Hindawi
Security and Communication Networks
Volume 2018, Article ID 2615249, 11 pages
https://doi.org/10.1155/2018/2615249

Research Article

An Effective Integrity Verification Scheme of Cloud Data Based on BLS Signature

Xiling Luo,[1] Zequan Zhou,[1] Lin Zhong,[1,2] Jian Mao ,[2] and Chaoyong Chen[3]

[1]*School of Electronic and Information Engineering, Beihang University, Beijing, China*
[2]*School of Cyber Science and Technology, Beihang University, Beijing, China*
[3]*The Second Research Institute of Civil Aviation Administration of China, Beijing, China*

Correspondence should be addressed to Jian Mao; maojian@buaa.edu.cn

Received 8 September 2018; Accepted 7 November 2018; Published 19 November 2018

Guest Editor: Yan Huo

Cloud storage services allow users to outsource their data remotely to save their local storage space and enable them to manage resources on demand. However, once users outsourced their data to the remote cloud platform, they lose the physical control of the data. How to ensure the integrity of outsourced data is the major concern of cloud users and also is the main challenge in the cloud service deployment. Limited by the communication and computation overheads, traditional hash-based integrity verification solutions in the stand-alone systems cannot be directly adopted in remote cloud storing environment. In this paper, we improve the previous privacy preserving model and propose an effective integrity verification scheme of cloud data based on BLS signature (EoCo), which ensures public audition and data privacy preserving. In addition, EoCo also supports batch auditing operations. We conducted theoretical analysis of our scheme, demonstrated its correctness and security properties, and evaluated the system performance as well.

1. Introduction

Recent years, cloud storage services are getting more and more popular, which allow users to outsource their data remotely to save their local storage space and enable them to manage resources on demand. However, once users outsourced the data to the remote cloud platform, they lose the physical control of their data. Some cloud service providers (CSPs) may remove users' data that are less accessed to gain more profits. In addition, the loss of data may also be caused by system crashes or operation errors [1, 2]. In this situation, CSPs may conceal the fact that users' data have no longer been stored correctly. How to ensure the integrity of outsourced data is a major concern of cloud users. Being limited by the communication and computation overheads, traditional hash-based integrity verification solutions for stand-alone systems cannot be directly adopted in remote cloud storing environments. For example, a strawman solution is to compute and keep the message authentication code of the file before outsourcing and then retrieve the whole file from the cloud server and check with the message authentication code. However, this will introduce unacceptable communication and computational overhead.

To verify the integrity of cloud data without retrieving the whole outsourced files, Ateniese et al. [3] proposed a provable data possession (PDP) scheme, which provides remote integrity verification based on homomorphic tag and sampling techniques. Their improved approach supports public audition but cannot ensure the privacy preserving property. Juels et al. [4] proposed a proof of retrievability (POR) protocol to audit and ensure the correctness of remote data once data corruption happened. Nevertheless, the scheme only supports limited rounds of verification without privacy protection of public audition. The scheme proposed by Shacham et al. [5] cannot protect data privacy either. Although many other public auditing solutions have been proposed [6–12] that preserves privacy during the integrity audition, some schemes [6, 8, 9] only prevent the data leakage, rather than provide rigorous privacy guarantee. The solutions proposed by Wang et al. and Worku et al. [6, 9] fail to achieve reliable data protection under their security models.

In this paper, we propose an effective verification of cloud data integrity scheme, EoCo, based on BLS (Boneh-Lynn-Shacham) signature with strong privacy protection. Besides efficient remote data integrity verification, our scheme also supports public auditability, privacy protection, blockless verification, and batch audition. We formally define a new integrity protection model based on the work proposed by Worku et al. [9] and prove that dishonest server cannot bypass the verification based on the CDH (computational Diffie-Hellman) problem. We also introduce a ZK-privacy (zero knowledge privacy) model and prove that our scheme is secure against CMA (adaptive chosen messages attack) attacks and CPA (chosen plaintext attack) attacks under the security assumption of the cryptographic primitives.

Contributions. Our contributions can be summarized as follows:

(i) We propose a remote data integrity protection scheme that ensures public auditability, privacy protection, and blockless verification. Besides, our scheme may also support batch auditing operations.

(ii) We introduce and formally define a ZK-privacy model, where the adversary obtains zero knowledge from the auditing interactions. We prove the privacy preserving property of our scheme under the ZK-privacy model.

(iii) We evaluate the performance of our proposed scheme through mathematically analysis and compare it with related schemes in communication and computation overhead. The communication overhead of EoCo is only $O(1)$.

Paper Organization. The rest of this paper is organized as follows. In Section 2, we discuss the related work and the key challenges; Section 3 presents the preliminaries of the proposed scheme; in Section 4, we describe the system model and the security goals; in Section 5, we propose our publicly auditable integrity verification scheme; we conduct theoretical analysis with security proof in Section 6; in Section 7, we evaluate the performance overhead of our approach; and Section 8 concludes the paper.

2. Related Works

2.1. Provable Data Possession and Proof of Retrievability. Ateniese et al. [3] first proposed a remote auditing system using a provable data possession (PDP) model to ensure data integrity in untrusted storage services. To deploy RSA (Rivest-Shamir-Adleman) digital signature, the scheme splits data into small file blocks. The high probability guarantee of data integrity is achieved by randomly selecting some blocks and check the correctness by using the attribute of RSA homomorphic linear validation. Ateniese et al. [13] introduced the retrievability property on the basis of PDP using error-correcting codes. However, as the linear combination of sample file blocks is exposed to external auditors in the public auditing process, the above two solutions cannot achieve provable privacy protection. Juels et al. [4] proposed POR model to construct data integrity verification. The scheme ensures data integrity and retrievability through spot-checking and error-correcting codes. Spot-checking is to randomly embed special check blocks, sentinels, into the data file, and then randomly select a number of sentinels to verify the file data's integrity. However, this scheme has a critical problem that once the times of validations is beyond a certain number, these fixed sentinels will be exposed and the data integrity will not be guaranteed. In addition, the scheme cannot support public audition. Bowers et al. [14] improved the POR model, and Shacham et al. [5, 15] also proposed an improved POR scheme base on BLS signature [16]. However, these schemes cannot ensure privacy protection for the same reason as the scheme proposed by Ateniese et al. [3]. Shah et al. [12, 17] introduced a third party auditor (TPA) and sent a number of precomputed symmetric-keyed hashes over the encrypted data to the TPA. Nevertheless, this scheme can only be applied to encrypted files and used in a limited way. When the keyed hashes are used up, this scheme will give the user additional online burden. Worku et al. [9] redefined a new integrity protection model that ensured a stronger definition of integrity by adding a second query phase and also proposed a scheme that claims to be able to acquire provable security under the model. However, Liu et al. [11] proved that the scheme in [9] does not satisfy the definition of its own security mode. Worku's scheme cite 10 selects a unique identifier *name* for each file, but the *name* cannot be well embedded into the scheme and is not tightly connected to the scheme. Therefore, the adversary can extract important knowledge in the second inquiry phase of the security model [18].

2.2. Public Audition. Wang et al. [6] proposed a public audition scheme with the privacy protection property that an adversary cannot obtain the information of data in the PDP model. Worku et al. [9] presented that the Wang's scheme [6] is not secure against the attacks from malicious servers and proposed a privacy preserving scheme. Wang et al. [8] proposed an improved scheme to achieve the property of privacy protection. However, these schemes [6, 8, 9] cannot satisfy the definition of the strict privacy protection model, IND-privacy (indistinguishability-privacy), presented by Fan et al. [10]. The IND-privacy model achieves privacy protection by proving the indistinguishability of responses in the auditing process to external auditors. The general idea and design of Worku's scheme [9] and Wang's scheme [8] are similar. The former uses $\mu = \mu' + rh(R)$ to hide the linear combination μ' of the blocks, while the latter is implemented by $\mu = r + h(R)\mu'$, where r and R are the random numbers and h stands for hash function. These two ways can only ensure that external auditors cannot obtain the relevant information of the data, but for the malicious auditors, it is easy to distinguish the different information to obtain the relevant knowledge of the cloud users. The protocol proposed by Fan et al. [10] is inefficient and its symmetric external Diffie-Hellman assumption can be solved in the presence of bilinear mapping.

Table 1: Schemes attribute comparison.

	Ateniese-[3]	Juels-[4]	[5]	Wang-[6]	Wang-[8]	Worku-[9]	Our scheme
Data integrity protection	Yes	Yes	Yes	No	Yes	No	Yes
Public auditability	Yes	No	Yes	No	Yes	Yes	Yes
Privacy protection	No	No	No	No	Yes	Yes	Yes
Blockless verification	Yes	No	Yes	Yes	Yes	Yes	Yes
Batch auditing	No	No	No	No	Yes	Yes	Yes
Retrievability	No	Yes	No	No	No	No	Yes
Audit times	Unlimit	limit	limit	Unlimit	Unlimit	Unlimit	Unlimit
Communication	$O(1)$	$O(n)$	$O(n)$	$O(n)$	$O(n)$	$O(1)$	$O(1)$

2.3. Dynamic Maintenance. In practical settings, supporting dynamic maintenance is desired in remote data attestation. Ateniese et al. [19] proposed the concept of dynamic operation and built a scheme that does not require batch encryption based on a symmetric cryptosystem. However, this scheme is limited to the number of queries and does not support full sense of dynamic scenarios. Wang et al. [7, 8] supported full dynamic operations by using Merkle hash tree (MHT) structure. Erway et al. [20] developed a skiplist-based scheme to enable the integrity of data with full dynamics operations. Sookhak et al. [21] also proposed a dynamic scheme. Xin et al. [22] proposed an effective and secure access control approach for multiauthority cloud storage systems.

In addition, the communication and computational overheads are also critical metrics to evaluate the efficiency of cloud services [23, 24]. Among the schemes above, the communication overhead of [8] during validation is $O(n)$, while [9] is more efficient in computational overhead. We illustrate the comparison of the relevant solutions in Table 1.

3. Preliminaries

3.1. Bilinear Map and GDH Groups. Let G_1, G_2, and G_T be multiplicative cyclic groups of the same large prime order p, where $|G_1| = |G_2| = |G_T|$. Let g_1, g_2 be the generators of G_1, G_2, respectively. $e : G_1 \times G_2 \longrightarrow G_T$ is a bilinear map if it satisfies the following properties:

(i) Bilinear: $\forall u \in G_1, v \in G_2$ and $\forall a, b \in Z_p$, $e(u^a, v^b) = e(u, v)^{ab}$ holds.

(ii) Nondegenerate: $\exists u \in G_1, v \in G_2$, such that $e(u, v) \neq I_{G_T}$, I_{G_T} is the identity element of the cyclic group G_T.

If there exists an isomorphism $\psi : G_2 \longrightarrow G_1$, with $\psi(g_2) = g_1$, (G_1, G_2) is a Gap Diffie-Hellman (GDH) group pair. We can set $G_1 = G_2 = G$ and $g_1 = g_2 = g$ and take ψ to be the identity map.

(a) Computational co-Diffie-Hellman (co-CDH) on GDH [16]. Given $g_2, g_2^\alpha \in G_2$ and $h \in G_1$ as input and $\alpha \in Z_p$, compute $h^\alpha \in G_1$.

(b) Decision co-Diffie-Hellman (co-DDH) on GDH [16]. Given $g_2, g_2^\alpha \in G_2$ and $h, h^\beta \in G_1$ as input, if $\alpha = \beta$ the output is `yes`; otherwise, the output is `no`. When the answer is `yes`, we say that $(g_2, g_2^\alpha, h, h^\alpha)$ is a co-Diffie-Hellman tuple.

When $G_1 = G_2$, these problems reduce to a standard CDH and DDH problems. The co-DDH problem is easy to be solved but co-CDH is hard on the GDH group [16].

3.2. BLS Signature. The BLS signature [16] includes three functions, **KeyGen**, **Sign**, and **Verify**. Let (G_1, G_2) be a GDH group where $|G_1| = |G_2| = p$. It makes use of a full-domain hash function $H : \{0,1\}^* \longrightarrow G_1$.

(i) **KeyGen.** Randomly choose $x \longleftarrow Z_p$ and compute $v \longleftarrow g_2^x$. The public key is $v \in G_2$ and the private key is x.

(ii) **Sign.** Given a private key $x \in Z_p$ and a message $M \in \{0,1\}^*$, compute $h \longleftarrow H(M) \in G_1$ and $\sigma \longleftarrow h^x$. The signature is $\sigma \in G_1$.

(iii) **Verify.** Given a public key $v \in G_2$, a message $M \in \{0,1\}^*$, and the signature $\sigma \in G_1$, compute $h \longleftarrow H(M) \in G_1$ and verify whether (g_2, v, h, σ) is a valid co-Diffie-Hellman tuple. If so, output *valid*; otherwise, output *invalid*.

4. Approach Overview

4.1. System Model. Our scheme, EoCo, is built on the system model presented in Figure 1. The model consists of three entities, *Users*, *CSP Servers*, and *The Third Party Auditor*.

(i) Users. Cloud users own the data and want to save local storage and computing resources by uploading them to the cloud.

(ii) CSP Servers. The CSP servers have a large amount of storage space available for users. At the same time, CSP servers provide effective cloud operations such as data update and queries and retrieve requests from the customers.

(iii) The Third Party Auditor (TPA). The TPA has more ability and expertise than the clients. Users could ask a TPA help to audit the integrity of the outsourced data on behalf of them.

In our system model, our remote data integrity protection scheme consists of five critical operations, *KeyGen*, *TokenGen*, *Challenge*, *Response*, and *CheckProof*. A cloud user runs **KeyGen** to generate her/his public key *pk* and private key *sk* (with a security parameter k as its input). The user

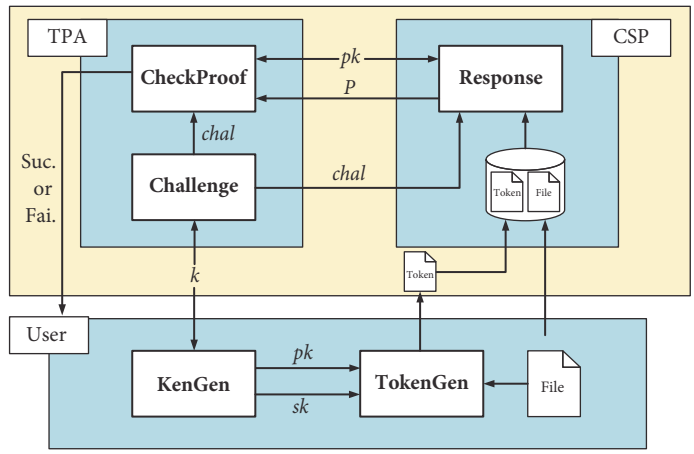

Figure 1: The EoCo-based network system model.

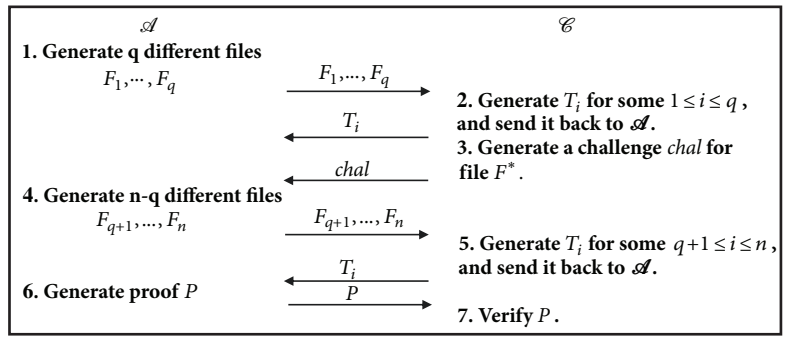

Figure 2: Integrity protection game.

runs the **TokenGen** algorithm to generate a token for the file and outsources the file with corresponding token to the CSP server. When the user wants to check the correctness of her/his data, she/he may delegate the integrity audition to a TPA. The TPA creates *chal* using **Challenge** and sends it to the CSP server. The server runs **Response** and returns the proof P to TPA. The TPA runs **CheckProof** to check whether it is correct, and if it is correct, the output is *success*; otherwise, the output is *failure*.

In the system shown in Figure 1, we mainly take two threats in to account, *the threat to integrity,* and *the threat to privacy* [25]. Accordingly, we classify attackers into two types based on the knowledge H that they processed.

(i) **Threat to Integrity.** The attacker observes the data DA, the authentication identifier T, and the public key pk, i.e., $H =< DA, T, pk >$. The purpose of such attacker is to produce a legitimate proof for the forge DA.

(ii) **Threat to Privacy.** The attacker observes only the public key pk and the proof P, i.e., $H =< P, pk >$. The purpose of such attacker is to acquire additional knowledge, such as the content of data or the type of data.

4.2. Integrity Protection Model. We improve the integrity protection model proposed by Worku et al. [9] and through this model, we demonstrate two objectives.

(1) If the data are not stay the original state, an adversary cannot successfully construct a valid proof in polynomial time with nonnegligible probability.

(2) If the adversary can always pass the verification, then it can be shown that the data remains intact.

Our integrity protection model allows an adversary to query large files F_i. The adversary $\mathscr{A}$, may be a dishonest cloud service provider who interacts with a challenger $\mathscr{C}$ (users or TPA). The integrity game consists of the following steps and we illustrate the details in Figure 2.

(i) **Setup.** The challenger $\mathscr{C}$ runs the algorithm of key generation, sends the public key pk to the adversary $\mathscr{A}$, and retains the private key sk secret.

(ii) **Query1.** The adversary $\mathscr{A}$ adaptively makes tagging queries: it selects a file F_1 and sends it to $\mathscr{C}$. The challenger $\mathscr{C}$ then computes the token T_1 and sends it back to $\mathscr{A}$. The adversary $\mathscr{A}$ continues to query $\mathscr{C}$ for the token $T_2, \ldots, T_q$ on the files of its choice $F_2, \ldots, F_q$. In general, the challenger $\mathscr{C}$ generates T_i for some $1 \le i \le q$.

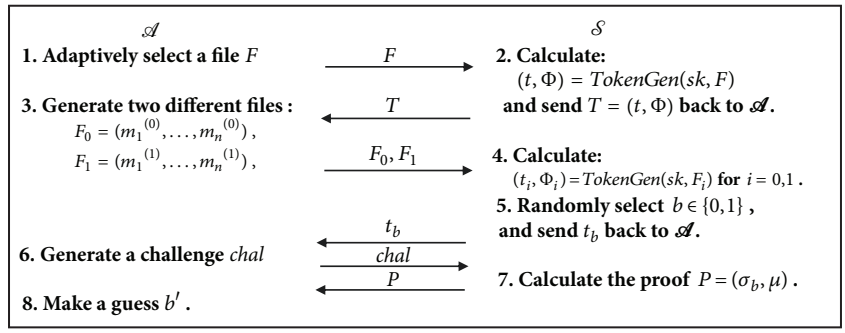

Figure 3: ZK-privacy game.

(iii) **Challenge.** The challenger $\mathscr{C}$ generates a challenge *chal* and requests the adversary $\mathscr{A}$ to provide a proof of integrity for the file F^* (note that file F^* must differ from the F_i in query phase).

(iv) **Query2.** Repeat Query1, the challenger $\mathscr{C}$ generates T_i for some i, $q + 1 \leq i \leq n$.

(v) **Forge.** The adversary $\mathscr{A}$ computes a proof of integrity P for the file F^* according to challenge *chal* and returns P to $\mathscr{C}$.

If the P can pass the verification, the adversary $\mathscr{A}$ wins the game.

Definition 1. An EoCo scheme guarantees data integrity if for any polynomial time adversary $\mathscr{A}$ cannot win the game with nonnegligible probability.

4.3. Privacy Protection Model. In this subsection, we define a new privacy protection model, *the ZK-privacy model*, and prove that the scheme does not have any information leakage by showing that the attacker has zero knowledge during the audit process. The ZK-privacy model takes place between a challenger $\mathscr{S}$ such as cloud server and an adversary $\mathscr{A}$ such as malicious TPA. The game model includes the following steps and we summarize the critical operations in Figure 3.

(i) **Setup.** The challenger $\mathscr{S}$ runs the algorithm of key generation to generate key pair (pk, sk) and sends the public key pk to the adversary $\mathscr{A}$.

(ii) **Phase 1 (Steps 1–3).** The adversary $\mathscr{A}$ adaptively makes queries: it selects a file F and sends it to the challenger $\mathscr{S}$, then the challenger $\mathscr{S}$ generates the token, $T = (t, \Phi)$, for the file F, and sends it to the adversary $\mathscr{A}$.

(iii) **Phase 2 (Steps 4–7).** The adversary $\mathscr{A}$ chooses two files F_0, F_1, and $F_0 \neq F_1$, which are different from the files in Phase 1. Then the challenger $\mathscr{S}$ generates corresponding $T_0 = (t_0, \Phi_0), T_1 = (t_0, \Phi_0)$. Next, the challenger $\mathscr{S}$ randomly selects $b \in \{0, 1\}$ and sends t_b to $\mathscr{A}$. The adversary $\mathscr{A}$ generates a challenge *chal* to $\mathscr{S}$. The challenger $\mathscr{S}$ generates proof, $P_b = (\sigma_b, \mu)$, to the $\mathscr{A}$. Finally, the adversary $\mathscr{A}$ outputs a bit b' as a guess of the b. If $b' = b$, the adversary wins the game.

Definition 2. We define the advantage of the adversary $\mathscr{A}$ as

$$Adv_{\mathscr{A}}(\lambda) = \left| View_{\mathscr{A}}(b = 0) - View_{\mathscr{A}}(b = 1) \right| \quad (1)$$

The $View_{\mathscr{A}}(b = 0 \quad or \quad 1)$ indicates the probability distribution of the view of $\mathscr{A}$ in the case of $b = 0$ or $b = 1$. An EoCo scheme guarantees that there is no an information leakage if $Adv_{\mathscr{A}}(\lambda) = 0$.

5. Our Schemes

5.1. Definition and Framework. Our scheme mainly consists of five polynomial time algorithms: KeyGen, TokenGen, Challenge, Response, and CheckProof. The cloud user is represented by C, the cloud server is represented by S, and the third party auditor is represented by TPA.

(i) $KeyGen(1^k) \longrightarrow (pk, sk)$. *KeyGen* is a probabilistic key generation algorithm, which is set up and initialized by C. It takes as input a security parameter k and outputs a pair of key (pk, sk).

(ii) $TokenGen(sk, pk, F) \longrightarrow T$. *TokenGen* is a deterministic algorithm to compute tokens. It takes as inputs a private key sk, a public key pk and a file $F = (m_1, \ldots, m_n)$. The output is the token $T = (t, \Phi = \{\sigma_i\}_{1 \leq i \leq n})$, where t is a file tag which includes a file *name*, Φ is an ordered collection, and σ_i is the unique authentication identifier corresponding to the blocks m_i in the file F.

(iii) $Challenge(1^k) \longrightarrow chal$. *Challenge* is a deterministic algorithm. It takes as input a security parameter k. It outputs a challenge *chal*.

(iv) $Response(pk, chal, \Phi, F) \longrightarrow P$. *Response* is to create the proof of the data integrity corresponding to the received challenge. It takes as inputs a public key pk, a file F, a challenge *chal*, and an ordered set Φ. The output is an integrity proof P of the file F.

(v) $CheckProof(pk, P, chal, t) \longrightarrow (suc.\ or\ fail.)$. *CheckProof* is the function to verify the returned proof P. It takes as inputs a public key pk, a challenge *chal*, and a proof P as input. It outputs *success* or *failure*.

Table 2: Notation list of symbols and parameters.

Symbols	Description
p	a large prime
Z_p	a group of integers with order p
G	a cyclic group
g	an element of G
H	a hash function, $\{0,1\}^* \longrightarrow G$
π	a pseudo-random permutation, $\{0,1\}^{\log_2(n)} \times K \longrightarrow \{0,1\}^{\log_2(n)}$
f	a pseudo-random functions, $\{0,1\}^* \times K \longrightarrow Z_p$
$SSig(\cdot)$	an additional BLS signature
$F = (m_1, ..., m_n)$	divide a big data file F into n small file blocks and $m_i \in Z_p$

A EoCo scheme can be summed up as two processes: one is the setup process to initialize the scheme, and the other is the verification process to confirm whether the data is integrity.

(i) **Setup.** The cloud user C has a file F and runs the algorithm **KeyGen** to generate the key pair (pk, sk). C then stores the key pair locally and uses it to run the algorithm **TokenGen** to generate a token, $T = (t, \Phi = \{\sigma_i\}_{1 \le i \le n})$. Finally, C sends T and F to S and deletes F and T from its local storage.

(ii) **Audition.** TPA runs the algorithm *Challenge* to generate a challenge *chal* corresponding to the target files. TPA sends *chal* to the server S. After receiving *chal*, S computes the proof P accordingly and returns the TPA the proof P and t by using the function *Response*. The TPA validates P and t using the function *CheckProof*.

5.2. Notations. In this subsection, we illustrate the parameters and notations used in our scheme. As listed in Table 2, p is a large prime and the EoCo scheme is built on the group Z_p and supports a cyclic group G with a bilinear setting. Let g be an element of cyclic group. In the setup phase, the date owner splits the file F into n small file blocks, that is, $F = (m_1, \ldots, m_n)$, and for $1 \le i \le n$, every block $m_i \in Z_p$. $H : \{0,1\}^* \longrightarrow G$ is a secure hash function. The additional BLS signature used in our scheme is represented as $SSig(\cdot)$. In addition, we take advantage of a pseudorandom permutation (PRP) π and a pseudorandom functions (PRF) f with the following parameters. We write $f_k(x)$ to denote f keyed with key k applied on input x.

(i) $\pi : \{0,1\}^{\log_2(n)} \times k \longrightarrow \{0,1\}^{\log_2(n)}$.

(ii) $f : \{0,1\}^* \times k \longrightarrow Z_p$.

5.3. EoCo Scheme

(i) **KeyGen**(1^k)**.** The algorithm first generates a secure BLS signature key pair (spk, ssk). Next, the algorithm randomly selects an element in the Z_p, $x \longleftarrow Z_p$, and computes $v \longleftarrow g^x \in G$. Then, the algorithm randomly selects an element in the G, $u \longleftarrow G$, and computes $w \longleftarrow u^x$. Finally, the algorithm returns public key $pk = (spk, g, v, u, w)$ and private key $sk = (ssk, x)$.

(ii) **TokenGen**$(sk, pk, F = (m_1, \ldots, m_n))$**.** The algorithm randomly selects an element in the Z_p as a unique identifier of file F, $name \longleftarrow Z_p$, and calculates file identification tag by BLS signature, $t = name \parallel SSig_{ssk}(name)$. Next, the algorithm generates unique authentication identifier for each block m_i in the file, $\sigma_i \longleftarrow (H(W_i) \cdot u^{m_i})^x \in G$. For $1 \le i \le n$, $W_i = name \parallel i$. The algorithm then outputs token $T = (t, \Phi = \{\sigma_i\}_{1 \le i \le n})$.

(iii) **Challenge**(1^k)**.** The algorithm first determines the number of file blocks c. According to the security parameters k, the algorithm randomly selects key k_1 for π and key k_2 for f. Finally, the algorithm returns $chal = (c, k_1, k_2)$.

(iv) **Response**$(pk, chal, \Phi = \{\sigma_i\}_{1 \le i \le n}), F)$**.** The algorithm first selects a secret random value $r \longleftarrow Z_p$. For $1 \le j \le c$, the algorithm computes $i_j = \pi_{k_1}(j)$ and $a_j = f_{k_2}(j)$, and the algorithm then computes aggregated authentication identifier $\sigma = \sigma_{i_1}^{a_1} \cdot \sigma_{i_2}^{a_2}, \ldots, \cdot \sigma_{i_c}^{a_c} \cdot w^r$ (note that σ_{i_j} is the i_j-th value in Φ). $\mu = v^{\mu' + r} \in G$, and $\mu' = a_1 m_{i_1} +, \ldots, a_c m_{i_c}$. Finally, the algorithm outputs proof $P = (\sigma, \mu)$.

(v) **CheckProof**$(pk, P, chal, t)$**.** The algorithm first verifies whether the file tag t is correct. If t is incorrect, the algorithm outputs *failure* and quit; otherwise the algorithm extracts *name* and for $1 \le j \le c$, the algorithm calculates $i_j = \pi_{k_1}(j)$, $a_j = f_{k_2}(j)$ and $W_{i_j} = name \parallel i_j$. Finally, the algorithm verifies the following equation:

$$e(\sigma, g) = e\left(\prod_{j=1}^{c} H\left(W_{i_j}\right)^{a_j}, v\right) \cdot e(u, \mu) \tag{2}$$

If (2) holds, *CheckProof* returns *success*; otherwise, it returns *failure*.

6. Scheme Analysis

6.1. Correctness. In this subsection, we discuss the correctness of our scheme; that is, the CSPs will definitely pass through the audition if they follow the protocol honestly. By signing the *name* of a file with an additional BLS signature,

we can prevent an adversary from tampering with the *name*. We use a hash function to ensure that the file tag, *name*, is perfectly embedded in the authentication identifier. We make use of a PRP function and a PRF function to ensure the randomness of the challenge content and the security of the response. If the data is kept properly, the correctness of the verification equation (2) can be shown as follows:

$$\begin{aligned} & e\left(\prod_{j=1}^{c} H\left(W_{i_j}\right)^{a_j}, v\right) \cdot e(u, \mu) \\ &= e\left(\prod_{j=1}^{c} H\left(W_{i_j}\right)^{a_j}, g^x\right) \cdot e\left(u^{\mu'+r}, g^x\right) \\ &= e\left(\prod_{j=1}^{c} H\left(W_{i_j}\right)^{a_j} u^{\mu'}, g^x\right) \cdot e\left(u^r, g^x\right) \\ &= e\left(\prod_{j=1}^{c} \sigma_{i_j}^{a_j}, g\right) \cdot e\left(\left(u^x\right)^r, g\right) = e\left(\prod_{j=1}^{c} \sigma_{i_j}^{a_j} w^r, g\right) \\ &= e(\sigma, g) \end{aligned} \tag{3}$$

6.2. Integrity Protection. The integrity protection of EoCo is based on the symmetric co-CDH problem or standard CDH problem.

Theorem 3. *Under the CDH assumptions, EoCo guarantees data integrity protection in the random oracle model.*

We first propose a simplified scheme, S-EoCo. By proving the security of S-EoCo, we can prove the security of EoCo. S-EoCo and EoCo differ only in the ***KenGen*** *and* ***Response*** *algorithms as follows:*

(i) ***KeyGen***(1^k)***.*** *The algorithm first generates a secure BLS signature key pair* (spk, ssk)*. Next, the algorithm randomly selects an element in the* Z_p*,* $x \longleftarrow Z_p$*, and computes* $v \longleftarrow g^x \in G$*. Then, the algorithm randomly selects an element in the G,* $u \longleftarrow G$*. Finally, the algorithm returns public key* $pk = (spk, g, v, u)$ *and private key* $sk = (ssk, x)$*.*

(ii) ***Response***$(pk, chal, \Phi = \{\sigma_i\}_{1 \le i \le n}), F)$***.*** *For* $1 \le j \le c$*, the algorithm computes* $i_j = \pi_{k_1}(j), a_j = f_{k_2}(j)$*. Then, for* $I = \{i_1, i_2, \ldots, i_c\}$*, the algorithm computes aggregated authentication identifier* $\sigma = \sigma_{i_1}^{a_1} \cdot \sigma_{i_2}^{a_2}, \ldots, \cdot \sigma_{i_c}^{a_c}$*,* $\mu = v^{\mu'} \in G$*, where* $\mu' = a_1 m_{i_1} +, \ldots, a_c m_{i_c}$*. Finally, the algorithm outputs proof* $P = (\sigma, \mu)$*.*

Proof. We reduce the security of our S-EoCo scheme to the security of the CDH problem. We model hash function $H(\cdot)$ as random oracle.

If an adversary $\mathscr{A}$ can break the integrity protection of the S-EoCo scheme, we show how to construct an adversary $\mathscr{C}$ that uses $\mathscr{A}$ in order to break CDH problem.

For the CDH problem, $\mathscr{C}$ is given $(g, g^\alpha, h) \in G$ and needs to calculate h^α. Then $\mathscr{C}$ will play the role of the challenger in the integrity protection game and will interact with $\mathscr{A}$ as follows.

(i) **Setup.** The challenger $\mathscr{C}$ selects a secret key pair (spk, ssk) and gets $pk = (spk, g, v, u)$, where $u = g^a h^b$, $a \longleftarrow Z, b \longleftarrow Z_p$, $v = g^\alpha$. Then $\mathscr{C}$ sends public key to the adversary $\mathscr{A}$.

(ii) **Query1.** The adversary $\mathscr{A}$ adaptively presents queries in the way that it selects different files $F_1, \ldots, F_n$ and sends them to $\mathscr{C}$ to create *Token*s. In order to answer queries, $\mathscr{C}$ simulates in a random oracle machine as follows:

(1) For $F = \{m_1, m_2, \ldots, m_n\}$, randomly selects an element in group Z_p as the unique identifier of file F, $name \longleftarrow Z_p$, and calculates file identification tags $t = name \parallel SSig_{ssk}(name)$.

(2)

(a) When $\mathscr{A}$ queries j for hash value, $\mathscr{C}$ first checks whether j is in the hash tuple list $(j, H(j))$. If it is in the list, $\mathscr{C}$ sends $H(j)$ to $\mathscr{A}$ as the answer; otherwise, $\mathscr{C}$ randomly selects and replies $\mathscr{C}$ an elements in G, $\omega \longleftarrow G$, as $H(j)$, and adds $H(j)$ to the hash tuple list $(j, H(j))$.

(b) When $\mathscr{A}$ performs the identifier query, $\mathscr{C}$ calculates $W_i = name \parallel i$. If $W_i = j$, $\mathscr{C}$ announces *failure*; otherwise, $\mathscr{C}$ randomly selects $r_i \longleftarrow Z_p$, calculates $H(W_i) = g^{r_i} \cdot g^{-am_i} \cdot h^{-bm_i}$, and builds a list of identifiers $(W_i, m_i, r_i, H(W_i), \sigma_i)$. $\mathscr{C}$ can compute σ_{m_i} using the formula below.

$$\begin{aligned} \sigma_{m_i} &= \left(H(W_i) \cdot u^{m_i}\right)^\alpha = \left(g^{r_i} \cdot g^{-am_i} \cdot h^{-bm_i} \cdot u^{m_i}\right)^\alpha \\ &= \left(g^{r_i} \cdot g^{-am_i} \cdot h^{-bm_i} \cdot g^{am_i} \cdot h^{bm_i}\right)^\alpha = \left(g^{r_i}\right)^\alpha \\ &= \left(g^\alpha\right)^{r_i} \end{aligned} \tag{4}$$

(3) $\mathscr{C}$ sends the file identifier $T = (t, \Phi = \{\sigma_i\}_{1 \le i \le n})$ to $\mathscr{A}$.

(iii) **Challenge.** For files $F^* = \{m_1^*, m_2^*, \ldots, m_n^*\}$, $\mathscr{C}$ sends challenge *chal* to $\mathscr{A}$, and F^* is different from above F that have been query.

(iv) **Query2.** Repeat the Query1, but $\mathscr{A}$ cannot query the file blocks included in *chal*.

(v) **Forge.** $\mathscr{A}$ creates and returns P to $\mathscr{C}$ according to challenge *chal*. If P can pass through the verification, $\mathscr{A}$ wins the game.

Suppose that an honest cloud server computes a proof, $P = (\sigma, \mu)$, then the following verification formula should be satisfied:

$$e(\sigma, g) = e\left(\prod_{i \in I} H(W_i)^{a_i}, v\right) \cdot e(u, \mu) \tag{5}$$

Suppose that the adversary$\mathscr{A}$ wins games with nonnegligible probability. The proof output by $\mathscr{A}$ is $P^* = (\sigma^*, \mu^*)$ that

can pass the verification with nonnegligible probability, thus formula (6) is satisfied.

$$e(\sigma^*, g) = e\left(\prod_{i\in I} H(W_i)^{a_i}, v\right) \cdot e(u, \mu^*) \tag{6}$$

According to the game, $\mu \neq \mu^*$. Let $\Delta\mu' = \mu'^* - \mu'$. Because G is a multiplicative cyclic group, we can find an inverse element of σ in the group. We substitute σ^{-1} for σ in formula (5) to get the deformation of formula (5).

$$e\left(\sigma^{-1}, g\right) = e\left(\prod_{i\in I} H(W_i)^{a_i}, v\right)^{-1} \cdot e\left(u, \mu^{-1}\right) \tag{7}$$

We multiply σ^* by σ^{-1} and obtain the following equation:

$$\begin{aligned} e\left(\sigma^*\sigma^{-1}, g\right) &= e\left(u, \mu^*\mu^{-1}\right) = e\left(u, v^{\mu'^* - \mu'}\right) \\ &= e\left(u, v^{\Delta\mu'}\right) = e\left(u^{\Delta\mu'}, v\right) \end{aligned} \tag{8}$$

Because $u = g^a h^b$, we replace u in formula (8).

$$\begin{aligned} e\left(\sigma^*\sigma^{-1}, g\right) &= e\left(\left(g^a h^b\right)^{\alpha\Delta\mu'}, g\right) \\ &= e\left(\left(g^a\right)^{\alpha\Delta\mu'}\left(h^b\right)^{\alpha\Delta\mu'}, g\right) \\ &= e\left(v^{a\Delta\mu'}\left(h^\alpha\right)^{b\Delta\mu'}, g\right) \end{aligned} \tag{9}$$

The above equation can be further deformed as follows:

$$e\left(\sigma^*\sigma^{-1}v^{-a\Delta\mu'}, g\right)^{-b\Delta\mu'} = e\left(h^\alpha, g\right) \tag{10}$$

It can be derived from formula (10) that $h^\alpha = (\sigma^*\sigma^{-1}v^{-a\Delta\mu'})^{-b\Delta\mu'}$. $\mathscr{C}$ takes back μ'^* from the adversary $\mathscr{A}$ and calculates $\Delta\mu' = \mu'^* - \mu'$. It means that as long as $b\Delta\mu' \neq 0 \mod p$ $\mathscr{C}$ can solve the CDH problem. As $\Delta\mu' \neq 0 \mod p$, the probability that $b\Delta\mu' \neq 0$ is $1 - 1/p$, which is nonnegligible. And the probability that simulation failure possibility of $\mathscr{C}$ is $1/q$ is negligible. So if the adversary wins the game with a nonnegligible probability, the CDH problem can be solved with a nonnegligible probability. Aggregation tags in **Response** in the EoCo scheme is that $\sigma = \sigma_{i_1}^{a_1} \cdot \sigma_{i_2}^{a_2}, \ldots, \cdot\sigma_{i_c}^{a_c} \cdot w^r$ and $\mu = v^{\mu'+r} \in G$. We add w^r on the basis of S-EoCo that makes scheme stronger.

Here we proved the Theorem 3. □

6.3. Privacy Preserving

Theorem 4. *According to Definition 2, an adversary is completely zero knowledge if, for any polynomial time algorithm $\mathscr{A}$, $Adv_{\mathscr{A}}(\lambda) = 0$.*

Proof. To prove Theorem 4, we construct a simulator $\mathscr{S}$ to interact with an adversary $\mathscr{A}$ following the steps below.

(1) $\mathscr{A}$ selects two different files $F_0 = (m_1^{(0)}, \ldots, m_n^{(0)})$ and $F_1 = (m_1^{(1)}, \ldots, m_n^{(1)})$, for $1 \leq i \leq n$, which satisfy $m_i^{(0)} \neq m_i^{(1)}$.

(2) The simulator $\mathscr{S}$ generates $T_0 = (t_0, \Phi_0)$ and $T_1 = (t_1, \Phi_1)$ for the files F_0 and F_1, respectively. Then $\mathscr{S}$ randomly selects $b \in \{0, 1\}$ and sends t_b to $\mathscr{A}$.

(3) When $\mathscr{A}$ received t_b, $\mathscr{A}$ randomly generates a challenge $chal = \{c, k_1, k_2\}$ and sends $chal$ to $\mathscr{S}$ request proof of F_b

(4) The simulator $\mathscr{S}$ calculates and sends response proof $P_b = (\sigma_b, \mu)$ to $\mathscr{A}$.

(5) The identifier in P_b is that $\sigma_b = \sigma_{i_1}^{a_1} \cdot \sigma_{i_2}^{a_2}, \ldots, \cdot\sigma_{i_c}^{a_c} \cdot w^r$, $\mu = v^{\mu'+r} \in G$, and $\mu' = a_1 m_{i_1}^{(b)} +, \ldots, a_c m_{i_c}^{(b)}$, and it satisfies

$$e(\sigma, g) = e\left(\prod_{i\in I} H(W_i)^{a_i}, v\right) \cdot e(u, \mu) \tag{11}$$

From the perspective of $\mathscr{A}$, $\mathscr{A}$ can only see (P_b, F_0, F_1, pk). Since $\mathscr{A}$ cannot know the value of r (unless $\mathscr{A}$ can solve the discrete logarithm (DL) problem) and r is randomly and uniformly distributed, for the adversary $\mathscr{A}$, P_0 and P_1 also are uniformly distributed.

Since pk is public, so $View_{\mathscr{A}}(P_0, F_0, F_1, pk)$ and $View_{\mathscr{A}}(P_1, F_0, F_1, pk)$ have the same probability distribution. The advantage of the adversary $\mathscr{A}$ is as follows:

$$\begin{aligned} Adv_{\mathscr{A}}(\lambda) = \big|&Pr\left[View_A\left(P_0, F_0, F_1, pk\right)\right] \\ &- Pr\left[View_A\left(P_1, F_0, F_1, pk\right)\right]\big| = 0 \end{aligned} \tag{12}$$

The above equation shows that the protocol is strict zero knowledge to the adversary $\mathscr{A}$, and thus, the adversary cannot get any information through the auditing process.

Here we complete the proof. □

6.4. Batch Auditing. Sometimes, a TPA will audit multiple files on behalf of different users. While one by one validation is too tedious and inefficient, it is desired that the TPA could parallelly audit the integrity of multiple cloud user data. Suppose that Q cloud users are entrusted with the same TPA. We slightly modify our scheme by using BLS signature to aggregate different signatures and thus provide the effective verification for all Q users/files simultaneously. In fact, our attribute of batch auditing is to enhance the audit efficiency of TPA. Therefore, in EoCo scheme, we only need to make minor modifications on the **CheckProof** algorithm.

(i) **CheckProof**$(pk, P, chal, t)$**.** For $1 \leq l \leq Q$, the algorithm first verifies whether the file tag t_l is correct. The algorithm then outputs *failure* if it is wrong; otherwise, the algorithm extracts $name_l$ and continues. Next, for $1 \leq j \leq c$, the algorithm calculates $i_j = \pi_{k_1}(j), a_j = f_{k_2}(j)$ and $(W_{i_j})_l = name_l \parallel i_j$ and computes aggregate authentication

Table 3: Communication complexity comparison.

	SW-[5]	Wang-[8]	Worku-[9]	Our scheme
Challenge form and the number of bits required	$\{(i, v_i)\}_{1\le i\le n}$	$\{(i, v_i)\}_{1\le i\le n}$	(c, k_1, k_2)	(c, k_1, k_2)
	$2n\|p\|$bits	$2n\|p\|$bits	$3\|p\|$bits	$3\|p\|$bits
Response form and the number of bits required	(σ, μ)	(σ, μ, R)	(σ, μ, R)	(σ, μ)
	$2\|p\|$bits	$3\|p\|$bits	$3\|p\|$bits	$2\|p\|$bits
Entire Overhead	$O(n)$	$O(n)$	$O(1)$	$O(1)$

identifier $\sigma_Q = \prod_{l=1}^{Q} \sigma_l$. Finally, the algorithm verifies the following equation:

$$\prod_{l=1}^{Q} \left\{ e\left(\prod_{j=1}^{c} H\left(\left(W_{i_j} \right)_l \right)^{a_j}, v_l \right) \cdot e\left(u, \mu_l\right) \right\} = e\left(\sigma_Q, g\right) \tag{13}$$

We validate the correctness of batch audition scheme below and show that the honest CSPs will pass the audition successfully if they follow the protocol.

$$\begin{aligned} &\prod_{l=1}^{Q} \left\{ e\left(\prod_{j=1}^{c} H\left(\left(W_{i_j} \right)_l \right)^{a_j}, v_l \right) \cdot e\left(u, \mu_l\right) \right\} \\ &= \prod_{l=1}^{Q} \left\{ e\left(\prod_{j=1}^{c} H\left(W_{i_j} \right)_l^{a_j} u^{\mu_l'}, g^{x_l} \right) \cdot e\left(u^{r_l}, g^{x_l}\right) \right\} \\ &= \prod_{l=1}^{Q} \left\{ e\left(\prod_{j=1}^{c} \left(\sigma_{i_j} \right)_l^{a_j}, g \right) \cdot e\left(\left(u^{x_l}\right)^{r_l}, g \right) \right\} \\ &= \prod_{l=1}^{Q} \left\{ e\left(\prod_{j=1}^{c} \left(\sigma_{i_j} \right)_l^{a_j}, g \right) \cdot e\left(\left(w_l\right)^{r_l}, g \right) \right\} \\ &= \prod_{l=1}^{Q} e\left(\prod_{j=1}^{c} \left(\sigma_{i_j} \right)_l^{a_j} \cdot \left(w_l\right)^{r_l}, g \right) = e\left(\prod_{l=1}^{Q} \sigma_l, g \right) \\ &= e\left(\sigma_Q, g\right) \end{aligned} \tag{14}$$

7. Evaluation

In this section, we evaluate the performance of our proposed scheme in the following three aspects:

(1) Computational complexity: the cost of the setup phase, the cost of producing response proof by S, and the cost of verify response proof by an auditor.

(2) Block complexity: S needs the number of file blocks to be accessed according to the challenge.

(3) Communication complexity: the data size and bandwidth of communication between an auditor and a CSP.

Suppose that the EoCo chooses c file blocks for audition and the security parameter is k. The large prime number p should be $2k$ according to the analysis by Shacham et al. [5]. If the security level is 80 bits, then $|p| = 160$ bits. This determines the required block size to achieve the desired security level. We compare the EoCo with the scheme in [5, 8, 9] in terms of computation complexity and communication complexity in this section.

7.1. Communication Complexity. We can see from Table 3 that the number of bits required of SW-[5] is $2n|p| + 2|p|$ bits. The number of bits required of Wang-[8] is $2n|p| + 3|p|$ bits. The number of bits required of Worku-[9] is $6|p|$ bits. The EoCo' the number of bits required is $5|p|$ bits. From this, we can see that the EoCo has the smallest communication overhead, which can greatly reduce the I/O burden of cloud providers and increase bandwidth utilization.

7.2. Computation Complexity. In order to calculate the cost of computing on both sides of the auditor and cloud service provider, we detail the operations on the basic computational symbols in Table 4. We can see comparison of several schemes from Table 5 in three aspects.

(i) *Server computation overhead:* for cloud service providers, we see that the Worku-[9] is obviously less than Wang-[8]. While the EoCo is compared with the Worku-[9], the EoCo adds one more exponential operation, but reduces one hash operation. Generally speaking, one hash is more time-consuming than the exponential operation, so that the EoCo is more efficient on this side.

(ii) *Auditor computation overhead:* for the auditor, Wang-[8] and Worku-[9] are only slightly different in efficiency; the latter uses one-time exponential operation less than the former. Although our scheme uses one-time bilinear mapping more than Worku-[9], our scheme uses one-time hash operation, three-time exponential operation and two-time multiplication operation less than Worku-[9]. In this respect, the EoCo has a great promotion.

(iii) *Setup phase overhead:* because the schemes are based on BLS short signatures, the computational overhead of the setup phase is equal.

According to the theoretical analysis above, our scheme ensures the desired security properties with relatively low performance overhead.

Table 4: Notation of cryptographic operations.

Operations	Description
$Hash^t_{Z_p}$	t map-to-Z_p operations
$Hash^t_G$	t map-to-G operations
$Mult^t_G$	t multiplication in group G
$Mult^t_{Z_p}$	t multiplication in group Z_p
Exp^t_G	t exponentiations in group G
BM^t_G	t bilinear mappings among elements of G
Add^t_G	t additions on group G
$Add^t_{Z_p}$	t additions on group Z_p
$c - MultExp^t_G(l)$	t c-term exponentiations in group G,$l \leq \lvert p \rvert$

Table 5: Computation comparison.

	Our scheme	Wang-[8]	Worku-[9]
Server computation overhead	$Mult^c_G + Exp^2_G +$ $Add^c_{Z_p} + Mult^{c-1}_{Z_p} +$ $c - MultExp^1_G(a_i)$	$Mult^{c-1}_G + BM^1_G +$ $Exp^1_G + Add^c_{Z_p} + Mult^{c+1}_{Z_p} +$ $Hash^1_{Z_p} + c - MultExp^1_G(v_i)$	$Mult^{c-1}_G + Exp^1_G +$ $Add^c_{Z_p} + Mult^{c+1}_{Z_p} +$ $Hash^1_{Z_p} + c - MultExp^1_G(v_i)$
Auditor computation overhead	$Hash^c_G + BM^3_G +$ $c - MultExp^1_G(a_i)$	$Hash^c_G + Mult^2_G + Exp^3_G +$ $BM^2_G + Hash^1_{Z_p} +$ $c - MultExp^1_G(v_i)$	$Hash^c_G + Mult^2_G + Exp^2_G +$ $BM^2_G + Hash^1_{Z_p} +$ $c - MultExp^t_G(l)$
Setup phase overhead	$Mult^n_G + Hash^n_G +$ Exp^{2n}_G	$Mult^n_G + Hash^n_G +$ Exp^{2n}_G	$Mult^n_G + Hash^n_G +$ Exp^{2n}_G

8. Conclusion

In this paper, we proposed a secure and effective cloud data integrity verification scheme with privacy protection, EoCo, which is based on the BLS short signature. We presented a more practical data integrity protection model and introduced a ZK-privacy model to ensure the privacy preserving property in public audition. We theoretically analyzed our approach and prove the security of EoCo based on the CDH problem. We also demonstrated that our scheme ensures zero knowledge leakage through indistinguishable views of the attacker. We evaluated the performance of EoCo and made the comparison with related solutions. The analysis results show that our approach has relatively small communication and computational complexity.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (no. 2015BAG15B01) and the National Natural Science Foundation of China (nos. U17733115, 61402029, 61379002, and 61370190).

References

[1] G. S. Aujla, R. Chaudhary, N. Kumar, A. K. Das, and J. J. P. C. Rodrigues, "SecSVA: Secure Storage, Verification, and Auditing of Big Data in the Cloud Environment," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 78–85, 2018.

[2] L. Xiao, D. Xu, C. Xie, N. B. Mandayam, and H. V. Poor, "Cloud storage defense against advanced persistent threats: a prospect theoretic study," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 3, pp. 534–544, 2017.

[3] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 598–609, Virginia, Va, USA, November 2007.

[4] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 584–597, ACM, Alexandria, VA, USA, November 2007.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology—ASIACRYPT 2008: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 2008*, vol. 5350 of *Lecture Notes in Computer Science*, pp. 90–107, Springer, Berlin, Germany, 2008.

[6] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the IEEE INFO-COM*, pp. 525–533, March 2010.

[7] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public audibility and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[8] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[9] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Computers and Electrical Engineering*, vol. 40, no. 5, pp. 1703–1713, 2014.

[10] X. Fan, G. Yang, Y. Mu, and Y. Yu, "On indistinguishability in remote data integrity checking," *The Computer Journal*, vol. 58, no. 4, pp. 823–830, 2013.

[11] H. Liu, L. Chen, Z. Davar, and M. R. Pour, "Insecurity of an efficient privacy-preserving public auditing scheme for cloud data storage," *Journal of Universal Computer Science*, vol. 21, no. 3, pp. 473–482, 2015.

[12] *PASOS. Revista de Turismo y Patrimonio Cultural*.

[13] G. Ateniese, R. Burns, R. Curtmola et al., "Remote data checking using provable data possession," *ACM Transactions on Information and System Security*, vol. 14, no. 1, article 12, 2011.

[14] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: theory and implementation," in *Proceedings of the ACM Workshop on Cloud Computing Security (CCSW '09)*, pp. 43–53, November 2009.

[15] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology. The Journal of the International Association for Cryptologic Research*, vol. 26, no. 3, pp. 442–483, 2013.

[16] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology—ASIACRYPT 2001*, vol. 2248 of *Lecture Notes in Computer Science*, pp. 514–532, Springer, Berlin, Germany, 2001.

[17] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," *Proc. of Hotos07: Workshop on Hot Topics in Operating Systems*, p. 1, 2007.

[18] V. A. Chang, "A model to compare cloud and non-cloud storage of Big Data," *Future Generation Computer Systems*, vol. 57, pp. 56–76, 2016.

[19] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm '08)*, pp. 1–10, ACM, Istanbul, Turkey, September 2008.

[20] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS 2009)*, pp. 213–222, ACM, Chicago, Ill, USA, November 2009.

[21] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, "Dynamic remote data auditing for securing big data storage in cloud computing," *Information Sciences*, 2015.

[22] L. Xin, X. Sun, Z. Fu, L.-A. Zhang, and J. Xi, "Effective and secure access control for multi-authority cloud storage systems," *International Journal of Security and Its Applications*, vol. 10, no. 2, pp. 217–236, 2016.

[23] L. Yu, H. Shen, K. Sapra, L. Ye, and Z. Cai, "CoRE: Cooperative End-to-End Traffic Redundancy Elimination for Reducing Cloud Bandwidth Cost," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 2, pp. 446–461, 2017.

[24] L. Yu, H. Shen, Z. Cai, L. Liu, and C. Pu, "Towards Bandwidth Guarantee for Virtual Clusters under Demand Uncertainty in Multi-Tenant Clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, no. 2, pp. 450–465, 2018.

[25] V. Attasena, J. Darmont, and N. Harbi, "Secret sharing for cloud data security: a survey," *The VLDB Journal*, vol. 26, no. 5, pp. 657–681, 2017.