter λ as input. It outputs a secret hash key sk_h and a pair of user's public-secret key (pk_u, sk_u).

TagGen$(F, sk_u, sk_h) \rightarrow \Phi$. The tag generation algorithm takes file F, secret user key sk_u and secret hash key sk_h as inputs. It splits F into n data blocks and

further splits a data block into s sectors. Then it computes data tags for each data block. It outputs a set of tags $\Phi = \{\sigma_j\}_{j\in[1,n]}$.

Challenge$(F_{info}) \rightarrow C$. The challenge algorithm takes the information of file F_{info} as inputs and outputs challenge C.

ProofGen$(C, F, \Phi) \rightarrow P$. The proof generation algorithm takes the challenge C, the file F and the set of tags Φ as inputs. Then it outputs response proof P.

Verify$(P, pk_u, sk_h, C, F_{info}) \rightarrow 0/1$. The verification algorithm uses the response proof P, public user key pk_u, secret hash key sk_h, the challenge C and the information of data file F_{info} as inputs. It outputs the verification result. If the file component is intact, it outputs 1 and 0 otherwise.

2.2. Threat Model and Design Goals

In the model of our auditing scheme, CSP is considered as a semi-trusted entity. It manages user's data but may conceal data corruption caused by CSP's fault or malicious attack. CSP may even delete some data which is hardly accessed by user to save storage space. We consider that TPA is curious but honest. It will honestly perform the auditing task according to the auditing scheme. However, it may learn user's data from received response proofs.

To efficiently check the data integrity, our auditing scheme should achieve the following goals:

Public Auditability: The scheme should allow TPA to verify the integrity of user's data without adding additional online burden on the cloud user.

Auditing Correctness: The scheme should ensure that CSP cannot pass the auditing when it cheats the TPA.

Privacy Preserving: TPA and CSP cannot learn any user's data content from the auditing scheme during the process.

Batch Auditing: The scheme should allow TPA to perform the auditing tasks delegated from multiple users and multiple CSPs.

Efficiency: The scheme should incur less computation cost than previous schemes on both TPA and CSP side.

3. The Proposed Auditing Scheme

In this section, we first introduce related notations and preliminaries which are applied in our auditing scheme. Then we propose detailed algorithms of our auditing scheme.

3.1. Notations and Preliminaries

1) $h(\cdot)$ is a cryptography hash function.

2) Bilinear Map: let G_1, G_2 and G_T be multiplicative cyclic groups of large prime order p. Let g_1 and g_2 be the generator of $G_1$and G_2 respectively. A bilinear map is a map e: $G_1 \times G_2 \rightarrow G_T$ with following properties:

- for all $u \in G_1, v \in G_2$ and a, b$\in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$, this property also means that for $u_1, u_2 \in G_1$, $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$.
- Nondegeneracy: $e(g_1, g_2) \neq 1$.
- Computability: there is an efficient algorithm to compute the e.

3.2. Algorithms for Auditing Scheme

In our scheme, we assume a file F is split into n data blocks and each data block is further divided into s sectors. By introducing data fragment technique in [16] and [17] that each data block corresponds to one tag, we could decrease the amount of tags to save the storage space in the cloud and reduce the communication cost. For simplicity, we assume all data blocks are split into same number of sectors.

To solve the problem of privacy leakage during auditing, we use random mask technique to preserve the privacy of user's data. Thus, TPA couldn't learn user's data from the response proof returned by CSP.

Let G_1, G_2 and G_T be multiplicative cyclic groups of large prime order p. Let e be the bilinear map which is introduced in notations. Let g_1 and g_2 be the generator of G_1 and G_2, respectively. Let $h(\cdot):\{0,1\}^* \to G_1$ be a cryptographic hash function which maps strings uniformly to a point in G_1.

The detailed algorithms of our auditing scheme are as follows:

KeyGen$(\lambda) \to (pk_u, sk_u, sk_h)$. The key generation algorithm only takes secure parameter λ as input. Cloud user chooses two random elements $sk_u, sk_h \in \mathbb{Z}_p$ and computes $pk_u = {g_2}^{sk_u} \in G_2$. Then it outputs public user key pk_u, secret user key sk_u and secret hash key sk_h.

TagGen$(F, sk_u, sk_h) \to \Phi$. The tag generation algorithm takes data file F, secret user key sk_u and secret hash key sk_h as inputs. It selects s random number $\eta_1, \eta_1, \cdots \eta_s \in \mathbb{Z}_p$ and computes $u_k = {g_1}^{\eta_k} \in G_1$, for $k \in [1, s]$. It computes a tag for each data block m_j as

$$\sigma_j = (h(sk_h, \mathrm{T}_j) \cdot \prod_{k=1}^{s} {u_k}^{m_{jk}})^{sk_u} \quad (1)$$

where $\mathrm{T}_j = FID \parallel j$. FID is identifier for each file and j is the index of m_j. The algorithm finally outputs the set of data tags $\Phi = \{\sigma_j\}_{j\in[1,n]}$.

Challenge$(F_{info}) \to C$. The challenge algorithm takes the information of data file F as input. It randomly chooses α data blocks to construct a challenge set Θ. For each selected data block m_j, it chooses a random value v_j. It then outputs the challenge $C = \{j, v_j\}_{j\in\Theta}$.

ProofGen$(C, F, \Phi) \to P$. The proof generation algorithm takes challenge C, file F and data tags Φ as inputs. It first computes an aggregated tag $\sigma = \prod_{j\in\Theta} {\sigma_j}^{v_j} \in G_1$. It then selects a random element $\gamma \in \mathbb{Z}_p^*$ and s random values $\theta_k \in \mathbb{Z}_p$ for $k \in [1, s]$. It computes masked data proof $\mu = \{\mu_k\}_{k\in[1,s]}$, where $\mu'_k = \sum_{j\in\Theta} v_j \cdot m_{jk}$ for $k \in [1, s]$ and $\mu_k = \gamma \cdot \mu'_k + \theta_k$. Then it calculates $R = \mathrm{e}(\prod_{k=1}^{s} u_k^{\theta_k}, pk_u)$. Finally, the algorithm outputs response proof $P = (\sigma, \mu, R, \gamma)$.

Verify$(P, pk_u, sk_h, C, F_{info}) \to 0/1$. The verification algorithm takes response proof P, public user key pk_u, secret hash key sk_h, the challenge C and the information of the file F_{info} as inputs.

It then verifies the response proof from the CSP by the following verification equation:

$$R \cdot \mathrm{e}(\sigma^{\gamma}, g_2) = \mathrm{e}(\prod_{j\in\Theta} h(sk_h, \mathrm{T}_j)^{\gamma v_j} \cdot \prod_{k=1}^{s} u_k^{\mu_k}, pk_u) \quad (2)$$

if the verification equation holds, then the algorithm outputs 1. Otherwise, it outputs 0.

3.3. Construction of Our Auditing Scheme

We further describe the whole construction of our auditing scheme. It consists of two phases: Setup and Auditing.

Setup. The cloud user first generates public and private keys by running the key generation algorithm KeyGen. It divides a file into n data blocks and further splits each data block into s sectors. Then the cloud user runs the tag generation algorithm TagGen to generate verification tag for each data block. After tag generation, it sends data blocks $\{m_j\}_{j\in[1,n]}$, verification tags Φ and parameters $\{u_k\}_{k\in[1,s]}$ to the CSP. The cloud user sends secret hash key sk_h, public user key pk_u and parameters $\{u_k\}_{k\in[1,s]}$ to the TPA together with information of each file F_{info}.

Auditing. When cloud user wants to check the data integrity in one CSP, it sends checking request to the TPA. TPA runs the challenge algorithm Challenge to generate the challenge C and sends it to the CSP. CSP runs proof generation algorithm ProofGen to generate response proof P. Then it sends the response proof back to the TPA. TPA finally runs the verification algorithm Verify to generate the auditing result. TPA then sends the result to the cloud user.

3.4. Support for Dynamic Auditing

After storing data in CSPs, cloud user may need to dynamically update their data in the cloud for various purposes. Thus, supporting dynamic operations in our auditing scheme is extremely important.

We introduce $Data\ Block\ Table$ ($DBTable$) to record the information of updated data blocks. $DBTable$ is a table contains information of data blocks in a file. The $DBTable$ consists of four components: No, O_j, FID and R_j. No is the current index of data block m_{ij} in data file F. O_j denotes the original index of the data block. FID is the file identifier that data block belongs to. R_j denotes a random number used for generating the data tag.

$DBTable$ is created by cloud user in the setup phase. It is then managed by the TPA.

3.4.1. Details for Dynamic Auditing: The cloud user could do three kinds of dynamic operations on its data: Modification, Insertion and Deletion.

Modification. The cloud user first modifies the data block m_j'. It then generates a new random number R_j' and a new tag σ_j'. It sends data block m_j' and the new tag σ_j' to the CSP to replace the old one. It sends a modification message with No and R_j' to the TPA. Upon receiving the message, TPA will update the record of index No with the new random value R_j'.

Insertion. If the cloud user wants to insert a data block into the j position, it first generates a new data block m_j' and a new tag σ_j'. Then it chooses a new random number R_j' for the data block. It sends the new data block m_j' and new tag σ_j' to the CSP. CSP inserts them into the server. The cloud user sends an insertion message with No, O_j, FID and R_j' to the TPA. TPA will insert the message into the j position and increase No of all records after m_j' by 1.

Deletion. The cloud user first deletes the data block m_j and the corresponding tag σ_j in the CSP. It then sends a deletion message with No of the data block to the TPA. TPA will delete the record of No and subtract all No of the data blocks after m_j by 1.

3.5. Support for Batch Auditing of Our Scheme

Due to a large amount of cloud users and CSPs in the cloud, the TPA may need to concurrently deal with many auditing tasks delegated from multiple cloud users and multiple CSPs. The performance of auditing scheme will be improved if these auditing tasks could be combined together and performed simultaneously by TPA.

3.5.1. Algorithms for Our Batch Auditing Scheme: Let U be the set of cloud users and P be the set of CSPs. In our model, each cloud user $U_x(x \in U)$ sends one of its data file F_{xy} to one CSP $P_y(y \in P)$. The algorithms of batch auditing scheme are presented as follows.

Setup. Cloud user U_x runs the key generation algorithm KeyGen to generate a pair of public-secret user key $(sk_{u,x}, pk_{u,x})$ and a secret hash key $sk_{h,xy}$. For each cloud user U_x, it first splits each data file of the data F_{xy} into $n \times s$ sectors. We denote a file as $F_{xy} = \{m_{xy,jk}\}_{j\in[1,n],k\in[1,s]}$. For simplicity, we assume each file has same number of data blocks and each data block has same number of sectors. Then, each cloud user U_x runs the tag generation algorithm TagGen to generate tag for each data block $m_{xy,j}$ as

$$\sigma_{xy,j} = (h(sk_{h,xy}, \mathrm{T}_{\mathrm{xy},j}) \cdot \prod_{k=1}^{s} u_{x,k}{}^{m_{xy,jk}})^{sk_{u,x}} \quad (3)$$

where $\mathrm{T}_{\mathrm{xy},j} = FID_{xy}||j$. Each cloud user U_x sends each file F_{xy}, set of verification tags Φ_{xy} to the corresponding CSP P_y together with the parameters $\{u_{x,k}\}_{k\in[1,s]}$. Then it sends the public user key $pk_{u,x}$, the secret hash key $sk_{h,xy}$ and the information of each file $F_{info,xy}$ to the TPA with the parameters $\{u_{x,k}\}_{k\in[1,s]}$.

Batch Auditing. Let U_{chal} be the subset of cloud users U which ask for auditing and P_{chal} denotes the subset of CSPs P which answer the challenge.

First, we introduce three new algorithms for batch auditing: BatchChallenge, BatchProofGen and BatchVerify to replace the original algorithms mentioned in Section 3.2: Challenge, ProofGen and Verify respectively.

BatchChallenge$(\{F_{info,xy}\}_{x\in U_{chal},y\in P_{chal}}) \rightarrow \{C_{xy}\}_{x\in U_{chal},y\in P_{chal}}$. Batch challenge algorithm takes the information of all files $\{F_{info,xy}\}_{x\in U_{chal},y\in P_{chal}}$ as input. For each involved cloud user U_x and CSP P_y, it randomly chooses α data blocks to construct a challenge subset Θ_{xy}. For each selected data block $m_{xy,j}$, it chooses a random value $v_{xy,j}$. It outputs the challenge sets $\{C_{xy}\}_{x\in U_{chal},y\in P_{chal}}$, where $C_{xy} = \{j, v_{xy,j}\}_{j\in\Theta_{xy}}$.

BatchProofGen$(\{C_{xy}\}_{x\in U_{chal}}, \{F_{xy}\}_{x\in U_{chal}}, \{\Phi_{xy}\}_{x\in U_{chal}}) \rightarrow P_y$. For each CSP P_y, the batch proof generation algorithm takes challenge sets $\{C_{xy}\}_{x\in U_{chal}}$, all files $\{F_{xy}\}_{x\in U_{chal}}$ and all data tags $\{\Phi_{xy}\}_{x\in U_{chal}}$ as inputs. It first computes an aggregated tag $\sigma_y = \prod_{x\in U_{chal}} \prod_{j\in\Theta_{xy}} \sigma_{xy,j}{}^{v_{xy,j}} \in G_1$. It randomly chooses a random element $\gamma_y \in \mathbb{Z}_p^*$ and selects s random values $\theta_{xy,k} \in \mathbb{Z}_p$ for $k \in [1,s]$. It computes masked data proof $\mu_y = \{\mu_{xy,k}\}_{x\in U_{chal},k\in[1,s]}$, where $\mu_{xy,k} = \gamma_y \cdot \mu'_{xy,k} + \theta_{xy,k}$ and $\mu'_{xy,k} = \sum_{j\in\Theta_{xy}} v_{xy,j} \cdot m_{xy,jk}$ for $k \in [1,s]$. It calculates $R_{xy} = \mathrm{e}(\prod_{k=1}^{s} u_{x,k}^{\theta_{xy,k}}, pk_{u,x})$ and further computes $R_y = \prod_{x\in U_{chal}} R_{xy}$. The algorithm outputs a set of response proofs $P_y = (\sigma_y, \mu_y, R_y, \gamma_y)$.

BatchVerify$(\{P_y\}_{y\in P_{chal}}, \{pk_{u,x}\}_{x\in U_{chal}}, \{C_{xy}\}_{x\in U_{chal},y\in P_{chal}}, \{sk_{h,x}\}_{x\in U_{chal}}, \{F_{info,xy}\}_{x\in U_{chal},y\in P_{chal}}) \rightarrow 0/1$. The batch verification algorithm takes a set of response proofs $\{P_y\}_{y\in P_{chal}}$, public user key $\{pk_{u,x}\}_{x\in U_{chal}}$, secret hash key

$\{sk_{h,x}\}_{x\in U_{chal}}$, the challenge set $\{C_{xy}\}_{x\in U_{chal},y\in P_{chal}}$ and the information of all files $\{F_{info,xy}\}_{x\in U_{chal},y\in P_{chal}}$ as inputs.

The algorithm first computes $R=\prod_{y\in P_{chal}} R_y$, $\mu_{x,k}=\sum_{y\in P_{chal}}\mu_{xy,k}$. It then verifies the response proofs from all CSPs by the following verification equation:

$$R\cdot \mathrm{e}\left(\prod_{y\in P_{chal}} \sigma_y{}^{\gamma_y}, g_2\right) = \prod_{x\in U_{chal}} \mathrm{e}\left(\left(\prod_{y\in P_{chal}}\prod_{j\in\Theta_{xy}} h(sk_{h,xy}, \mathrm{T}_{xy,j})^{\gamma_y v_{xy,j}}\right)\cdot \prod_{k=1}^{s} u_{x,k}^{\mu_{x,k}}, pk_{u,x}\right) \quad (4)$$

if the verification equation holds, then the algorithm outputs 1. Otherwise, it outputs 0.

Second, we describe the construction of batch auditing phase.

For each cloud user $U_x(x\in U_{chal})$, it sends auditing request to TPA. TPA runs the batch challenge algorithm BatchChallenge to generate the challenge $\{C_{xy}\}_{x\in U_{chal},y\in P_{chal}}$ for each cloud user and each CSP and sends it to the CSP. Each CSP runs batch proof generation algorithm BatchProofGen to generate response proof P_y. Then it sends the response proof back to the TPA. TPA finally runs the batch verification algorithm BatchVerify to generate the verification result. TPA then sends the verification result to all cloud users.

4. Security Analysis

We evaluate the security of our auditing scheme by analyzing the achievement of the security goals in Section 2.2, which are public auditability, auditing correctness and privacy preserving property.

4.1. Public Auditability

Theorem 1. Anyone, not only the cloud user, could challenge the CSP to verify the integrity of cloud data without possessing any secret parameters.

Proof. In the Auditing phase, for every available data file F on the CSP, anyone could construct a random challenge C. It can also obtain secret hash key sk_h, public user key pk_u, parameters $\{u_k\}_{k\in[1,s]}$ and information of each file F_{info} from TPA and the response proof P from CSP. Then anyone can run the verification algorithm Verify to check the integrity of cloud data file with all inputs mentioned above. Hence the auditing scheme has public auditability.

4.2. Auditing Correctness

Theorem 2. A CSP can pass the verification phase only if it generates proofs with the specific intact data.

Proof. We first prove that the equation (2) is correct.

$$\begin{aligned} R\cdot \mathrm{e}(\sigma^{\gamma}, g_2) &= \mathrm{e}\left(\prod_{k=1}^{s} u_k^{\theta_k}, pk_u\right)\cdot \mathrm{e}\left(\prod_{j\in\Theta} \sigma_j{}^{v_j\gamma}, g_2\right) \\ &= \mathrm{e}\left(\prod_{k=1}^{s} u_k^{\theta_k}, pk_u\right)\cdot \mathrm{e}\left(\prod_{j\in\Theta} (h(sk_h, \mathrm{T}_j)\cdot \prod_{k=1}^{s} u_k{}^{m_{jk}})^{sk_u v_j\gamma}, g_2\right) \\ &= \mathrm{e}\left(\prod_{k=1}^{s} u_k^{\theta_k}, pk_u\right)\cdot \mathrm{e}\left(\prod_{j\in\Theta} h(sk_h, \mathrm{T}_j)^{\gamma v_j}\cdot \prod_{k=1}^{s}\prod_{j\in\Theta} u_k^{\gamma v_j\cdot m_{jk}}, pk_u\right) \end{aligned}$$

$$= \mathrm{e}(\prod_{k=1}^{s} u_k^{\theta_k}, pk_u) \cdot \mathrm{e}(\prod_{j\in\Theta} h(sk_h, \mathrm{T}_j)^{\gamma v_j} \cdot \prod_{k=1}^{s} u_k^{\gamma\cdot\sum_{j\in\Theta} v_j\cdot m_{jk}}, pk_u)$$

$$= \mathrm{e}(\prod_{j\in\Theta} h(sk_h, \mathrm{T}_j)^{\gamma v_j} \cdot \prod_{k=1}^{s} u_k^{\gamma\cdot\mu'_k+\theta_k}, pk_u)$$

$$= \mathrm{e}(\prod_{j\in\Theta} h(sk_h, \mathrm{T}_j)^{\gamma v_j} \cdot \prod_{k=1}^{s} u_k^{\mu_k}, pk_u)$$

Finally, according to [21], the correctness of the proposed scheme can be reduced to computational Diffie-Hellman problem, which is an unsolved problem in polynomial time.

4.3. Privacy Preservation

Theorem 3. TPA can't retrieve any information of user's data according to the response proof from CSPs.

Proof. The proposed auditing scheme introduces random values γ and $\{\theta_k\}_{k\in[1,s]}$ to prevent the data leakage risk generated from linear combination of data blocks [15]. Because of the property of bilinear map, TPA can't reveal the random parameter θ_k from R. Therefore, the response proof cannot leak any information of data blocks to TPA.

5. Performance Analysis

In this section, we first analyze the verification probability of our auditing scheme. Then we evaluate the performance of the proposed auditing scheme by comparing communication and computation cost between our auditing scheme and the existing work: the efficient audit protocol proposed by Yang et al. [17].

5.1. Verification Probability

Our auditing scheme uses sampling strategy to check the data corruption in CSP. Therefore, the verification probability is essential for ensuring the accuracy of our auditing scheme. Suppose the corruption probability of each sector is δ in one CSP. For simplicity, we assume that sectors in each CSP have same corruption probability. Then the verification probability of auditing scheme can be calculated as

$$Pr = 1 - (1-\delta)^{s\cdot t} \tag{5}$$

where s is the number of sectors in each data block and t is the number of data blocks challenged by TPA. Consider $s = 50$ and 0.1% sectors are corrupted. When $t = 60$, the verification probability Pr is more than 95%. When $t = 94$, the verification probability is over 99%. Furthermore, in multiple users and multiple CSPs case, t represents the sum of the number of challenged data blocks from each cloud user in each CSP, which means batch auditing can highly reduce the number of challenged data blocks for each cloud user in each CSP. For instance, if TPA performs batch auditing for 10 cloud users, it just needs to check no more than 10 data blocks on average for each cloud user in order to achieve the 99% verification probability.

5.2. Communication Cost Analysis

We only compare the communication cost on TPA and CSP in Challenge and ProofGen algorithms because our scheme and existing scheme have similar communication cost on cloud user side in Setup phase.

Table 1. Communication Cost Comparison (k users, l CSPs)

Scheme	Challenge	ProofGen
Existing Scheme [17]	$kl(t+1)$	$2l$
Our Scheme	klt	$3l+kls$

Consider there are k cloud users and l CSPs in the cloud. t denotes the number of challenged data blocks for each cloud user in each CSP. s denotes the number of sectors in each data block. For simplicity, we assume all users have the same number of challenged data blocks and same number of sectors in one data block.

From the table, it is easy to find that although our scheme has more communication cost in ProofGen algorithm, it has less communication cost than existing scheme in Challenge algorithm. This is because in ProofGen algorithm our auditing scheme should transmit masked data proof of each sector for each cloud user and in Challenge algorithm the existing scheme should transfer challenge stamp for each user to each CSP.

5.3. Computation Cost Analysis

We conduct the experiments on a Ubuntu 14.04.4 Linux system with an Intel Core i5 CPU at 3.30GHz and 4GB DDR3 RAM. We use pairing-based cryptography (PBC) library version 0.5.14 to simulate our auditing scheme and the existing scheme. The elliptic curve used in our implementation is a MNT d159-curve. In all experiments, the number of sectors of each block s is fixed to 50.

5.3.1. Computation Cost of TPA: Because the computation cost on cloud user side is almost the same between our scheme and Yang's scheme, we just compare the computation cost on TPA and CSP. We first analyze the computation complexity on TPA side.

To demonstrate the efficiency of our auditing scheme in single user and single CSP case, we present computation time comparison of individual auditing of our scheme and existing scheme in Fig 2. It shows that computation cost of both our scheme and existing scheme grows linearly. Besides, our scheme costs slightly less computation time than the existing scheme on the TPA side as the number of challenged blocks increases.

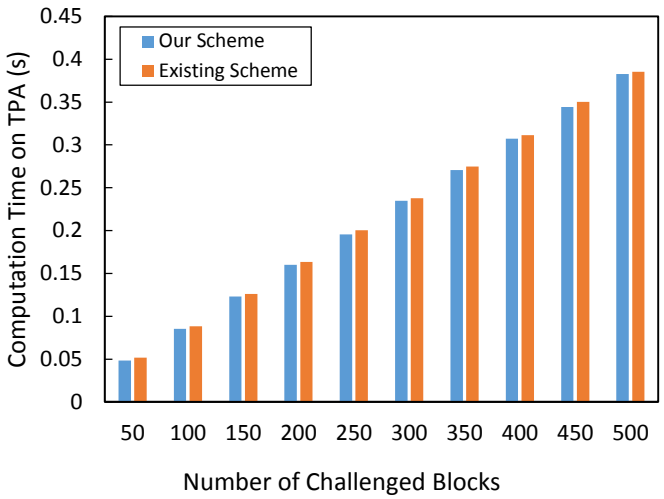


Figure 2. Performance Comparison of Individual Auditing on TPA

Although our auditing scheme is efficient in single user and single CSP case, it is not practical enough because in real cloud storage environment, more users from different CSPs will send requests for cloud auditing service. Thus, in order to illustrate the performance of our batch auditing scheme in both multiple users and multiple CSPs case, we compare the computation time of TPA versus the number of challenged users and the number of challenged CSPs, which are shown in Fig 3 and Figure 4, respectively. Figure 3 shows that compared with the existing scheme, our batch auditing scheme for multiple users can greatly reduce more than half of the computation cost. Even the number of challenged users goes to 500, the computation time of TPA doesn't exceed 1.5 seconds, which is practical and efficient.

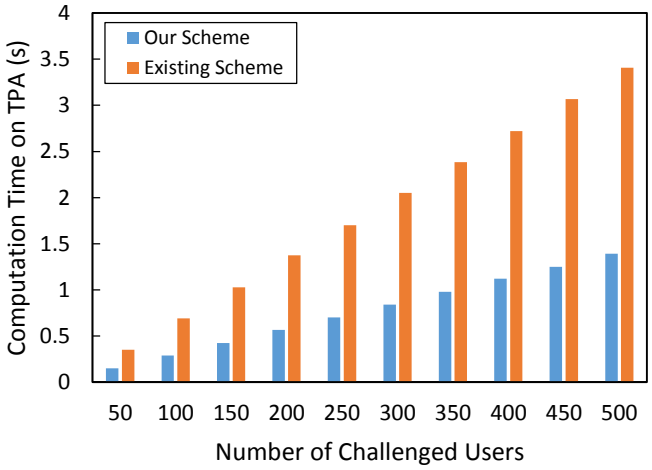


Figure 3. Performance Comparison of Batch Auditing on TPA (Single CSP, 5 blocks/User)

Figure 4 demonstrates that our batch auditing scheme for multiple CSPs incurs more computation cost than the existing scheme. However, even the number of challenged CSPs goes to 500, the difference in computation time between two schemes is less than 0.5 seconds, which is acceptable in real cloud storage systems, especially when considering the optimization of our scheme on CSP side and on TPA during other auditing cases.

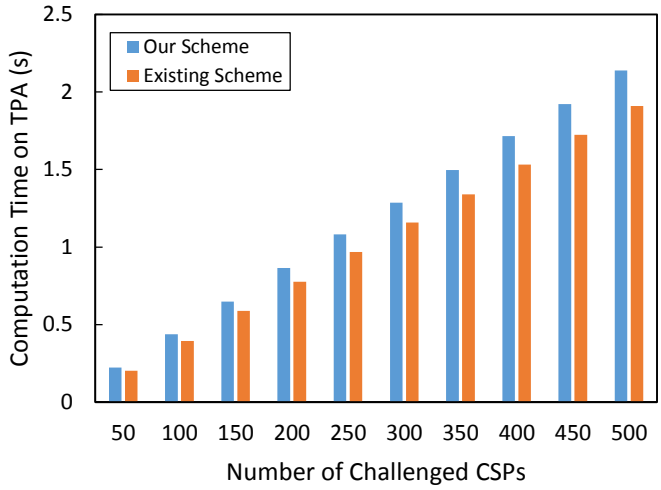


Figure 4. Performance Comparison of Batch Auditing on TPA (Single User, 5 blocks/CSP)

5.3.2. Computation Cost of CSP: To evaluate the performance of our auditing scheme on CSP, we compare the computation time of CSP versus the number of challenged data blocks, the number of challenged users and the number of challenged CSPs in Figure 5, Figure 6 and Figure 7 respectively.

Simulation result on CSP side in single user and single CSP case is shown in Figure 5. In this figure, compared with existing scheme, our auditing scheme can save nearly 70% computation time on average on each CSP.

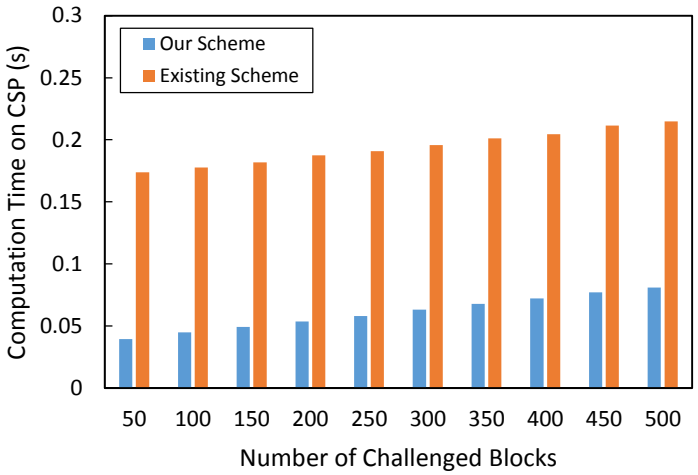


Figure 5. Performance Comparison of Individual Auditing on CSP

Figure 6 shows the simulation results of two schemes in multiple users case. It describes different linear growth rate of two schemes as the number of challenged users increases. From Figure 6, we can easily see that our auditing scheme greatly reduces the computation cost when compared with existing scheme. This is because in our auditing scheme, when generating proofs, the CSP has less pairing operations than the existing scheme, which are the most time-consuming operations compared with other operations such as exponentiation operations and multiplication operations.

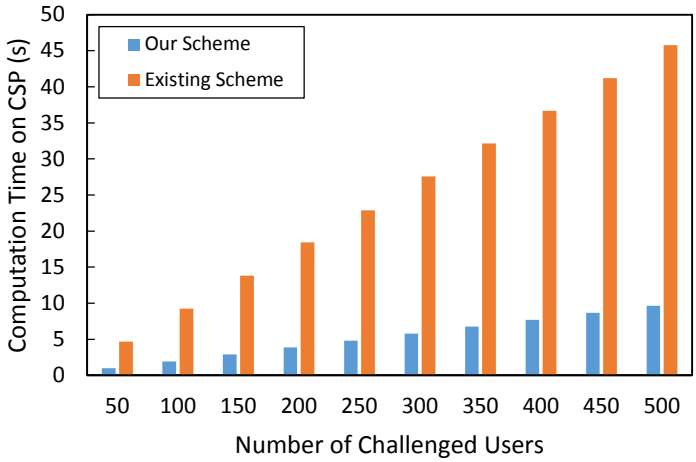


Figure 6. Performance Comparison of Batch Auditing on CSP (Single CSP, 5 blocks/User)

The performance of batch auditing for multiple CSPs is shown in Figure 7. It illustrates that our scheme incurs much less computation cost than the existing scheme. Although the computation time is almost constant on the CSP side with the increase of challenged CSPs, the accumulation of saved time will benefit the CSPs, especially in a large scale cloud storage system.

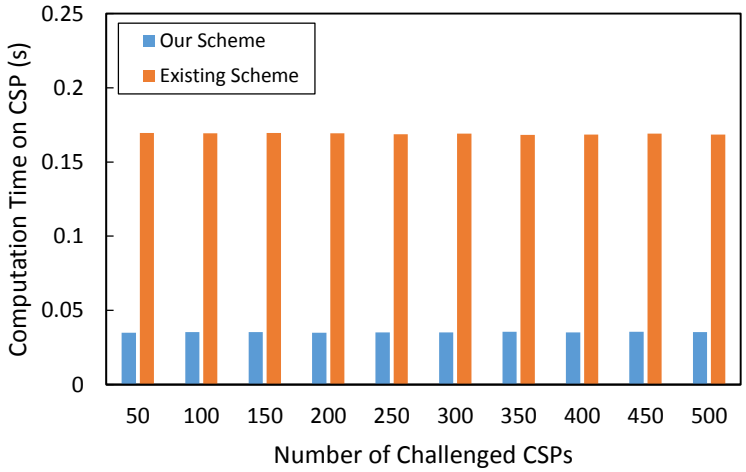


Figure 7. Performance Comparison of Batch Auditing on CSP (Single User, 5 blocks/CSP)

It is noteworthy that although our auditing scheme hugely reduces the computation cost on the CSP, it is achieved not by adding any computation loads to other parties like [17] but by combining random mask technique and bilinear map to design a new protocol. Thus, our auditing scheme is much more efficient when compared with Yang's efficient auditing schemes.

6. Conclusion

In this paper, we proposed a privacy-preserving public auditing scheme which also supports dynamic operations and batch auditing. We use homomorphic verifiable tags to construct the auditing scheme. Compared with previous work, the proposed scheme combines the random mask technique with the property of bilinear map, which improves the efficiency of both the TPA and CSP without moving any computation burden from TPA to CSP or other entities in the cloud. It can also prevent TPA from learning any information from the response proofs. Furthermore, the proposed scheme can greatly reduce the computation cost of TPA by achieving batch auditing for both multiple cloud users and multiple CSPs. Thus the proposed auditing scheme is efficient and can be applied into large scale cloud storage systems.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grant 61170221.

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and M. Zaharia, "A View of Cloud Computing", Communications of the ACM, vol. 53, no. 4, **(2010)**, pp. 50-58.
[2] P. Mell and T. Grance, "The NIST Definition of Cloud Computing", Communications of the ACM, vol. 53, no. 6, **(2010)**, pp.50.
[3] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen and A. V. Vasilakos, "Security and Privacy for Storage and Computation in Cloud Computing", Information Sciences, vol. 258, **(2014)**, pp. 371-386.
[4] S. Tan, Y. Jia and W. H. Han, "Research and Development of Provable Data Integrity in Cloud Storage", Chinese Journal of Computers, vol. 1, **(2015)**, pp. 13.
[5] W. Jansen and T. Grance, "Guidelines on Security and Privacy in Public Cloud Computing", NIST Special Publication, Gaithersburg, MD, United States, **(2011)**.
[6] K. Julisch and M. Hall, "Security and Control in the Cloud", Information Security Journal: A Global Perspective, vol. 19, no. 6, **(2010)**, pp. 299-309.
[7] D. G. Feng, M. Zhang, Y. Zhang and Z. Xu, "Study on Cloud Computing Security". Journal of Software, vol. 22, no. 1, **(2011)**, pp. 71-83.
[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable Data Possession at Untrusted Stores", Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, USA, **(2007)**, pp. 598-609.
[9] G. Ateniese, R. D. Pietro, L. V. Mancini and G. Tsudik, "Scalable and Efficient Provable Data Possession", Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, Istanbul, Turkey, **(2008)** September 22-25.
[10] C. C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic Provable Data Possession", ACM Transactions on Information and System Security, vol. 17, no. 4, **(2015)**, pp. 213-222 .
[11] Q. Wang, C. Wang, J. Li, K. Ren and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing", European Symposium on Research in Computer Security, Saint-Malo, France, **(2009)** September 21-23.
[12] R. Curtmola, O. Khan, R. Burns and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession", The 28th International Conference on Distributed Computing Systems, Beijing, China, **(2008)** June 17-20.
[13] A. F. Barsoum and M. A. Hasan, "Provable Possession and Replication of Data over Cloud Servers", Centre For Applied Cryptographic Research (CACR) Report, University of Waterloo, **(2010)**, pp. 1-32.
[14] A. F. Barsoum and M. A. Hasan, "On Verifying Dynamic Multiple Data Copies over Cloud Servers", IACR Cryptology ePrint Archive 2011, **(2011)**, pp. 447-476.
[15] C. Wang, S. S. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-preserving Public Auditing for Secure Cloud Storage", IEEE Transactions on Computers, vol. 62, no. 2, **(2013)**, pp. 362-375.
[16] Y. Zhu, H. Hu, G. J. Ahn and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 12, **(2012)**, pp. 2231-2244.

[17] K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing", IEEE Transactions on Parallel and Distributed Systems, vol. 24, no.9, **(2013)**, pp. 1717-1726.
[18] X. Liu and Y. Jiang, "Batch Auditing for Multi-client Dynamic Data in Multi-cloud Storage", International Journal of Security and Its Applications, vol. 8, no. 6, **(2014)**, pp. 197-210.
[19] M. Sookhak, A. Akhunzada, A. Gani, M. Khurram Khan and N. B. Anuar, "Towards Dynamic Remote Data Auditing in Computational Clouds", The Scientific World Journal 2014, **(2014)**, 12. DOI: http://dx.doi.org/10.1155/2014/ 269357.
[20] G. Yang, H. Xia, W. Shen, X. Jiang and J. Yu, "Public Data Auditing with Constrained Auditing Number for Cloud Storage", International Journal of Security and Its Applications, vol. 9, no. 9, **(2015)**, pp. 21-32.
[21] H. Shacham and B. Waters, "Compact Proofs of Retrievability", Journal of cryptology, vol. 26, no.3, **(2013)**, pp. 442-483.

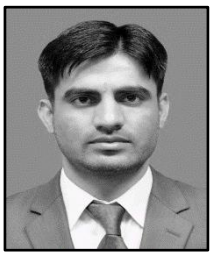
Authors

Haiyang Yu, he was born in 1991. He is currently a PhD candidate at the College of Computer Science, Beijing University of Technology. He received his BEng degree from Beijing University of Technology, China, in 2013. His research interests are Information Security and Cloud Computing.

Yongquan Cai, he is a Professor and PhD supervisor of College of Computer Science, Beijing University of Technology. His research interests include Information Security, Computer network and Cryptographic Protocols Analysis.

Shanshan Kong, she is a PhD candidate at the College of Computer Science, Beijing University of Technology. Her research interests are in the area of Information Security and Cloud Computing.

Fei Xue, he received his PhD degree from Beijing University of Technology, China, in 2016. His research interests include computational intelligence and complex systems.



Ditta Allah, he is currently a PhD candidate at Beijing University of Technology. His research interests are in the area of Information Security, Stegnography and Cryptography Protocols.