Hindawi
Security and Communication Networks
Volume 2017, Article ID 2948025, 13 pages
https://doi.org/10.1155/2017/2948025

WILEY | Hindawi

Research Article

Co-Check: Collaborative Outsourced Data Auditing in Multicloud Environment

Jian Mao,[1] Wenqian Tian,[1] Yan Zhang,[1] Jian Cui,[1] Hanjun Ma,[1] Jingdong Bian,[1] Jianwei Liu,[1] and Jianhong Zhang[2,3]

[1]*School of Electronic and Information Engineering, Beihang University, Beijing, China*
[2]*School of Science, North China University of Technology, Beijing, China*
[3]*Guangxi Key Lab of Multi-Source Information Mining & Security, Guilin, China*

Correspondence should be addressed to Jianhong Zhang; jhzhang@ncut.edu.cn

Received 1 April 2017; Accepted 1 October 2017; Published 31 October 2017

Academic Editor: Qing Yang

With the increasing demand for ubiquitous connectivity, wireless technology has significantly improved our daily lives. Meanwhile, together with cloud-computing technology (e.g., cloud storage services and big data processing), new wireless networking technology becomes the foundation infrastructure of emerging communication networks. Particularly, cloud storage has been widely used in services, such as data outsourcing and resource sharing, among the heterogeneous wireless environments because of its convenience, low cost, and flexibility. However, users/clients lose the physical control of their data after outsourcing. Consequently, ensuring the integrity of the outsourced data becomes an important security requirement of cloud storage applications. In this paper, we present Co-Check, a collaborative multicloud data integrity audition scheme, which is based on BLS (Boneh-Lynn-Shacham) signature and homomorphic tags. According to the proposed scheme, clients can audit their outsourced data in a one-round challenge-response interaction with low performance overhead. Our scheme also supports dynamic data maintenance. The theoretical analysis and experiment results illustrate that our scheme is provably secure and efficient.

1. Introduction

With the increasing demand for ubiquitous connectivity, wireless technology has significantly improved our daily lives. Meanwhile, together with cloud-computing technology (e.g., cloud storage services and big data processing), heterogeneous wireless networking technology has become a foundation infrastructure widely adopted by emerging communication networks, for instance, IoT (Internet of Things), C-RAN (cloud radio access network), and body-area network, as shown in Figure 1. Particularly, the cloud storage technique has been widely used in services, such as wireless data outsourcing and resource sharing, thanks to its convenience, low cost, and flexibility. Nowadays, online service providers, such as Amazon and Baidu, operate large data centers and offer unlimited storage capacity for users, relieving their burden of local data management and maintenance [1, 2]. In addition, cloud storage enables universal data access in any place. However, users lose the physical control of their outsourced data, while the cloud storage service provider is not always trustworthy. Dishonest service providers may conceal the fact that users' data have been damaged due to some misoperations or unexpected accidents. Even worse, malicious service providers also may delete the data seldom accessed by users to gain more benefits. How to ensure the integrity of their remotely outsourced data becomes a serious concern for users selecting cloud storing services.

Traditional data integrity verification solutions [3, 4], which are based on hash functions and digital signatures, are impractical to audit cloud data remotely due to their unacceptable communication and computational overhead to retrieve the outsourced files. To check the remote data integrity effectively without retrieving the whole outsourced document, Ateniense et al. presented the first probabilistic verification model called *provable data possession* (PDP) based on homomorphic cryptography algorithm and *sampling* techniques [5]. Taking the public verifiability into

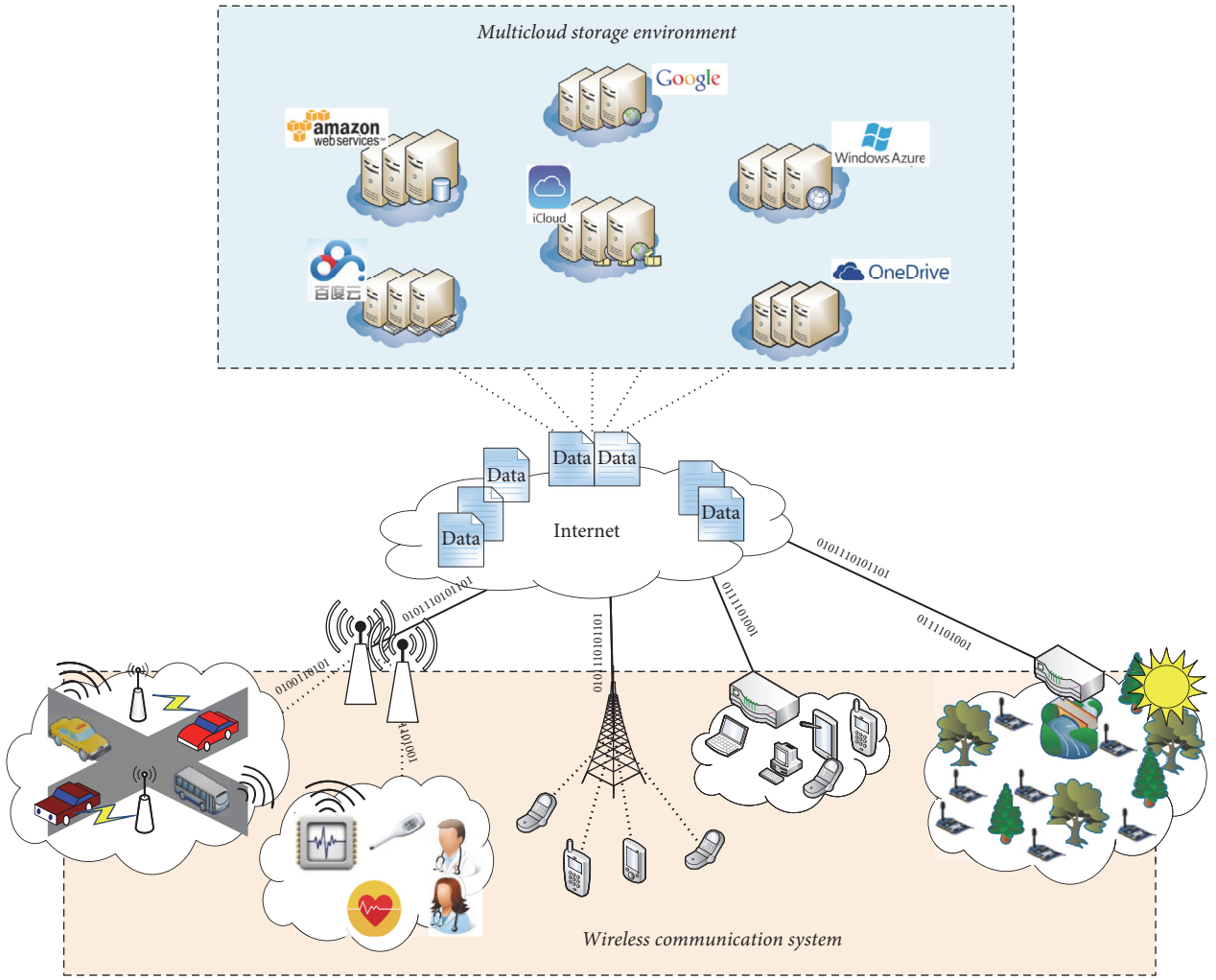

Figure 1: Overall architecture of the multicloud based heterogeneous wireless network.

account, Ateniense et al. improved their approach [6]; Wang et al. also proposed a publicly verifiable cloud data audition scheme that supports dynamic data maintenance by using Merkle Hash Tree data structure [7]. Juels et al. introduced error correcting coding techniques and proposed *Proof of Retrievability* (POR) mechanisms to audit cloud data and ensure data correction if data corruption happened.

Most of these previous works mainly target the problem of data integrity audition in a single-cloud storage environment rather than a heterogeneous cloud infrastructure that collaborates multiple internal (private) and/or external (public) cloud resources [8, 9]. In the multicloud environment, users split their data, duplicate file blocks, and outsource them to different CSP (Cloud Service Provider) servers. The solutions above cannot enforce the data integrity checking efficiently in such an environment where data spread over multiple servers. Aiming at this problem, Zhu et al. propose a cooperative provable data possession (CPDP) scheme [8, 10] in the multicloud environment. However, in the CPDP scheme, the security parameter π' is independent of other parameters; and thus servers can bypass the authentication by forging the parameter π' in the response sequence. Moreover, in the process of third-party public verification, the third party needs to know where every data block is exactly stored. It poses a threat to users' data storage privacy and increases the operation overhead for the third auditing party to maintain the storing state of file blocks. Moreover, besides the effectiveness, efficiency is also a significant concern for a data integrity auditing solution in the multicloud storage environment.

In this paper, we present *Co-Check*, a collaborative multicloud data integrity audition scheme, which is based on BLS signature and homomorphic tags. According to proposed scheme, users can audit their outsourced data in one challenge-response interaction with low communication cost. Our scheme also enables public verification and supports dynamic data maintenance that users can modify and delete the data with low performance overhead. The contributions made by this paper are summarized as follows.

(i) We propose an effective collaborative multicloud data audition scheme enabling users to conduct data integrity checking among multiple CPS server

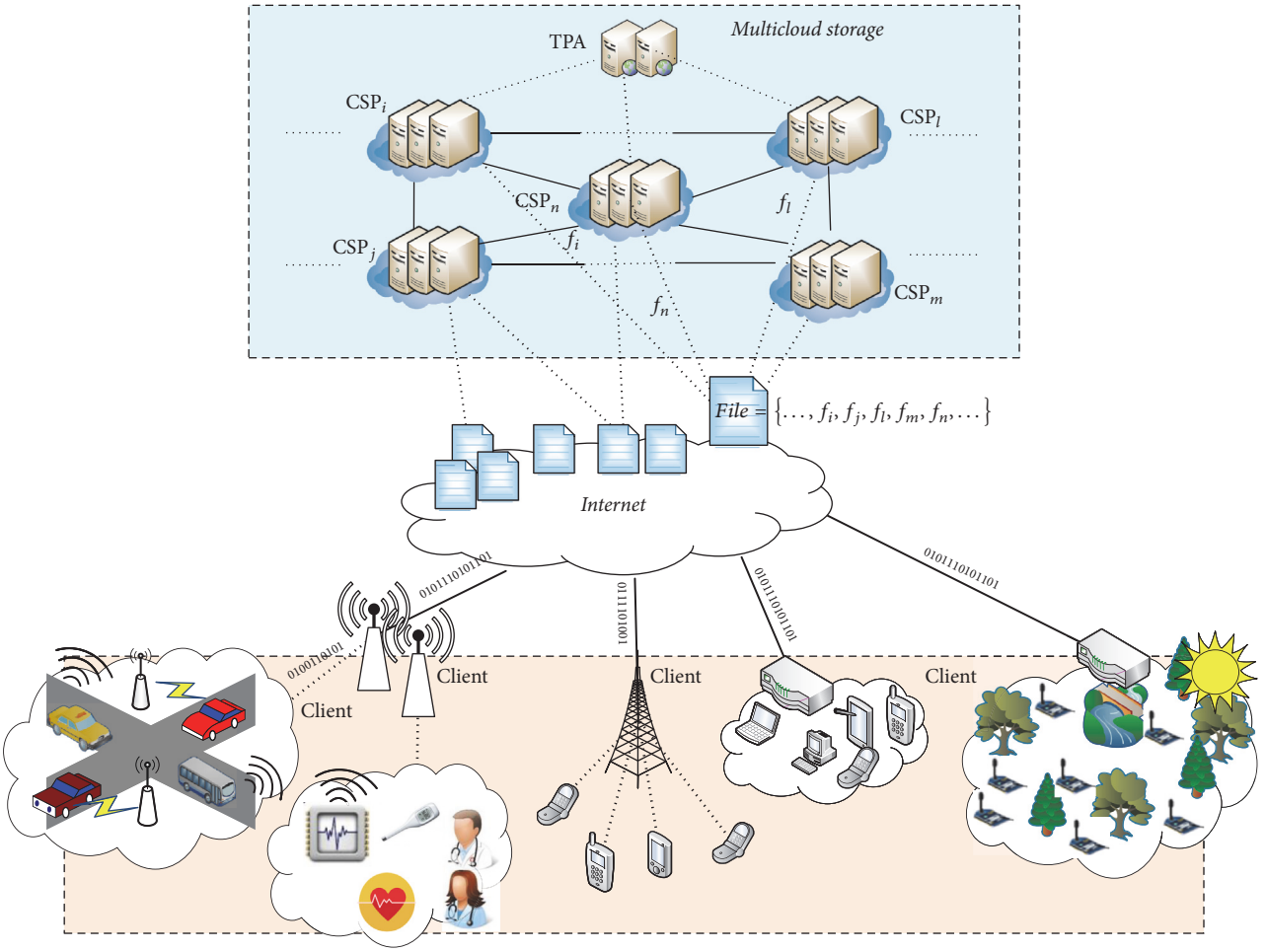

Figure 2: Multicloud based data storing for wireless communication systems.

simultaneously in one-round challenge-response procedure.

(ii) The audition procedure of our scheme is stateless and supports unlimited challenge-response interactions. Moreover, the proposed scheme supports dynamic data maintenance efficiently.

(iii) We prototype our scheme and conduct system evaluation. The theoretical analysis and experiment results illustrate that our scheme is provably secure and efficient.

Paper Organization. The rest of this paper is organized as follows. Section 2 describes the security goals, system model, and the overall architecture of our approach; Section 3 presents the collaborative multicloud data integrity audition scheme; in Section 4, we make the theoretical analysis and evaluate our protocol on security and performance aspects; Section 5 discusses the related work; and Section 6 concludes the paper.

2. Approach Overview

2.1. System Framework. As shown in Figure 2, the general multicloud storage system includes three types of network entities.

(i) Client (or User). (We use the term *user* and *client* exchangeably in this paper.) Clients outsource data to reduce local storage overhead and make use of the computation resources provided by the cloud service providers in multicloud storage system.

(ii) Cloud Service Provider (CSP). CSPs that possess a large quantity of hardware and software resources are clustered to provide remote data storing services. We assume that there is an organizer in the CSP cluster, a mediation node that interacts with users and other CSPs.

(iii) Third-Party Authority (TPA). TPA is an optional entity being partially trusted in the multicloud scenario.

In the multicloud storage system shown in Figure 2, the user splits her/his documents into several file blocks. The

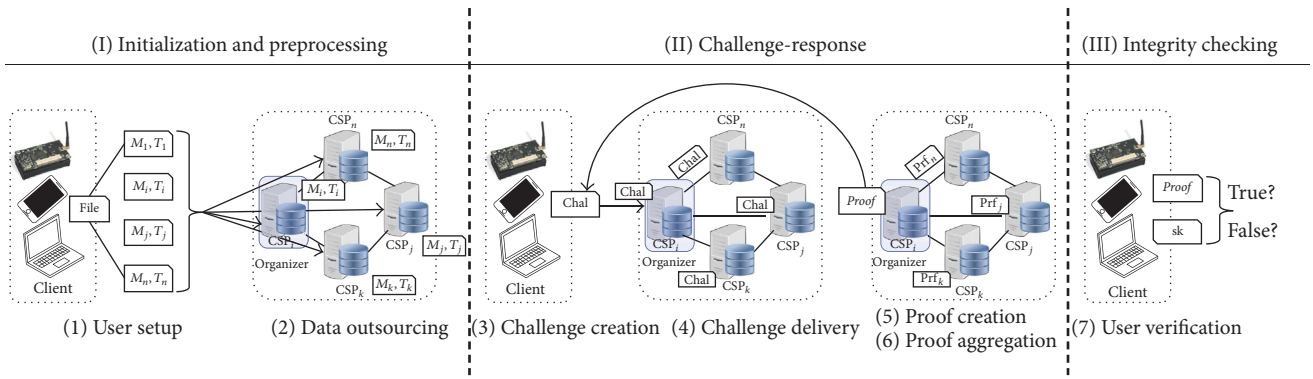

Figure 3: Collaborative data integrity audition procedure.

file blocks will distribute the cloud storage servers deployed by different cloud service providers. In addition, to promote the access efficiency and ensure the data retrievability, users might also duplicate the file blocks and spread the copies to several cloud servers.

2.2. Challenges and Goals. As the CSPs in the multicloud system cannot always be trustworthy, it is necessary for users to establish the integrity audition mechanism that ensures their outsourced data are stored correctly without unauthorized access by CSP servers or other entities. To make the audition more efficient, another challenge of data integrity audition in the multicloud environment is to conduct parallel checking, which means verifying the integrity of block files stored in different CSP servers simultaneously. Moreover, supporting securely dynamic maintenance is also a major concern of the multicloud data audition.

Aiming to address the above challenges, the goal of this paper is to propose an effective multicloud data integrity audition mechanism satisfying the following requirements.

(i) Correctness: benign servers will prove themselves successfully and none of the misbehaved servers can bypass the checking.

(ii) Batch verification: the client can simultaneously verify the integrity of the file blocks distributed in different CSP servers without retrieving the file.

(iii) Stateless and unbounded checking: the audition procedure is stateless and supports unlimited challenge-response interactions.

2.3. Collaborative Data Integrity Audition Model. Our collaborative data audition model consists of three stages as we defined in our preliminary version [11]: *initialization*, *challenge-response*, and *integrity checking*. Motivated by the sampling technique introduced by Ateniese et al. [5], users split their files and distribute the file blocks among the cloud service providers (CSPs) in *initialization and preprocessing* stage. Meanwhile, users keep the corresponding metadata for the future audition. Here we use BLS signature to create the *homomorphic tags* due to its homomorphic property. Instead of retrieving the whole file to verify its correctness, in stages II and III, users generate the challenges for audition by using parts of the metadata restored at the client side to prompt the audition efficiency and ensure that malicious CSPs cannot bypass the check with a high confidence rate. Additionally, our scheme also designates a subprocedure to support dynamic maintenance. The procedure of our scheme is shown in Figure 3.

(1) Stage I: Initialization and Preprocessing. Stage I consists of steps (1)-(2) in Figure 3. In step (1), the user selects system parameters and generates keys for BLS algorithm used in the successive steps. Meanwhile, the user splits the file F into file block set and each file block m_{ij} consists of several file sectors. Then the user computes the homomorphic tags σ_{ij} corresponding to the file sectors. After preprocessing the outsourced file, the user distributes the file blocks with the metadata for audition into the cloud servers belonging to the different CSPs and keeps the secret parameter locally.

(2) Stage II: Challenge-Response. Stage II includes steps (3)–(6) in Figure 3. When the user wants to audit her/his outsourced file, she/he computes a challenge sequence corresponding to the file blocks under test. The user sends *organizer* to the challenge sequence and *organizer* will forward the challenges to the aimed CSP servers that contain the user's file blocks. CSP servers calculate and return their proofs to *organizer*. *Organizer* aggregates the proof received and sends the corresponding answer to the user.

(3) Stage III: Integrity Checking. Based on the received response from *organizer*, the user verifies the data integrity in step (7) shown in Figure 3. If data are stored correctly, the algorithm outputs "*TRUE*"; otherwise, it outputs "*FALSE,*" which means that there exist misbehaved CSP servers.

Dynamic Maintenance. When users need to conduct dynamic operations on their outsourced data, they recreate tags corresponding to the new file sectors and send them to the organizer for updating.

All the symbols used in this paper are listed in Notation.

3. Collaborative Multicloud Data Integrity Audition Scheme

In this section, we present our collaborative multicloud data integrity audition scheme in detail. The notations and concepts employed in our work are illustrated below.

(i) $\pi = (p, G, G_T, e, g)$ is the system parameter. p is a big prime number and is the order of the cyclic group G; $e : G \times G \to G_T$ is a nondegenerate bilinear map. g is the generator of G.

(ii) l is the number of the CSPs, and the CSP set is represented as $\{\mathrm{CSP}_1, \mathrm{CSP}_2, \ldots, \mathrm{CSP}_l\}$.

(iii) F is user's file and F_n is the file name. The file F is separated into n blocks, each of which contains s sectors, $F = \{m_{ij}\}_{n\times s}$, where $m_{ij} \in Z_p$.

(iv) Q is the challenge generated by users.

(v) $H : \{0,1\}^* \to G$ is a hash function.

As shown in Figure 3, our scheme includes three entities, a *user*, CSP servers, and an *organizer*, which is also one of the CSP servers. The integrity checking scheme is fulfilled by the following eight steps.

Step 1 (user setup).

(1) KeyGen: $KeyGen(1^\lambda) \to \{\mathrm{sk}, \mathrm{pk}\}$.

The user selects secure parameter λ and system parameters π and H. She/he randomly selects an $\alpha \in Z_p^*$ as the private key. The public key is $v \leftarrow g^\alpha \in G$. Then the user gets $\mathrm{pk} = \{v, g\}$, $\mathrm{sk} = \{\alpha\}$.

(2) File preprocessing: $F \to \{m_{ij}\}_{n\times s}$.

The user splits the file F into n blocks, each of which contains s parts. The file F is represented as follows:

$$F = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_n \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1s} \\ m_{21} & m_{22} & \cdots & m_{2s} \\ \vdots & \vdots & \cdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{ns} \end{bmatrix}, \quad m_{ij} \in Z_p. \tag{1}$$

We assume that num_i $(i = 1, \ldots, n)$ is the total number of copies corresponding to each data block m_i $(i = 1, \ldots, n)$ stored in different CSPs, and V_i $(i = 1, \ldots, n)$ represents how many times each data is updated. The initial value of V_i $(i = 1, \ldots, n)$ is 0 for all the elements. We use $\chi_i = i \parallel \mathrm{num}_i \parallel V_i$ $(i = 1, \ldots, n)$ to represent it. $\parallel$ represents concatenation.

(3) TagGen: $TagGen(\mathrm{sk}, \mathrm{pk}, F) \to \{\sigma\}$.

The user randomly selects s parameters $u_1, \ldots, u_s \in G$ and computes the tags $\sigma_{ik} \leftarrow (H(F_n \parallel i \parallel k \parallel V_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^\alpha$ for $k = 1, \ldots, \mathrm{num}_i$ corresponding to each data block m_i $(i = 1, \ldots, n)$ and thus the set of all tags is obtained. As shown in Figure 4, m_i $(i = 1, \ldots, n)$ represents data blocks from the file; each block is separated into s parts and every part of a block is represented as m_{ij} $(i = 1, \ldots, n; j = 1, \ldots, s)$; σ_{ik} $(i = 1, \ldots, n; k = 1, \ldots, \mathrm{num}_i)$ represent num_i tags corresponding to m_i.

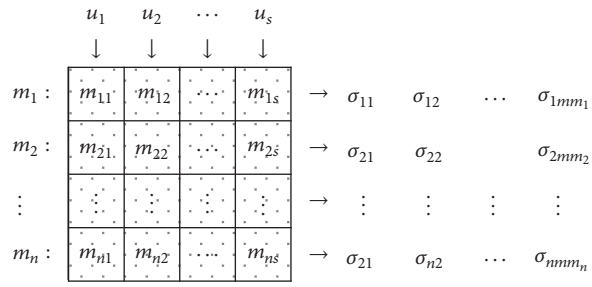

Figure 4: Tag generating.

Step 2 (data outsourcing). The user sends the file F and corresponding tags to the organizer, and the organizer distributes data blocks with corresponding tags to different CSP servers (as shown in Figure 5). If a file block is stored with several copies, every copy of the file block has a tag. For instance, data block m_i $(i = 1, \ldots, n)$ is stored with num_i copies, then there are num_i tags, which means the CSPs should store data m_i along with the tag σ_{ik} $(k \in [1, \ldots, \mathrm{num}_i])$ from the num_i labels. The user computes the public parameter $\psi = (u, \chi)$ $(u = \{u_1, \ldots, u_s\}, \chi = \{\chi_1, \ldots, \chi_n\})$ and sends it to the trusted third party for storage. The user keeps the private key at the client side.

Step 3 (challenge creation, challenge (chal)). When the user wants to audit the outsourced data, he or she computes a challenge, $\mathrm{chal} = Q = \{(i_j, a_j) \mid i_j \in [1, \ldots, n], a_j \in Z_p^*, j = 1, \ldots, c\}$, and sends it to the organizer.

Step 4 (challenge delivery, forward (chal)). The organizer forwards the received challenge $\mathrm{chal} = Q$ to the CSP servers, $\mathrm{CSP}_{k\in[1,\ldots,l]}$. Without losing generality, we assume there are t CSP servers that store the blocks challenged by the user.

Step 5 (proof creation and delivery, $\mathrm{GenProof}(\mathrm{pk}, Q, m_i, \sigma_{ix}) \to \{P_k\}$). $\forall k \in [1, \ldots, l]$, the service provider CSP_k computes the evidence according to the following formula:

$$\begin{aligned} p_{1kj} &= \sum_{m_i \in \mathrm{CSP}_k} a_i m_{ij} \mod p \quad (j = 1, \ldots, s) \\ p_{2k} &\longleftarrow \prod_{m_i \in \mathrm{CSP}_k} \sigma_{ix}^{a_i} \in G. \end{aligned} \tag{2}$$

CSP_k returns the proofs shown in (3) to the organizer:

$$\begin{aligned} P_k &= \{p_{1k}, p_{2k}\} \\ p_{1k} &= \{p_{1k1}, p_{1k2}, \ldots, p_{1ks}\}. \end{aligned} \tag{3}$$

Step 6 (proof aggregation and response, $\mathrm{Aggregation}(\mathrm{pk}, Q, P_1, P_2, \ldots, P_t) \to \{P\}$). The organizer computes $P_{1j} =$

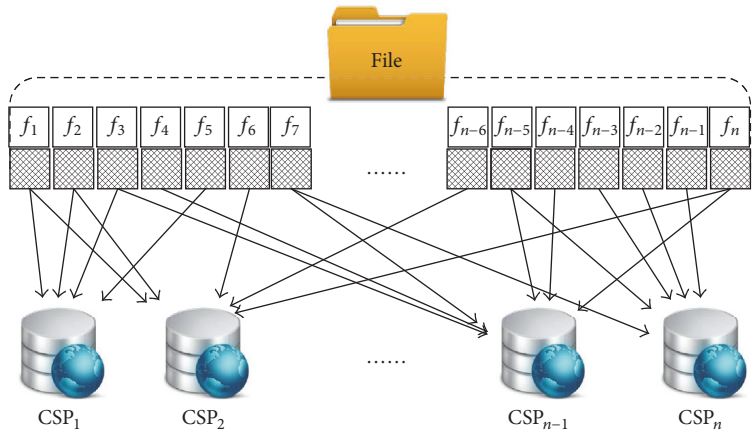

Figure 5: Data outsourcing.

$\sum_{k=1}^{t} P_{1kj} \bmod p$ $(j = 1, 2, \ldots, s)$, $P_2 \leftarrow \prod_{k=1}^{t} P_{2k} \in G$. The organizer returns the aggregated proof P, $P = \{P_1, P_2\}$, to the user, where $P_1 = \{P_{11}, P_{12}, \ldots, P_{1s}\}$.

Step 7 (user verification). After the user received the data $P = \{P_1, P_2\}$ sent by the organizer, she/he gets the parameter $\psi = (u, \chi)$ from the trusted third party and verifies the response according to the formula

$$e(P_2, g) = e\left(\prod_{(i,a_i)\in Q} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{a_i} \prod_{j=1}^{s} u_j^{p_{1j}}, v\right). \quad (4)$$

If formula (4) holds, it means the outsourced data are stored correctly and the output is "*TRUE*"; otherwise, the output is "*FALSE*."

We summarize the interactions of collaborative auditing in Figure 6.

Dynamic Update. When users need to update data $m_i \rightarrow m'$, they should make a modification $V_i = V_i + 1$ from $\chi_i = i \parallel \text{num}_i \parallel V_i$, compute the new label $new\_\sigma_{ik} \leftarrow (H(F_n \parallel i \parallel k \parallel V_i) \cdot \prod_{j=1}^{s} u_j^{m_j'})^{\alpha}$, $k = 1, \ldots, \text{num}_i$, and send the updated χ_i along with the corresponding label σ_{ik} $(k = 1, \ldots, \text{num}_i)$ to the organizer. After that, the organizer conducts the distributed storing operation. Due to the relevance between the label and the sequence of the data, the scheme could only realize part of the update operations, namely, data modification and deletion.

4. Evaluation

4.1. Security Analysis. In this section, we prove two properties to ensure data integrity under our scheme.

Theorem 1. *Correctness. If all CSP servers keep user's data correctly, they can successfully pass the challenge-response verification procedure initiated by the user.*

Proof. To verify the data correctness, according to step (7), the use computes $e(P_2, g)$. It can be noticed in step (5)-(6) that $P_{1j} = \sum_{k=1}^{t} P_{1kj} \bmod p$ $(j = 1, 2, \ldots, s)$ and $P_2 = \prod_{k=1}^{t} P_{2k}$, where $p_{1kj} = \sum_{m_i \in \text{CSP}_k} a_i m_{ij} \bmod p$ $(j = 1, \ldots, s)$ and $p_{2k} = \prod_{m_i \in \text{CSP}_k} \sigma_{ix}^{a_i} \in G$.

According to the bilinear property of the Weil-paring function, we get

$$\begin{aligned}
e(P_2, g) &= e\left(\prod_{k=1}^{t} P_{2k}, g\right) = e\left(\prod_{k=1}^{t}\left(\prod_{m_i \in \text{CSP}_k} \sigma_{ix}^{a_i}\right), g\right) \\
&= e\left(\prod_{k=1}^{t}\left(\prod_{m_i \in CSP_k}\left(H(F_n \parallel i \parallel k \parallel V_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}}\right)\right)^{a_i \alpha}, g\right) \\
&= e\left(\prod_{k=1}^{t}\left(\prod_{m_i \in \text{CSP}_k} H(F_n \parallel i \parallel k \parallel V_i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}}\right)^{a_i}, g^{\alpha}\right) \\
&= e\left(\prod_{(i,a_i)\in Q} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{a_i} \cdot \prod_{j=1}^{s} u_j^{\sum_{k=1}^{t} \sum_{m_i \in \text{CSP}_k} a_i m_{ij}}, v\right) \\
&= e\left(\prod_{(i,a_i)\in Q} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{a_i} \prod_{j=1}^{s} u_j^{p_{1j}}, v\right).
\end{aligned} \quad (5)$$

This completes our proof. □

Theorem 2. *If there exists a probabilistic polynomial time adversary adv and it is able to successfully convince the TPA to accept the fake proof information for a corrupted file in nonnegligible probability, then it is possible to construct a polynomial algorithm B to solve the computational Diffie-Hellman (CDH) problem by invoking adv with nonnegligible probability.*

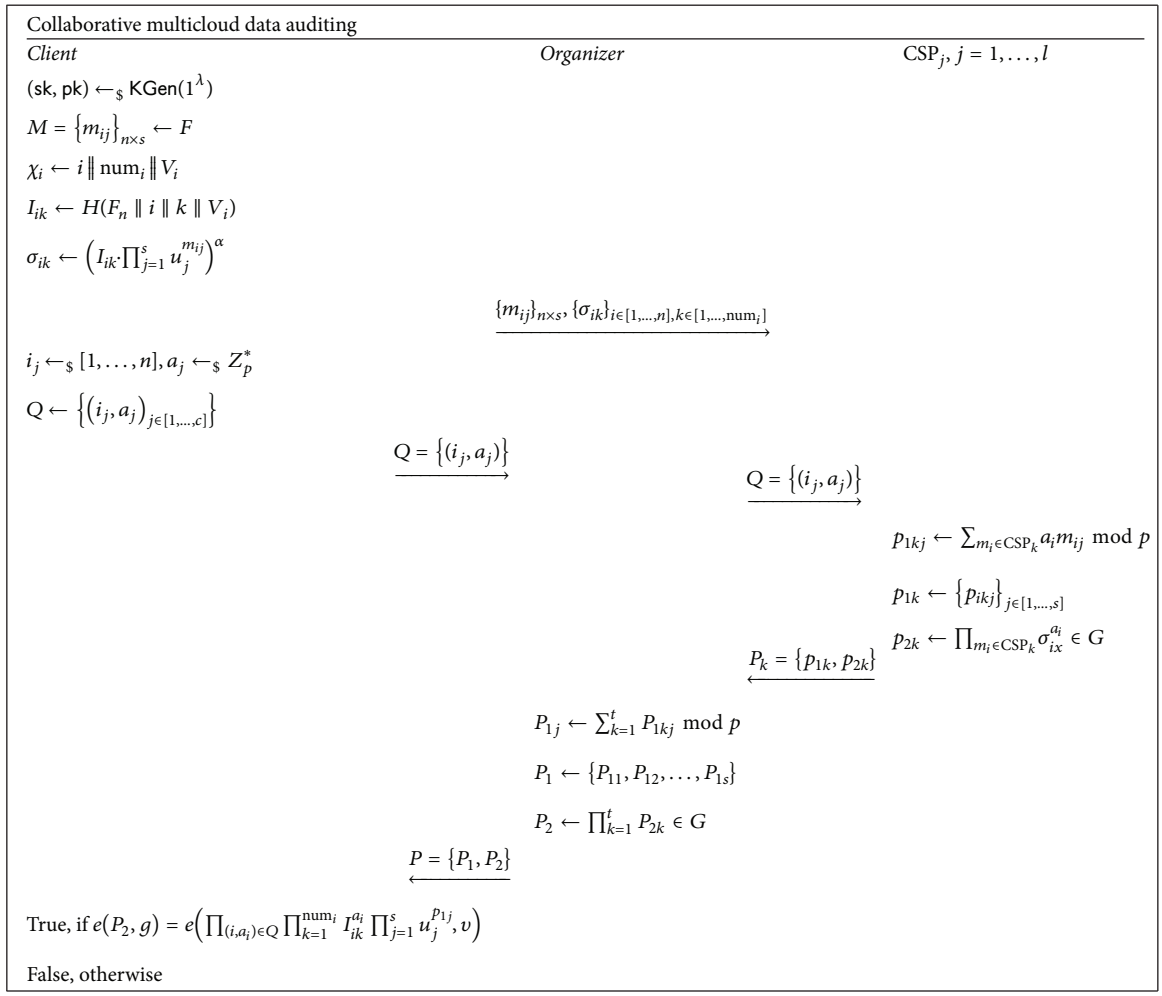

Figure 6: Collaborative multicloud data integrity audition scheme.

Proof. Suppose that the algorithm B is given an instance of the CDH problem tuple shown as follows:

$$\left(g, g^a, g^b\right) \tag{6}$$

and its goal is to compute g^{ab}. The algorithm B will execute an interactive game with adv in the following game of security model.

Setup. Let $v = g^a$ be the public key of the user, and choose a hash function $H : \{0,1\}^* \rightarrow G$ which acts as random oracle in the following security proof. And for $i = 1$ to s, it randomly selects r_i to set $u_i = g^{r_i}$. Finally, it returns the public parameter $\mathrm{params} = \{p, G, g, u_1, \ldots, u_s, H\}$ to the adversary adv.

Hash Query. At any time, the adversary adv is able to adaptively query hash oracle for the string $F_n \| i \| k \| V_i$ it submits. And to respond to these queries, the algorithm maintains an H-list which is initially empty and responds as follows:

(1) If $(F_n \| i^* \| k \| V_i, *, *)$ exists in the H-list, B retrieves the tuple $(F_n \| i^* \| k \| V_i, d_i, h_i)$ and sends h_i the adversary adv.

(2) Otherwise, B chooses a bit $d_i \in \{0,1\}$ according to a bivariate distribution function $\Pr[d_i = 0] = \tau$ and $\Pr[d_i = 1] = 1 - \tau$, where τ is a fixed probability value which will be determined later. Then B answers as follows:

   (a) If $d_i = 0$, B chooses a random number t_i to compute $h_i = g^{t_i}$ and return h_i to the adversary. Then B inserts the tuple $(F_n \| i^* \| k \| V_i, d_i, h_i)$ to the H-list.

   (b) If $d_i = 1$, B chooses a random number t_i to compute $h_i = (g^b)^{t_i}$ and return h_i to the adversary. Then B inserts the tuple $(F_n \| i^* \| k \| V_i, d_i, h_i)$ to the H-list.

TagGen Oracle. At any time, the adversary can adaptively query the TagGen oracle with message m. To respond to it, B executes as follows:

(1) First, it divides message into $m = m_1 \parallel \cdots \parallel m_n$.

(2) Then, it checks whether the tuple $(F_n \parallel i \parallel k \parallel V_i, d_i, h_i)$ exists in the H-list:

(a) if there exists $d_i = 1$ for $i = 1$ to n, it aborts.

(b) if $d_i = 0$ for $i = 1$ to n, it computes

$$\begin{aligned}\delta_i &= (g^a)^{t_i} \prod_{j=1}^{s} (g^a)^{r_i} \\ &= \left(H(F_n \parallel i \parallel k \parallel V_i) \prod_{j=1}^{s} u_i^{m_{ij}} \right)^a.\end{aligned} \tag{7}$$

(3) Otherwise, it makes a Hash Query with $(F_n \parallel i \parallel k \parallel V_i, d_i, h_i)$ and executes as above.

Challenge. The adversary adv chooses a subset $I \subseteq \{1, \ldots, n\}$ of indices of the data blocks such that at least one index in set I satisfies $d_i = 1$ in the tuple $(F_n \parallel i \parallel k \parallel V_i, d_i, h_i)$. And $\forall i \in I$ one has queried the hash oracle before.

The challenge sets the challenge information chal $= \{(i, s_i) \mid i \in I\}$ and sends it to the adversary. □

Proof. Finally, the adversary outputs the response as (P_1^*, P_2^*).

If the adversary wins the above game, the returned proof information (P_1^*, P_2^*) can pass the verification, which means that (P_1^*, P_2^*) should satisfy

$$\begin{aligned}&e(P_2^*, g) \\ &= e\left(\prod_{(i,s_i)\in I} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{s_i} \prod_{j=1}^{s} u_j^{p_{1j^*}}, v \right),\end{aligned} \tag{8}$$

where $p_{1j} = \sum_{k=1}^{t} p_{1kj} \bmod p$ for $j = 1$ to s.

Without loss of generality, we assume that there is index $i^* \in I$ whose $d_i = 0$ in the H-list and the other index $j \in I/\{i^*\}$ satisfies $d_j = 0$. For simplicity, we assume all $d_{i^*j} = 1$ of $H(F_n \parallel i^* \parallel k \parallel V_j)$.

Thus, we have

$$\begin{aligned}&e(P_2^*, g) \\ &= e\left(\prod_{(i,s_i)\in I} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{s_i} \prod_{j=1}^{s} u_j^{p_{1j^*}}, v \right) \\ &= e\left(\prod_{(i,s_i)\in I} \prod_{k=1}^{\text{num}_i} H(F_n \parallel i \parallel k \parallel V_i)^{s_i} \prod_{j=1}^{s} u_j^{p_{1j^*}}, v \right) \\ &= e\left(\left(g^b\right)^{\sum_{k=1}^{\text{sum}_{i^*}} s_{i^*} t_{i^*k}} \prod_{(i,s_i)\in I/\{i^*\}} \right. \\ &\left. \cdot \prod_{k=1}^{\text{num}_i} \left(g^{t_{ik}}\right)^{s_i} \prod_{j=1}^{s} u_j^{p_{1j^*}}, v \right).\end{aligned} \tag{9}$$

It means that

$$g^{ab} = \frac{P_2^*}{\prod_{k=1}^{\text{num}_i} (g)^{a t_{ik} s_i} \prod_{(i,s_i)\in I/\{i^*\}} \prod_{j=1}^{s} g^{r_j p_{1j^*}}}. \tag{10}$$

It means that the solution of the CDH problem can be solved.

From the above simulation, we know whether B could output the correct solution of CDH problem depends on whether the simulation aborts during the TagGen Query and Challenge phases and whether the adversary could output a valid proof information for the challenge information. The adversary is allowed to make the Hash Query at most q times. Nonabort probability during TagGen Query phase requires that all $d_j = 0$ for $j = 1$ to s; thus its probability is $(\tau)^s$. Nonabort probability during Challenge phase requires that at least one index i^*'s $d_{i^*} = 1$; thus, its probability is at least $(1 - \tau)\tau^{l-1}$, where l is the size of subset I. Thus, its success probability is

$$\epsilon > (1 - \tau)\, \tau^{l-1}\, (\tau)^s. \tag{11}$$

When $\tau = 1/q$, then

$$\epsilon > \left(1 - \frac{1}{q}\right) \left(\frac{1}{q}\right)^{s+l-1}. \tag{12}$$

This completes our proof. □

4.2. Performance Analysis. We prototyped our algorithm and the evaluation is conducted on a desktop with Intel Core 2 Duo CPU @2.66 GHz, running Ubuntu 10.10 in Oracle VM VirtualBox Version 4.2.10 configured with 2 GB memory, and adopted PBC library to implement the crypto primitives. The security parameter of the bilinear pairing function is configured as 80, which means the prime number p is 160 bits. In the evaluation, we set the file size as 80 KB, 160 KB, and 320 KB, respectively. The result of evaluation is illustrated in Table 1.

The experiment results shown in Table 1 illustrate that the time cost of preprocessing and challenge generating will not be influenced by the number of file blocks. The time cost of proof generating decreases with the decline of N, the number of file blocks; in contrast the time cost of verification will increase when N decreases. The time cost of preprocessing increases proportionally with the increase of file size. When file size increases, the challenge generation time cost almost remains unchanged and the time cost of proof generating and verification increases.

5. Related Work

Based on different properties of the proposed models or schemes, related work can be classified as static data verification schemes, integrity verification schemes supporting dynamic operation on data, and verification schemes in multicloud environments. In this section, we discuss the related work in detail.

Table 1: Evaluation results of our approach.

Size (bits)	Blocks (#)	Sectors (#)	Challenge blocks (#)	T_{Pre} (s)	T_{Gch} (us)	T_{Gpr} (ms)	T_{Ver} (ms)
80	64	64	32	24.223	20	3121	1063
80	32	128	16	23.326	15	154	1223
80	16	256	8	25.352	18	81	1414
320	128	128	64	84.337	29	927	1671
320	64	256	32	81.216	19	512	2085
320	32	512	16	84.164	16	305	2847
1280	256	256	128	330	44	4950	3289
1280	128	512	64	336	119	9110	6761

T_{Pre} is the time cost for preprocessing; T_{Gch} is the time cost for generating challenge; T_{Gpr} is the time cost for generating proof; T_{Ver} is the time cost for verification.

5.1. Static Data Integrity Verification. Early research of outsourced verification focuses on static archive data. Deswarte et al. [3] are the first to propose remote data integrity verification. They proposed two solutions to this problem, one is to precompute hash value of files and compare whether the hash value returned by server is equivalent to that of the local storage; this solution could significantly reduce the communication bandwidth between users and the server to $O(1)$. Another solution, which is based on RSA signature, requires users to sign the data before it is outsourced with the labels at the server side. Challenges could be issued randomly in the process of verification, and the bandwidth is $O(1)$. The computational cost of the server is $O(n)$ (n is the number of the file blocks), which increases linearly with respect to the file size.

Gazzoni Filho and Barreto [4] proposed a remote data integrity verification scheme by combining RSA signature and hash function techniques. Their method could verify the same file for unlimited times, but the whole package of data is required to conduct a specific verification.

Sebe et al. [12] proposed a new integrity verification scheme based on Diffie-Hellman key exchange. In their scheme, the computational overhead at the user and server sides is $O(1)$, while the storage cost at the user side, $O(n)$, increases linearly with respect to the entire data size. Their follow-up work [13] combines Diffie-Hellman key exchange and RSA signature to realize remote data integrity verification.

To reduce computational overhead, Ateniese et al. [5, 14] propose a probabilistic remote data integrity checking scheme called *provable data possession* (PDP) by using homomorphic verification tags and *sampling* technique.

Ateniese et al. [6] proposed a framework to adopt homomorphism identification protocol in data integrity verification and they demonstrated this under the instance of homomorphism identification authentication protocol by Shoup [15]. The authors define the model of homomorphism identification authentication, the model of data integrity verification, and the corresponding attack models.

The schemes above can only detect whether data is properly stored but could not correct the mistakes (like retrieving the data). Another branch of remote data integrity checking focuses on the error correction and retrievability along with the cloud data audition.

Therefore, the study emphasis lies on data error correction and retrieval along with data integrity verification.

Juels and Kaliski Jr. [16] combined data possession checking and error correction of coding technique and became the first to propose the model of POR (proof of retrievability) for remote storage of data. This model adds indistinguishable sentinel to the original code which is not only able to preserve data integrity, and data availability is also realized. Their scheme is used to handle encrypt data.

Shacham and Waters [17] proposed two types of POR schemes: one is a public authentication scheme based on BLS signature, the other is private authentication on the basis of pseudorandom function, and both of the schemes have low interactions and computations. Bowers et al. [18, 19] introduced POR scheme in distributed static data storage system and realized and practiced it.

Naor and Rothblum [20] study the issue of whether files are damaged badly when they are stored in remoter server. They firstly focus on the entire file correcting error code, then compute message authentication code (MAC) for every data block to verify its integrity. When the integrity is damaged and it is within the range of correcting error, then the error detection and correction are to be realized.

Xu and Chang [21] proposed a high efficiency POR scheme, in which data block is involved with s group elements and L child data blocks, the storage overhead is $1/S$ of the file block, and computational costs is $O(s)$.

In addition, for the static data in cloud, multiple integrity verification schemes have been proposed which support public verification and users' privacy preservation. In the cloud storage users worry that data in the cloud server is damaged; on the other hand, they worry about the leakage of their data to the unauthorized third party especially for the sensitive information such as personal health report, corporation financial report. Therefore, to preserve privacy the most direct method is that users preprocess the data to encrypt it before they store the sensitive data into the cloud. With data integrity detection scheme, they could verify the data at any time.

Shah et al. [22] considered the problem raised by integrity verification of data storage after it is encrypted and proposed

an effective solution. The trusted third-party verdict is introduced; on one hand this is for servers reputation to attract more users by concealing the truth of losing data; on the other hand, users may mistake their own fault of losing data as that of the servers, and their server wants to avoid this situation. Considering that the average users could not preserve their secret information for a long time, shah proposed storing the users' data along with the keys in the cloud, and hence the data integrity verification is needed as well as the key authentication. In the meantime, to not let the key be leaked to the third party, the article adopts multiple zero-knowledge proving [22] scheme to conduct key authentication.

Wang et al. [23] proposed a scheme to ensure public authentication and privacy preservation. The scheme is based on the discrete logarithm zero-knowledge proof scheme combined with bilinear pairing signature. The trusted third party cannot access any outsourced data in a half honesty state. Moreover, their scheme ensures the batch verification property.

5.2. Integrity Verification That Supports Dynamic Data Operation. Ateniese et al. [24] presented an EPDP solution based on symmetric cryptographic algorithm. They use MAC to get the hash values of data blocks and keep them locally. They illustrate that even if the users save 70 trillion outsourced data, the local users just need to save 128 M bit data. If verifying the integrity every 15 minutes, the saved hashed data are enough to use 16 years. They divide the data to blocks so that when the cloud data are modified by some operations such as updating, deleting, and adding, they do not need to download all data to calculate the hash values and they just need to operate on certain blocks.

Wang et al. [25] presented a solution that adds a preprocessing by RS codes. When users find that the data are incorrect, they can retrieve their data and correct $t = (d-1)/2$ errors, in which d is the minimum code distance. Although EPDP and Wang's solutions can support some dynamic data operation, they still can not achieve full dynamic maintenance and their performance overheads are relatively high for data addition operation.

Erway et al. [26] first discussed the complete dynamic operation problem. They used memory detection [27] and skip-lists [28] related technology to support the DPDP and improve the security. They proved that, under the standard module, this solution is more completeness and robustness than the PDP solution which is based on random oracle module. This solution also causes performance overhead. Its computing overhead and communication overhead have the $O(\log n)$ relationship with file size.

Wang et al. [7] also proposed a solution that supports the DPDP. However, their solution is limited in data updating, deleting, and appending. It is going to be very complicated when inserting data. In their follow-up work, they combined the bilinear pairing BLS signature [29] and Merkle Hash Tree integrity verification technology [30]. They assigned the data to a binary tree and signed the leave nodes to realize dynamic operation on data blocks. Their scheme supports the public authentication and its computational and communication overhead are $O(\log n)$, where n is the file size. Yuan and Yu [31] proposed a public integrity auditing scheme to support dynamic data sharing with multiuser maintenance.

Hao et al. [32] proposed a privacy protected solution that supports dynamic data operations. In this solution, the interaction data size is $O(1)$ while both the local saved data size and the server computing overhead are $O(n)$ (in which n is the saving data size).

Zheng and Xu [10] presented the FD-POR module that supports dynamic operations. Their module is based on 2-3 trees, which is a verified data structure, combined with an incremental signature method, which is also called hash signature. However, this solution cannot support public authentication.

5.3. Privacy Preserving in Cloud Data Checking. Ensuring the data auditing without any unnecessary information leakage is a critical concern in the practical application. Yu et al. [33] introduced the term, *zero-knowledge privacy*, to define the goal of privacy preserving in data integrity verification, which ensures that the TPA cannot obtain any additional information of file content from all the auxiliary verification information available. Fan et al. [34] proposed an indistinguishability-game-based definition, *IND-Privacy*, to evaluate the cloud data privacy preserving. They point out that many approaches are not theoretically secure according to the IND-Privacy definition. They also presented their example protocol that ensures content-integrity checking and satisfies the IND-Privacy.

5.4. Integrity Verification on Multicloud. By the extensive use of cloud storage, people start to consider saving their data among more than one cloud service provider. The integrity verification of multicloud becomes especially important [10].

Zhu et al. [9] proposed a scheme called CPDP that can achieve the integrity authentication of multicloud. The security of CPDP mechanism is based on zero-knowledge proof system. The verifier connects with the organizer, which may reduce the communication overhead and provide computing flexibility for the verifier. However, the protocol is found to be vulnerable by Wang and Zhang [35]. Any malicious CSP or organizer can generate response that can pass the authentication, even when it has already deleted all the data. Therefore, it does not have soundness guarantee. Wang [36] presented ID-DPDP (identity-based distributed provable data possession) scheme. Under the standard CDH problem assumption, the scheme is provably secure and can support regular verification, delegate verification, and public verification as well.

We present the theoretical comparison of various schemes in Table 2. In summary, our scheme has the following features: fast computation speed, low storage overhead, low bandwidth requirement, and support for sampling, unlimited challenge-response interactions.

Table 2: Theoretical analysis of various solutions.

Scheme	Communication	Computation		Storage		Property			
		Client	Server	Client	Server	Type	Unbounded	Error correct	Batch verify
PDP [5]	$O(1)$	$O(c)$	$O(c)$	$O(1)$	$O(n)$	Pr.	√	×	×
RIC [3]	$O(1)$	$O(1)$	$O(n)$	$O(t)$	$O(n)$	Det.	√	×	×
POR [16]	$O(1)$	$O(1)$	$O(c)$	$O(t)$	$O(n+k)$	Pr.	×	√	×
CPOR [17]	$O(1)$	$O(c)$	$O(c)$	$O(1)$	$O(n+k)$	Pr.	√	√	×
DPDP [6]	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$	$O(n)$	Pr.	√	×	×
CPDP [9]	$O(1)$	$O(c)$	$O(c)$	$O(t)$	$O(n)$	Pr.	×	×	×
Wang [37]	$O(\log n)$	$O(\log n)$	$O(\log n)$	$O(1)$	$O(n)$	Pr.	√	×	×
Hao [32]	$O(1)$	$O(1)$	$O(c)$	$O(n)$	$O(n)$	Det.	√	×	×
Our Scheme	$O(1)$	$O(c)$	$O(c)$	$O(1)$	$O(n)$	Pr.	√	×	√

t is the verification time allowed by the scheme; Type means guarantee types provided by the scheme; Det. and Pr. represent *Deterministic* guarantee and *Probabilistic* guarantee, respectively; n is the number of the file blocks.

Redundancy Analysis. Basically, there are three ways to ensure the remote cloud storage audition, that is, hash-function-based solutions, homomorphic-authentication-tag-based solutions, and network-coding-based solutions. All these approaches will cause unavoidable redundancy and thus additional storage overhead at the server side. Furthermore, there are potential security vulnerabilities in hash-function-based solutions, while network-coding-based solutions cause higher storage overhead than homomorphic-authentication-tag-based solutions. In our scheme, we create the authentication tags based on Boneh-Lynn-Shacham (BLS) signature, whose additional redundancy is $|p|$ bits for one file block. In contrast, in Zhu's scheme [9], their storage redundancy for authentication tag per file block is also $|p|$ bits; the redundancy caused by Wang's approach [35] is $|H(\cdot)| + p$ bits per file block; and the approach proposed by Yuan and Yu [31] introduced $|p|$-bit additional storage overhead at the server side to verify one file block. Compared with existing related solutions, the redundancy rate introduced by our scheme is relative low.

6. Conclusion

Together with cloud-computing technology, heterogeneous wireless networking technology has become a critical infrastructure adopted by emerging communication networks. Due to the convenience, low cost, and flexibility, cloud storing techniques become widely used in remote services, such as wireless data outsourcing and resource sharing. However, users lose the physical control of their outsourced data, while the cloud storage service provider is not always trustworthy. Consequently, how to ensure the integrity of their remotely outsourced data becomes a serious concern for users to select cloud storing services. In this paper, we present a collaborative multicloud data integrity audition scheme, which is based on BLS signatures and homomorphic tags. According to the proposed scheme, users can audit their outsourced data in a one-round challenge-response process. In addition, our scheme also enables dynamic data maintenance (e.g., data modification, insertion, and deletion). The theoretical analysis demonstrates the effectiveness of our scheme and the probability that the dishonest CSP server can bypass the checking successfully is neglectable if the one-way hash function is collision-resistant and the computational Diffie-Hellman (CDH) assumption holds.

Notation

m_{ij}: The jth file sector of the ith file block
p: A large prime
G: A cyclic group with order p and generator g
e: $e : G \times G \to G_T$ is a nondegenerate bilinear map
π: The system parameter and $\pi = (p, G, G_T, e, g)$
l: The number of CSPs
F: The file outsourced by a user
F_n: The file name
Q: Challenge created by users
H: Hash function
pk: Public key
sk: Private key
λ: Security parameter
σ_{ik}: The tag for the kth copy of the file block m_i
num_i: The number of copies of the file block m_i
V_i: The updated times of the file block m_i
χ_i: The initial representation of the file block m_i
CSP_k: The kth CSP server
c: The amount of sampled file blocks to be sampled.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China (no. 61402029), the National Key R&D Program of China (no. 2017YFB0802400), the National Natural Science Foundation of China (no. 61379002, no. 61370190), Beijing Natural Science Foundation (no. 4162020), the Funding Project of Education Ministry for

the Development of Liberal Arts and Social Sciences (no. 12YJAZH136), and the Funding Project of Shanghai Key Laboratory of Integrated Administration Technologies for Information Security (no. AGK201708).

References

[1] L. Yu, L. Chen, Z. Cai, H. Shen, Y. Liang, and Y. Pan, "Stochastic Load Balancing for Virtual Resource Management in Datacenters," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2016.

[2] L. Yu and Z. Cai, "Dynamic scaling of virtual clusters with bandwidth guarantee in cloud datacenters," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, IEEE, San Francisco, CA, USA, April 2016.

[3] Y. Deswarte, J.-J. Quisquater, and A. Sadane, "Remote integrity checking," *Integrity and Internal Control in Information Systems VI*, vol. 140, p. 11, 2004.

[4] D. L. Gazzoni Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," *Cryptology ePrint Archive*, vol. 2006, pp. 150–158, 2006.

[5] 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.

[6] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *Proceedings of International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pp. 319–333, Springer-Verlag, London, UK, 2009.

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

[8] Y. Zhu, H. Hu, G. Ahn, Y. Han, and S. Chen, "Collaborative Integrity Verification in Hybrid Clouds," in *Proceedings of the 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, pp. 191–200, Miami, Fla, USA, October 2011.

[9] 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, pp. 2231–2244, 2012.

[10] Q. Zheng and S. Xu, "Fair and dynamic proofs of retrievability," in *Proceedings of the first ACM conference on Data and application security and privacy*, pp. 237–248, ACM, Tempe, AZ, USA, February 2011.

[11] J. Mao, J. Cui, Y. Zhang, H. Ma, and J. Zhang, "Collaborative outsourced data integrity checking in multi-cloud environment," in *Proceedings of the International Conference on Wireless Algorithms, Systems, and Applications (WASA '16)*, pp. 511–523, 2016.

[12] F. Sebe, A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferrer, and J. Quisquater, "Time-bounded remote file integrity checking," Tech. Rep., July 2004, 04429, LAAS.

[13] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte, and J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1034–1038, 2008.

[14] 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.

[15] S. Shoup, "On the security of a practical identification scheme," *Journal of Cryptology*, vol. 12, no. 4, pp. 247–260, 1999.

[16] 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.

[17] 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.

[18] 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, ACM, Chicago, IL, USA, November 2009.

[19] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: a high-availability and integrity layer for cloud storage," in *Proceedings of the 16th ACM conference on Computer and Communications Security*, pp. 187–198, ACM, Chicago, IL, USA, November 2009.

[20] M. Naor and G. N. Rothblum, "The complexity of online memory checking," in *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2005*, pp. 573–582, Pittsburgh, PA, USA, October 2005.

[21] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2012*, pp. 79-80, Seoul, Republic of Korea, May 2012.

[22] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," *IACR Cryptology EPrint Archive*, vol. 2008, pp. 186–206, 2008.

[23] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proceedings of the 29th Annual IEEE International Conference on Computer Communications (INFOCOM 2010)*, pp. 1–9, Toronto, Canada, March 2010.

[24] 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.

[25] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proceedings of the 17th International Workshop on Quality of Service (IWQoS '09)*, pp. 1–9, IEEE, July 2009.

[26] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 4, article 15, 2015.

[27] D. Clarke, S. Devadas, M. Van Dijk, B. Gassend, and G. E. Suh, "Incremental multiset hash functions and their application to memory integrity checking," in *Proceedings of the 9th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 188–207, Springer, 2003.

[28] M. T. Goodrich, R. Tamassia, and A. Schwerin, "Implementation of an authenticated dictionary with skip lists and commutative hashing," in *Proceedings of the DARPA Information Survivability Conference and Exposition II, DISCEX 2001*, pp. 68–82, USA, June 2001.

[29] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings of the ASIACRYPT 2001*, pp. 514–532, Springer, Berlin, 2001.

[30] R. C. Merkle, "Protocols for public key cryptosystems," in *Proceedings of the 1980 IEEE Symposium on Security and Privacy, SP 1980*, pp. 122–134, Oakland, Calif, USA, April 1980.

[31] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 8, pp. 1717–1726, 2015.

[32] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[33] Y. Yu, M. H. Au, Y. Mu et al., "Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage," *International Journal of Information Security*, vol. 14, no. 4, pp. 307–318, 2015.

[34] 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.

[35] H. Wang and Y. Zhang, "On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 264–267, 2014.

[36] H. Wang, "Identity-based distributed provable data possession in multi-cloud storage," *IEEE Transactions on Services Computing*, 2014.

[37] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proceedings of the 2009 European Conference on Research in Computer Security*, pp. 355–370, Springer, Saint-Malo, France, 2009.