information

MDPI

Article

Efficient Dynamic Integrity Verification for Big Data Supporting Users Revocability

Xinpeng Zhang [1,2,*], Chunxiang Xu [1], Xiaojun Zhang [1], Taizong Gu [2], Zhi Geng [2] and Guoping Liu [2]

[1] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China; chxxu@uestc.edu.cn (C.X.); xiaojunzhang_019@126.com (X.Z.)

[2] Logistic Information Center, Joint Logistics Department, Chengdu Military Region, Chengdu 610015, China; gutaizhong2015@tom.com (T.G.); gengzhi2016@21cn.com (Z.G.);18981990008@189.cn (G.L.)

* Correspondence: carriage1029@163.com; Tel.:+86-28-86683696

Academic Editor: Gordana Dodig-Crnkovic
Received: 6 February 2016; Accepted: 12 April 2016; Published: 27 May 2016

Abstract: With the advent of the big data era, cloud data storage and retrieval have become popular for efficient data management in large companies and organizations, thus they can enjoy the on-demand high-quality cloud storage service. Meanwhile, for security reasons, those companies and organizations would like to verify the integrity of their data once storing it in the cloud. To address this issue, they need a proper cloud storage auditing scheme which matches their actual demands. Current research often focuses on the situation where the data manager owns the data; however, the data belongs to the company, rather than the data managers in the real situation which has been overlooked. For example, the current data manager is no longer suitable to manage the data stored in the cloud after a period and will be replaced by another one. The successor needs to verify the integrity of the former managed data; this problem is obviously inevitable in reality. In this paper, we fill this gap by giving a practical efficient revocable privacy-preserving public auditing scheme for cloud storage meeting the auditing requirement of large companies and organization's data transfer. The scheme is conceptually simple and is proven to be secure even when the cloud service provider conspires with revoked users.

Keywords: cloud storage; privacy-preserving; integrity verification; user revocation

1. Introduction

Nowadays, a large amount of data has been gathered and produced by individuals, companies and organizations. Moore's law is broken by the rapid growth of the data scale. The growth of the data scale is far more than the growth of the processing and storage capacity of computer. For companies and organizations, the volumes of those data are often so tremendous that they cannot process and manage it effectively by themselves. In fact, some of them even don't have sufficient disk space to store their data because it's an enormous burden to purchase such a large number of disks. Facing this reality, companies and organizations have to turn to cloud service provider (CSP) for help, e.g., Dropbox, Google Drive and skyDrive.

As one of the dominate services in cloud computing, cloud storage allows users to store data on clouds instead of their local computing systems. By data outsourcing, this kind of new storage service has many advantages such as relieving users' burden in terms of data management and maintenance, universal data access with independent geographical locations and avoiding capital cost on hardware and software. However, at the meantime, cloud storage also brings a number of challenging security problems [1–3] despite its appealing features. Security concerns still deter potential consumers from using the service. One of the major security concerns [1] on the cloud storage service is whether

the cloud could ensure the integrity of the stored data. Integrity challenges of data corruption are inevitable [4–6], but cloud service providers may not be fully trusted from the view of the interests. Cloud Security Alliance (CSA) conducted a systematic investigation into reported vulnerabilities in cloud computing such as outages, downtimes, and data loss. CSA also released a white paper [7] in 2013 which revealed that the top three threats were "Insecure Interfaces & APIs", "Data Loss & Leakage" and "Hardware Failure". These three threats accounted for 64% of all cloud outage incidents while "Data Loss & Leakage" accounted for 25%. Consequently, guaranteeing the integrity of the data, or data auditing, in cloud is a highly desirable security demand for secure cloud storage. Many researches have been done on checking the integrity for outsourcing data in the cloud. Despite a number of cloud data auditing schemes [8–15] have been proposed with different requirements so far, they are all designed for traditional cloud storage environment without considering the applications for user revocable.

We notice that almost all of the previous public auditing systems are fixed by the user who computes the block tags. In other words, those auditing schemes require that the user of the cloud storage service is always the same one during the entire data period. However, it is impractical. On one hand, the verification information of an auditing system such as the user's public key may expire after a period of time. On the other hand, the user may be a data manager of a company for a time and may leave for some reasons. For example, the data manager may go to work in another company for a higher salary. Therefore, for practical considerations, an auditing scheme should support efficient user revocation.

Recently, a few public auditing schemes for cloud storage systems with user revocation have been presented, e.g., [16–18]. However those schemes are designed for auditing shared cloud data rather than for revoking inappropriate users when auditing owned cloud data. Moreover, we note that the existing users revocable public cloud storage auditing schemes are either involved or less secure. Specifically, the revocable public cloud storage auditing schemes in [17] and [18] employ the unwieldy dynamic broadcast encryption [19] and group signature [20] techniques respectively. Although the scheme in [16] is more efficient, it can't resist collusion attacks between the cloud and a revoked user. That is, the collusion of the cloud and a revoked user could always deceive an incumbent user into belief that the data in the cloud remains intact even if it's actually not. Thus collusion attack resistance is indispensable in a revocable public cloud storage auditing schemes. As a result, it's crucial to design efficient and collusion-resistant user revocable public auditing schemes.

1.1. Related Work

Juels *et al*. proposed an auditing scheme called Proofs Of Retrievability (POR) while the auditing scheme proposed by Ateniese *et al*. is called Provable Data Possession (PDP). Shacham-Waters used BLS signature constructed an efficient public verifiable POR scheme [13]. Based on their research, many cloud storage auditing schemes have been proposed to verify the data integrity without needing to retrieve entire data [8–13]. However, the privacy protection of user's data has not yet been considered in most of these schemes [11,13]. This shortcoming can greatly affect the safety of these schemes. Therefore, the auditing process should not leak the knowledge of the challenged files to the third-party auditor. In 2013, Wang *et al*. [9] presented a privacy-preserving public auditing scheme for cloud storage; it resorts to the homomorphic authenticator technique and random masking technique to realize privacy-preserving public auditing and take advantage of the technique of bilinear aggregate signature to realize batch auditing.

All the auditing schemes mentioned above do not consider the user revocation problem, thus those schemes can only be applied to static users. However, user revocation is an obviously inevitable problem. Recently, a few auditing schemes supporting user revocation are published for realizing multi-user shared cloud storage audit. In 2012 Wang *et al*. [21] first introduced the shared cloud storage auditing issue and proposed a private auditing scheme with user revocation based on group

signature [20]. In 2013, Wang *et al*. [17] presented a public auditing scheme with user revocation for shared cloud storage, based on the dynamic broadcast encryption scheme of [19] and the bidirectional proxy re-signature scheme of [21]. Later, using a group signature like technique, Yuan and Yu proposed a public version of the scheme in [18]. As group signature and dynamic broadcast encryption techniques are both involved, the above revocable auditing schemes are all less efficient in practice. To address this problem, in 2015 Wang *et al*. presented an efficient revocable public auditing scheme in [16] by just using the bidirectional proxy re-signature scheme of [22]. However, we note that the bidirectional proxy re-signature scheme cannot resist the collusion attack of the cloud and a revoked user since an incumbent user's secret key can be recovered from the cloud's update key and a revoked user's secret key.

We also notice that all the previous papers focus on the data integrity and security are under the shared cloud storage model [23,24]. Although these schemes involve user revocation problem, the main research is still cloud data sharing, where security problems cannot be ignored. Therefore, we analyze the revocation need of companies and organizations cloud storage data users, propose the model of user revocable auditing schemes and design an efficient dynamic integrity verification scheme for big data supporting user revocability. This is the major work we are doing in this paper.

1.2. Our Contributions

Motivated by above, in this paper, an efficient dynamic integrity verification for big data supporting users revocability and third-party privacy-preserving auditing scheme be proposed. To achieve this, we make the following contributions: we analyze the revocation need of companies and organizations cloud storage data users. Based on technique of bilinear aggregate signature, a specific revocable public cloud storage third-party auditing scheme be presented. It can help the current user audit the data which was sent to the cloud by all the previous users, and can satisfy the user transfer demand of large companies and organizations. Meanwhile cloud users can delegate a third party (TPA) to perform security auditing tasks as it is not economically feasible for them to handle it by themselves. By given a precise definition of security that collusion resistance is mandatory. At last by analyzing the performance of scheme and the results, we demonstrate that our scheme is efficient.

1.3. Paper Organization

The remainder of this paper is organized as follows. Preliminaries is described in Section 2. Section 3 formalizes the concept of revocable third-party privacy-preserving auditing scheme for cloud storage and also presents our design goals. The revocable third-party privacy-preserving auditing scheme for cloud storage is given in Section 4. Section 5 analyzes the scheme security. Section 6 analyzes the performance of it. Finally, Section 7 concludes this paper.

2. Preliminaries

2.1. The User Data Stored in the Cloud

As illustrated in Figure 1, a basic cloud storage auditing system involves two main entities: a user and the CSP. The user would be a company or an organization (more precisely, it is usually a data manager of them who uses the cloud storage service to store its superabundant data. The CSP is cloud service provider who has ample storage space, and could offer economical and professional storage services to users. Specifically, a cloud storage auditing scheme works as follows. A user first splits the data M into n blocks such that each block is m_i in Z_p, *i.e.*, $M = (m_1, \ldots, m_n) \in (Z_p)^n$, $M \in \{0,1\}^*$, $i \in \{1, \cdots, n\}$, and computes the signatures of all blocks using its secret key like $\sigma = (\sigma_1, \ldots, \sigma_n)$. Here the signatures are known as block tags. Then the user sends the data and all tags to the cloud, and deletes them locally. When their outsourced data needs to be checked, the user picks a random set of data blocks and sends a corresponding $Q = \{(i, v_i)\}$ to the cloud, where i and v_i

indicate the identity and random coefficient of a selected data block respectively. After receiving Q , the cloud calculates and returns a proof by using those data blocks as well as the corresponding tags. Finally, the user verifies the validity of the proof. If the proof is invalid then the user can confirm that its data has been damaged. Otherwise, it may be intact; the user could repeat the challenge verification procedure until getting a confirmation. It obviously shows that the cloud only stored the user data block and the corresponding blocks tags.

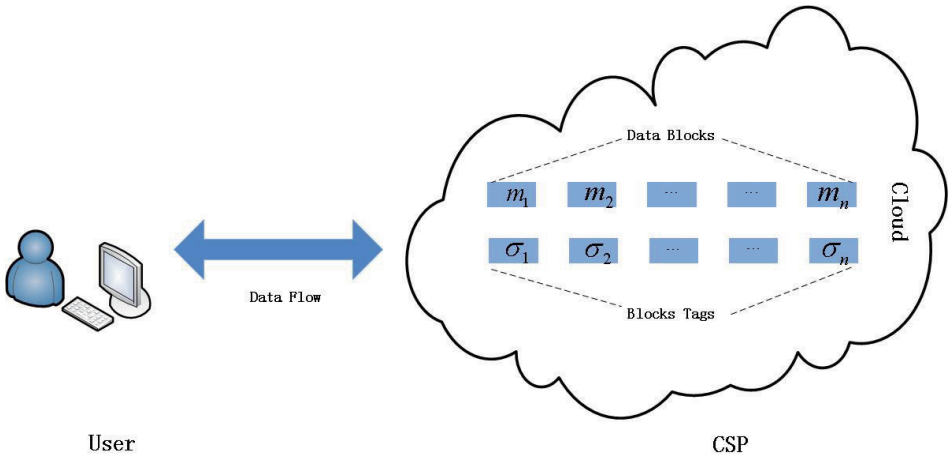

Figure 1. The user data stored in the cloud.

2.2. *Multi-User Data Stored in the Cloud with the Revocable System*

As shown in Figure 2, cloud storage system supporting revocable user is quite different from the basic cloud storage auditing system, as there are many users who are able to manage the same piece of data. In reality, the data stored in the cloud belongs to the company, not to the data manager. In a specific period, there is usually one data manger that is responsible for managing the data, but in a longer time period, there might be many users who are able to managing the data. That is, after some time, a data manager who is responsible for managing the data is no longer suitable to manage the data, e.g., the data manager leaves the company and work for another company, thus, a successor of the data manager is needed.

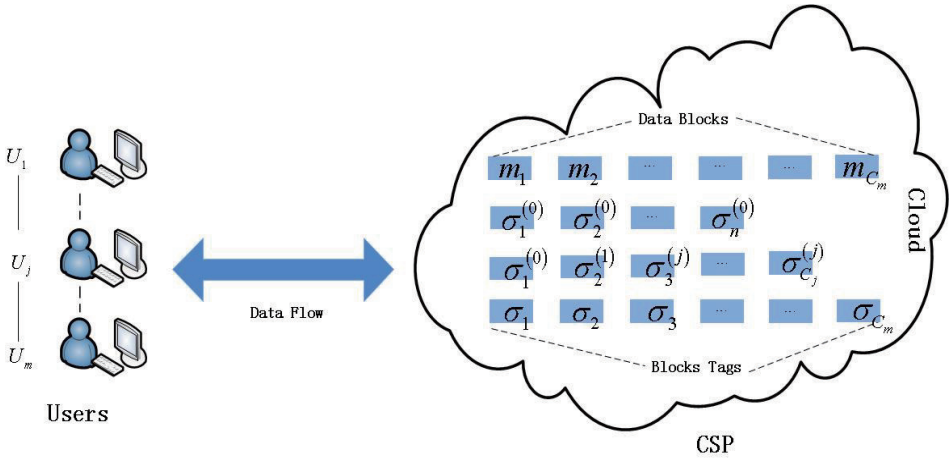

Figure 2. Multi-user data stored in the cloud.

We assume that there is an initial user who uploads the company's data to the cloud on behalf of the company, we regard this initial user as U_0, then the company recruit a data manger to manage those data stored on the cloud. Clearly the data manager is not tenure. Before leaving, a data manger

needs to transfer all data he managed to his successor. The successor also needs to verify these data to make sure all the data stored on the cloud is intact. Assume that a company or organization only needs one data manager to manage the data in a specific period. Then we have $U_1, U_2, \cdots, U_m$ users in the company or organization where m is a positive integer. So the data management period is divided into $T_1, T_2, \cdots, T_m$ accordingly. And the user has to transfer the data to the successor at the end of the period. (Note: In the paper the initial user U_0 can't do anything except uploads the data to the cloud. Only $U_1, U_2, \cdots, U_m$ can management the data.)

The initial user U_0 first divides all the files into n blocks, and calculates its corresponding tag σ using his secret key, then uploads the data and tag to the cloud. U_1 manages the data during the period of T_1, then U_1 will be replaced by its successor U_2 at the end of T_1, U_2 will also be replaced by U_3 some time later, and so on till the U_j replaces U_{j-1},where $j \in \{0, \cdots, m\}$ and U_j is the current user.

As the tag is signed by the user, if a user has been revoked, the tags computed by the user should be modified. An obvious approach to update those tags is re-computing the tags of data blocks using the current user's secret key. However, this is not a cloud storage auditing scheme supporting user revocation, as this method introduces large communication and computation overhead. All the data manager can add, modify and delete the data which is stored on the cloud. For the current user U_j all these operations can only happen during the T_j period. For the add operation, U_j divides the data into blocks, computes the tags of each block and sends all the blocks and tags to the cloud. For the modify operation, U_j first retrieves the data which needs to be modified and its corresponding tags. U_j verifies the correctness of the data, and discards the tags. If the data is intact, then U_j modifies the data and computes the tags for the data using his secret key and uploads the data and tags to the cloud. For simplicity, we assume that the cloud server can handle the delete operation effectively. (e.g., if some deleted data are selected by a challenge, all the data are set to 0, this will not affect the alter verification process of the data. In fact, those deleted data will no longer take any space on the cloud server.) Thus for serial number of blocks, its value will never decrease.

The value of the $i-th$ of blocks C is related to the period T and the operation P. Assume that $C_1, \cdots, C_m$ are the $i-th$ of blocks at the end of period $T_1, T_2, \cdots, T_m$, and $c_1, \cdots, c_m$ are the increment of the data block at the end of period $T_1, T_2, \cdots, T_m$, and $p_{j,1}, p_{j,2}, \cdots, p_{j,\theta}$ are the increment of the data block by the operation $P_{j,1}, P_{j,2}, \cdots, P_{j,\theta}$ during the period T_j. Then we get $C_j = n + \sum_{\ell \in [1,j]} c_\ell = C_{j-1} + c_j$, where $\ell \in \{1, \cdots j\}$, and $\sum_{k \in [1,\theta]} p_{j,k} = c_j$, where $p_{j,k}$ is a positive integer and $k \in \{1, \cdots \theta\}$. So at the auditing time the value of the $i-th$ of blocks is $C_{j,k} = C_{j-1} + p$, where $p = p_{j,1} + p_{j,2} + \cdots + p_{j,k}$.

For a more realistic cloud storage system supporting user revocation, all the data stored on the cloud included the data $m_1, \ldots, m_{C_m}$ and its corresponding tags $\sigma_1, \ldots, \sigma_{C_m}$, and its corresponding period $T_1, T_2, \cdots, T_m$. They are uploaded by the initial users U_0 and all the other data managers $U_1, U_2, \cdots, U_m$. So as shown in Figure 3, the integrity verification of m_i will be verified by σ_i , C, T.

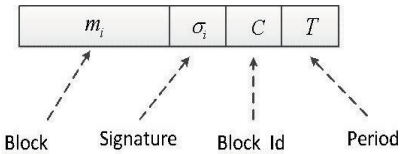

Figure 3. Each block is attached with a signature, a block id and a current period.

As mentioned above, U_j can only add and modify data at the time period of T_j, and compete the tags of data blocks using his own secret key. In order to distinguish those tags, we use $\sigma_i^{(j)}$ to represent the tags computed by the user U_j for data block m_i. For the current user, he has to not only manage the data blocks $m_{C_{(j-1)+1}}, \ldots, m_{C_{j,k}}$ and their corresponding tags, but also manage all the data blocks and tags which were uploaded to the cloud by all of his predecessors. Some of the tags might be

signed by different users. For example, in time period T_1, user U_1 did modify operation which gets data block m_2 and tag $\sigma_2^{(1)}$; at the current period, user U_j modifies data block m_i and computes its tag $\sigma_i^{(j)}$.

2.3. The Revocable Scheme Supported Third-Party Privacy-Preserving Auditing

Due to reason of the online burden which potentially brought by the periodic storage correctness verification, cloud users tend to delegate a third-party auditor (TPA) to execute security auditing tasks. Through the TPA automatic execution periodic auditing tasks can save communication resources effectively. Therefore, the third-party auditing schemes are more desirable in the real world. As illustrated in Figure 4, a revocable cloud storage third-party auditing scheme works as follows. When the user wants to check its outsourced data, it sends a verify request to the TPA. When the TPA receives the request, it picks a random set of data blocks and sends a corresponding $Q = \{(i, v_i)\}$ to the cloud, where i and v_i indicate the identity and random coefficient of a selected data block respectively. After receiving Q, the cloud calculates and returns a proof using those data blocks as well as the corresponding tags. Then, the user verifies the validity of the proof. If the proof is invalid then the TPA can confirm that its data has been damaged. Otherwise, it may be intact; the TPA could repeat the challenge verification procedure until getting a confirmation. Finally, the TPA sends the result to the user. It is obvious that the cloud only stored the user's data block and the corresponding blocks tags.

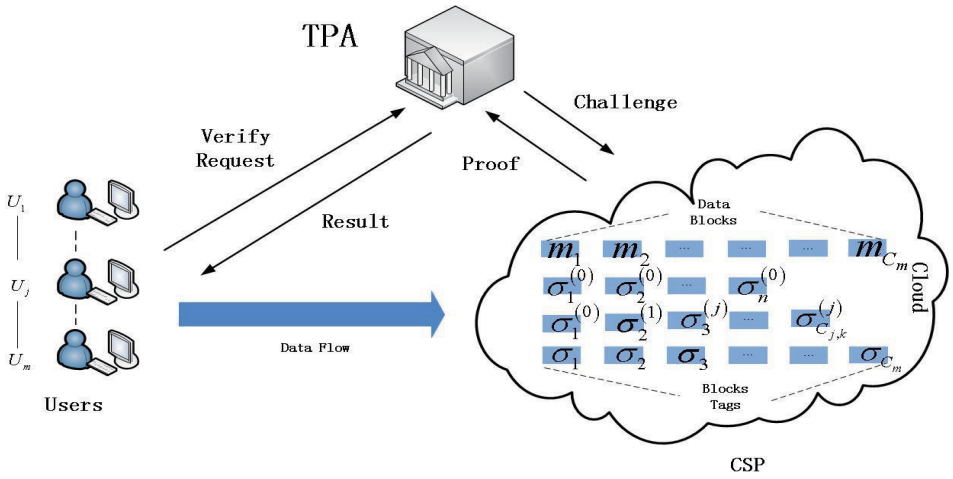

Figure 4. An efficient and security revocable third-party privacy-preserving auditing scheme for cloud storage.

3. Formalization and Definitions

Without loss of generality , a revocable third-party privacy-preserving auditing scheme for cloud is assumed as shown in Figure 4, which involves in $m + 1$ authorized users for some $m \in Z > 0$ and their sequence is U_0, U_1, U_2, $\cdots$, U_m, corresponding the manager period T_1, T_2, $\cdots$, T_m. (Notice that unauthorized users can be easily recognized and additionally they cannot impair the integrity of the outsourced data. Thus it can be assumed that there is no unapproved user in our auditing schemes.) Then such an auditing scheme can be defined as below.

3.1. Definition 1: Revocable Third-Party Privacy-Preserving Auditing Scheme for Cloud Storage

A revocable third-party privacy-preserving auditing scheme for cloud storage consists of six probabilistic polynomial time (PPT) algorithms ($Setup$, $SigGen$, $Update$, $Challeng$, $ProofGen$, $ProofVerify$), where:

Setup: This algorithm is to generate each user's public/secret keys and run by each user U_j, where $j \in \{0, \cdots, m\}$. For the $j-th$ user U_j, the algorithm takes as input a security parameter λ and outputs $U_j's$ public-secret key pair (pk_j, sk_j).

SigGen: This algorithm is to generate the tags of the stored data. It consists of three child probabilistic polynomial time algorithms($SigGen\,(U_0)$, $SigGen\,(U_j)$, $SigGen\,(U_\ell \to U_j)$).

$SigGen\,(U_0)$: This algorithm is to generate the initial block tags of the stored data in the initial time and thus will be run by the initial user U_0 The algorithm takes as input $U_0's$ secret key sk_0 and block data $(m_1, \ldots, m_n)$, $m_i \in \{0,1\}^*$, $i \in \{1, \cdots, n\}$, and outputs the verification metadata V of $(m_1, \ldots, m_n)$ associated with the user U_0. After that, U_0 sends V and $(m_1, \ldots, m_n)$ to the cloud and deletes them locally.

$SigGen\,(U_j)$: This algorithm is to generate the block tags of the stored data in the period T_j on the operation $p_{j,k'}$, $k \in \{1, \cdots, \theta\}$, $k' \in \{1, \ldots, k\}$ and thus will be run by the current user U_j. The algorithm takes as input $U_j's$ secret key sk_j and block data $\left(m_{C_{j,(k'-1)}+1}, \ldots, m_{C_{j,k'}}\right)$, $m_i \in \{0,1\}^*$, where $C_{j,k'}$ is a positive integer. And the output of verification metadata V of $\left(m_{C_{j,(k'-1)}+1}, \ldots, m_{C_{j,k'}}\right)$ is associated with the user U_j. After that, U_j sends V and $\left(m_{C_{j,(k'-1)}+1}, \ldots, m_{C_{j,k'}}\right)$ to the cloud and then deletes them locally.

$SigGen\,(U_\ell \to U_j)$: This algorithm is to generate the block tags of the stored data in the period T_j when U_j wants to update the data which the previous user U_ℓ uploaded to the cloud. So it will be run by the current user U_j. The algorithm first retrieving the data block $m_i$and it corresponding tags, then verified it if invalid turn out, if valid U_j replaced the m_i by m_i^* (For simple reason we also record it as m_i too). Later the algorithm takes U_j's secret key sk_j and block data m_i as input and the outputs of verification metadata V of m_i associated with the user U_j. After that, U_j sends V and m_i to the cloud and then deletes them locally.

Update: This is an interactive algorithm for updating users. Suppose the user U_j needs to be replaced by the user U_{j+1}, then U_{j+1} will initiate the algorithm. After the algorithm ends, U_{j+1} would obtain an update $uk_{j\to j+1}$ for the cloud, and finally sends it to the cloud.

Challeng: This is an interactive algorithm for users send checking order. Assume U_j is the current user and wants to check its outsourced data, it sends a verify request to the TPA. When TPA received the request, it picks a random set of data blocks and sends a corresponding $Q = \{(i, v_i)\}$ to the cloud, where i and v_i indicate the identity and random coefficient of a selected data block respectively.

ProofGen: After receiving *Challeng*, the cloud would run the algorithm to return a response. To do this, the algorithm takes as input the *Challeng*, the block data m_i, $i \in Q$ and the verification metadata V of $\{m_i\}_{i\in Q}$, and outputs a verification *proof*.

ProofVerify: This algorithm is run by the TPA to verify the correctness of the *proof*. The algorithm takes as input $U_j's$ public key pk_j, the *Challeng* and the corresponding *proof*, and outputs VALID if *proof* is valid; INVALID otherwise. Finally, the TPA sends the result to the U_j.

For easier understanding, the revocable third-party privacy-preserving auditing scheme for cloud storage intuition behind the definition is given here. The basic idea of our security definition is: if the data in the cloud is indeed damaged but the cloud cannot admit, even by colluding with the revoked users, fool the current user into believing that the data remains intact. Let the cloud be an adversary A. To model the collusion between the cloud and revoked users, we permit to query a Corrupt oracle which takes a revoked user's identity as input and outputs the user's secret key. However, according to the aforementioned reasons we prohibit A from querying the Corrupt oracle on the user's identity. Additionally, like other security models, our security model also allows A to query *SigGen* oracle, *Update* oracle as well as the *ProofGen* oracle for obtaining the initial block tags, all update keys and valid proofs of any challenges.

3.2. Definition 2: Security Model

Now we describe the security definition of revocable third-party privacy-preserving auditing scheme for cloud storage. A revocable third-party privacy-preserving auditing scheme for cloud storage is secure if for any polynomial time adversary A the probability wins the following game played between a challenger C and the adversary A is negligible.

Setup: The challenger C first runs the algorithm KeyGen(λ) to generate $U_j's$ public-secret key pair (pk_j, sk_j) for all $j \in \{0, \cdots, m\}$, and then sends all public $\{pk_j\}_0^m$ to the adversary A.

Query: The adversary A could query the following oracles adaptively.

SigGen-Oracle: For any data block $m \in \{0,1\}^*$, if A wants to get the initial block tags of m, it will query the oracle on m. After receiving the query, the challenger C first runs the algorithm $SigGen(sk_0, m)$ to produce a result V_0 and then returnsV_0 as response.

Update-Oracle: When A believes some user is not suitable for auditing, A will query the oracle on the user's identity to replace the user with its successor. Assume the user to be replaced is U_{j+1} for $j \in \{1, \cdots, m-1\}$. The challenger C first runs the algorithm $Update\left(U_{j+1}\right)$ to produce a update key $uk_{j\to j+1}$ and then sends it to A. After receiving $uk_{j\to j+1}$, A could generate the verification metadata V_{j+1} of a data block m associated with the user U_{j+1} using the update key $uk_{j\to j+1}$, the data block m and the verification metadata V_j of m associated withU_j.

Corrupt-Oracle: Suppose all revoked users at present are $U_0, U_1, \cdots, U_d$ for some $d \in \{0, \ldots, m-1\}$, then the adversary A could query the oracle on any of them, with the exception of only U_0. When receiving such a query on the user U_ℓ for $\ell \in \{1, \ldots, d\}$, the challenger C returns $U_\ell's$ secret key sk_ℓ as response.

Proof. In order to verify whether the data block m stored in the cloud is the same as before, the challenger C generates a random challenge $Chal$ and requests the adversary A to return a proof of m associated with user U_j where $j \in \{0, \ldots, m\}$. On input the challenge $Chal$, the data block m and the verification metadata U_j of M associated with U_j, the adversary A outputs a proof as response. □

Forgery. When the above process ends, the adversary A finally outputs a proof of some challenge $Chal$ on file M with respect to user U_j, where $j \in \{0, \ldots, m\}$. We say A wins the game if the following conditions hold:

1. $Verification\,(pk_\ell,\ chal,\ proof) \to Valid$;
2. The data block m is not the original one.

3.3. Design Goals

To support secure and efficient user revocable and data privacy preserving in a public cloud data auditing scheme, we have the following design goals:

(i) TPA is allowed to verify the correctness of the cloud data. It executes data auditing without retrieving entire data and introduces none additional online burden to the user.
(ii) Storage correctness: If the cloud indeed stores entire data, then it would always output valid proofs.
(iii) Privacy-preserving: TPA learns no information of the stored data from information collected during the auditing process.
(iv) Revocability: If a user is revoked, then its successor could establish a new auditing procedure efficiently.
(v) Collusion resistance: If the data stored in the cloud is changed, then the auditing scheme should be able to detect it with high probability even though the cloud colludes with revoked users.
(vi) Efficiency: the computation, communication and storage overhead should be as small as possible.

4. The Revocable Third-Party Privacy-Preserving Auditing Scheme for Cloud Storage

This section gives some preliminaries to be used in this work, including bilinear map and hardness assumptions.

4.1. Bilinear Map

Let G_1, G_2 and G_T be cyclic groups with the same prime order p. A map e: $G_1 \times G_2 \to G_T$ is called a bilinear map if it satisfies the following three properties.

1. Bilinearity: For all $a, b \in Z_p$, and $u \in G_1, v \in G_2$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g_1, g_2) \neq 1$.
3. Computability: There exists an efficient algorithm to compute the map e.
4. Exchangeability: $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$

4.2. Hardness Assumptions

The security of our constructions will rest on the Computational Diffie-Hellman (CDH) assumption and the Discrete Logarithm (DL) assumption.

Definition 3 (CDH Assumption)**.** *The CDH assumption is given a group of prime order p, a generator g, and two random element $g^a, g^b \in G$ it's hard to output g^{ab}.*

Definition 4 (DL Assumption)**.** *The DL assumption is given a group G of prime order p, a generator g, and a random element $g^c \in G$ it's hard to output c.*

4.3. Specification

In this section, the revocable third-party privacy-preserving auditing scheme for cloud storage is proposed. The auditing scheme is illustrated in Figure 4. Here, a semi-trusted TPA is needed to define, which is only responsible for auditing the integrity of data blocks honestly. However, it is curious and may try to reveal the user' primitive data blocks based on verification information. In this paper, the scheme includes of the following six algorithms: $Setup$, $SigGen$, $Update$, $Challeng$, $ProofGen$, $ProofVerify$.

Let G and G_T be two cyclic groups with the same prime order p, and g be a generator of G.t.

Let $e : G \times G \to G_T$ be a bilinear map and $H : \{0,1\}^* \to G$, $h : \{0,1\}^* \to G$, $f_{k_3} : \{0,1\}^* \to Z_p$ be a hash function. The auditing scheme is specified as follows.

$Setup$. On input a security parameter λ, each user U_j where $j \in \{0, \cdots, m\}$ does the following steps:

1. Select a random $x_j \in Z_p$.
2. Compute its public key g^{x_j}.
3. Output $pk_j = g^{x_j}$ and $sk_j = x_j$.

$SigGen$. This algorithm is to generate the tags of the stored data. It consists of three child algorithms ($SigGen\,(U_0)$, $SigGen\,(U_j)$, $SigGen\,(U_\ell \to U_j)$).

$SigGen\,(U_0)$. When the companies and organizations delegate a trust user U_0 upload the initial data at the initial time period.

1. The initial user U_0 encodes all the files and then splits them into n block such that each block is in Z_p, *i.e.*, $(m_1, \ldots, m_n) \in (Z_p)^n$.
2. For all $\{m_i\}$, $i \in \{1, \cdots, n\}$ compute the tag of $i-th$ data block m_i as $\sigma_i = (H(W_i)\, u^{m_i})^{x_0}$, $t_i = W_i || Sig_{ssk}(W_i)$, where u is a public parameter chosen randomly from G, $W_i = i||T_j$ and $j \in \{1, \cdots, m\}$.
3. Send the initial verification metadata $V = \{\sigma_i, t_i\}_{1 \le i \le n}$ and the data blocks $\{m_i\}_{1 \le i \le n}$ to the cloud and then deletes them locally.

SigGen (U_j). This algorithm is to generate the block tags of the stored data in the period T_j on the operation $p_{j,k'}, k \in \{1, \cdots, \theta\}, k' \in \{1, \ldots, k\}$.

1. The data blocks is processed as $\{m_i\}_{C_{j,(k'-1)} \le i \le C_{j,k'}}$ by current user U_j, where $C_{j,k'}$ is a positive integer. The increment of the data block by the operations denoted by $p_{j,k'}$, which user U_j will add these data to the cloud in the period T_j.
2. For all $\{m_i\}_{C_{j,(k'-1)}+1 \le i \le C_{j,k'}}$, U_j compute the tag of $i-th$ data block m_i as $\sigma_i = (H(W_i)\, u^{m_i})^{x_j}$, $t_i = W_i || Sig_{ssk}(W_i)$, where u is a public parameter chosen randomly from G, $W_i = i||T_j$ and $j \in \{1, \cdots, m\}$. Send the verification metadata $V = \{\sigma_i, t_i\}_{C_{j,(k'-1)}+1 \le i \le C_{j,k'}}$ and data blocks $\{m_i\}_{C_{j,(k'-1)}+1 \le i \le C_{j,k'}}$ to the cloud and then deletes them locally.

SigGen $(U_\ell \to U_j)$. This algorithm is to generate the block tags of the stored data in the period T_j. If the current user U_j wants to update the data of previous user U_ℓ do.

1. When the current user U_j wants to update the data m_i in the previous T_ℓ period for some reason, the m_i and $V = \{\sigma_i, t_i, C_\ell\}$ should be retrieved firstly.
2. Then the user U_j verified the $t_i = W_i || Sig_{ssk}(W_i)$ with the previous user public key. If wrong the auditing scheme ends, if right the user U_j deals with the data block as m_i and replaced the tag $\sigma_i^{(\ell)} = (H(W_i)\, u^{m_i})^{x_\ell}$ by $\sigma_i^{(j)} = (H(W_i)\, u^{m_i})^{x_j}$ At the same time the user U_j replaced the $t_i = (i||T_\ell)\, ||Sig_{ssk(U_\ell)}(i||T_\ell)$ by $t_i = (i||T_j)\, ||Sig_{ssk(U_j)}(i||T_j)$.
3. At last the user U_j sends verification metadata $V = \{\sigma_i, t_i\}$ and the block m_i to the cloud and then deletes them locally.

Update. If the user U_{j+1} would take the place of the user U_j, then U_{j+1} computes the update key $uk_{j \to j+1}$ as $uk_{j \to j+1} = (g^{x_j})^{\frac{1}{x_{j+1}}} = g^{\frac{x_j}{x_{j+1}}}$, and sends it to the cloud.

The cloud does the following steps:

1. Set $\alpha_j = pk_j$, $\beta_{j+1} = uk_{j \to j+1}$.
2. For any $j \in \{0, \cdots, m-1\}$, let the verification metadata of block data m_i associated with user U_j be $V_i = (\sigma_i, t_i, \alpha_1, \ldots, \alpha_j, \beta_1, \ldots, \beta_j)$.

For the sake of clarity, we list some used signals in Table 1. The protocol is illustrated in Figure 5.

Table 1. Signal and its explanation.

Sig.	Repression
n	the number of the initial data block;
$T_1, T_2, \cdots, T_m$	the period of data manager's management;
T_j	the current period is correspondence the current user U_j;
C	the number of the total data blocks at the auditing time: $C_{j,k} = C_{j-1} + p$ where $p = p_{j,1} + p_{j,2} + \cdots + p_{j,k}$;
$C_1, \cdots, C_m$	the $i-th$ of blocks at the end of period $T_1, T_2, \cdots, T_m$;
$c_1, \cdots, c_m$	the increment of the data block at the end of period $T_1, T_2, \cdots, T_m$;
$p_{j,1}, p_{j,2}, \cdots, p_{j,\theta}$	the increment of the data block by the operation $P_{j,1}, P_{j,2}, \cdots, P_{j,\theta}$ during the period T_j ;
$\sigma_i^{(j)}$	the tag is generated by the U_j and data block m_i ;
Q	the set of index-coefficient pairs, *i.e.*, $Q = \{(i, v_i)\}$;
t	it used to verify if the block i-th match the data block;
V	the response for the challenge Q;

TPA		**The cloud server**
(1) Retrieve file *tag*, verify its signature, and quit if fail; (2) Generate a challenge message challenge: $Q = \{(i, v_i)\}, i \in \left\{1, \ldots, C_{j,k}\right\}$;	$\overrightarrow{Q = \{(i, v_i)\}}$	(3) Spilt this Q to $\left\{Q_{T_0}, \ldots, Q_{T_\ell}, \ldots, Q_{T_j}\right\}$; (4) Compute $\mu' = \sum_{i \in Q} v_i m_i$, $\sigma = \prod_{i \in Q} \sigma_i^{v_i} \in G_1$; (5) compute $r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G_1$, $\mu = \mu' + rh(R) \in Z_p$, and $t = \{t_i\}_{i \in Q}$;
	$\overleftarrow{(\mu, \sigma, t, \alpha, \beta)}$	
(6) Compute: $r = f_{k_3}(challenge) \in Z_p$; verify $PK_{(U_\ell)}\left(Sig_{ssk(U_\ell)}(W_i)\right) = (i \| T_\ell)$, then verify $(\mu, \sigma, t, \alpha, \beta)$ via the verification equation.		

Figure 5. Revocable third-party privacy-preserving auditing protocol.

Challeng. When the U_j wants to verify the integrity of the data block stored in the cloud in the period of T_j (U_j's period), it would send a verity request to the TPA.

ProofGen. When the TPA receives the request of user, it would issue a random set $Q = \{(i, v_i)\}$ and a communication key k_3 to the cloud as a *Challeng*, where $i \in \left\{1, \cdots, C_{j,k}\right\}$ and $v_i \in Z_p$. After receiving *Challeng*, the cloud can spilt this Q to $\left\{Q_{T_0}, \ldots, Q_{T_\ell}, \ldots, Q_{T_j}\right\}$ then computes and returns $(\mu, \sigma, t, \alpha, \beta)$ as a *proof*, where $r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G_1$, $\mu' = \sum_{i \in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$, $\sigma = \prod_{i \in Q} \sigma_i^{v_i} \in G_1$, and $t = \{t_i\}_{i \in Q}$.

ProofVerify. When the TPA receives the *proof*, input the public key pk_ℓ of user U_ℓ, the *Challeng*, $Q = \{(i, v_i)\}$, k_3 and the *proof*, $(\mu, \sigma, t, \alpha, \beta)$, the algorithm outputs VALID to the U_j as the Result if the following equalities simultaneously hold.

First for each t_i verifies $PK_{(U_\ell)}\left(Sig_{ssk(U_\ell)}(W_i)\right) = (i \| T_\ell)$.

Second verifies the data block and tags $e(\sigma, g) = \prod_{\ell \in [0,j]} e\left(\alpha_\ell, u^\mu \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right)$.

Third verifies the user $e(\alpha_{j-1}, g) = e(pk_j, \beta_j)\, e(\alpha_\ell, g) = e(\alpha_{\ell+1}, \beta_\ell)\, for \ell \in \{0, \ldots, j-2\}$.

Remark 1. *The update process of the revocable third-party privacy-preserving auditing scheme is simple and is also efficient in terms of both computation and communication costs because it only needs to compute and send one update key* $uk_{l \to l+1}$.

Remark 2. *There is only one public key,* i.e., *the current user's public key, in the revocable third-party privacy-preserving auditing scheme for any period of time. All public keys of revoked users are not certified any more, and thus a malicious cloud could modify them discretionarily.*

5. Analysis of the Proposed Auditing Scheme

5.1. Correctness

Now we prove the correctness and security of our revocable third-party privacy-preserving auditing scheme.

Theorem 1. *The auditing scheme satisfies correctness.*

Proof. According to the above construction, that for any challenge $Q = \{(i, u_i)\}$, this challenge can be spilt to $Q = \left\{Q_{T_0}, \ \ldots, \ Q_{T_\ell}, \ \ldots, Q_{T_j}\right\}$, $\ell \in \{0, \ \ldots, \ j\}$, have

$$\begin{aligned} & e\left(\sigma, g\right) = e\left(\prod_{i \in Q} \sigma_i^{v_i}, g\right) \\ & = \prod_{\ell \in [0,j]} e\left(\prod_{i \in Q_{T_\ell}} \sigma_i^{v_i}, g\right) \\ & = \prod_{\ell \in [0,j]} e\left(\prod_{i \in Q_{T_\ell}} \left(H\left(W_i\right) u^{m_i}\right)^{v_i}, g^{x_\ell}\right) \\ & = \prod_{\ell \in [0,j]} e\left(\alpha_\ell, u^{\mu - rh(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \\ & = \prod_{\ell \in [0,j]} e\left(\alpha_\ell, u^{\mu} \cdot R^{-h(R)} \cdot \prod_{i \in Q_{T_\ell}} H(W_i)^{v_i}\right) \end{aligned}$$

Also, for any user U_j where $j \in \{1, \ \cdots, \ m\}$, we know that

$$e\left(\alpha_{j-1}, g\right) = e\left(g^{x_{j-1}}, g\right) = e\left(g^{x_j}, g^{\frac{x_{j-1}}{x_j}}\right) = e\left(pk_j, \beta_j\right)$$

and for all $\ell \in \{0, \ \cdots, \ j-2\}$ have

$$e\left(\alpha_\ell, g\right) = e\left(g^{x_\ell}, g\right) = e\left(g^{x_{\ell+1}}, g^{\frac{x_\ell}{x_{\ell+1}}}\right) = e\left(\alpha_{\ell+1}, \beta_\ell\right)$$

Therefore, the auditing scheme is correct. □

5.2. Security Analysis

Theorem 2. *The auditing scheme is secure in the random oracle model under the CDH assumption.*

Proof. According to Definition 2, if there exists a polynomial time adversary A who breaks the scheme with non-negligible probability ϵ, we construct an algorithm B that uses the adversary A as a subroutine to solve a hard CDH problem with probability ϵ too. Algorithm B does so by interacting with A as follows.

Setup. Given a security parameter λ , the algorithm B first randomly picks a generator g of G, $g^\alpha \in G$ and a hash function $H : \{0,1\}^* \to G$ that will be modeled as a random oracle in the proof. B also chooses random g^{x_0} from G for an unknown x_0 as U_0's public key and computes U_j's public key g^{x_j} for all $j \in \{0, \ \cdots, \ m\}$, where x_j is picked from Z_q. Then B sets $u = g^\alpha$ and sends the system parameters g, u and all users' public keys $\{g^{x_j}\}_0^m$ to the adversary A.

Query. The adversary A can query the following types of oracles adaptively. It is assumed that for any data block m_i, A will first make a H-Oracle query on the block before others.

H-Oracle. When A queries the oracle on a data block m_i, B looks up m_i in H-list, an initial empty list with the tuples $(m_i, s_i, H\left(W_i\right))$. If B finds a matched tuple, it outputs $H\left(W_i\right)$ as response. Otherwise B first picks a random value $s_i \in Z_p$ and then computes $H\left(W_i\right) = g^{s_i}/u^{m_i}$, stores $(m_i, s_i, H\left(W_i\right))$ in H-list and finally outputs $H\left(W_i\right)$ as response.

SignGen-Oracle. To get the tags of data blocks $\{m_i\}_{i \in [1, C_{j,k}]}$, A queries the oracle on the file. Upon receiving the query, for all $i \in \left\{1, \ \cdots, \ C_{j,k}\right\}$, $j \in \{1, \ \cdots, \ m\}$, $\ell \in \{0, \ \cdots, \ j\}$, B looks up

m_i in H-list, finds a matched tuple $(m_i, s_i, H(W_i))$, computes $\sigma_i = (g^{x_\ell})^{s_i}$ and finally outputs the set $V = \left(\sigma_1, \ \ldots, \ \sigma_{C_{j,k}}\right)$ as response. Since $\sigma_i = (H(W_i)u^{m_i})^{x_l}$, plugging $H(W_i) = g^{s_i}/u^{m_i}$ into the equality, we can see that $\sigma_i = (g^{x_\ell})^{s_i}$ for all $i \in \left\{1, \ \cdots, \ C_{j,k}\right\}$.

Update-Oracle. If A wants to replace the user U_j with its successor U_{j+1} for some $j \in \{1, \ \cdots, \ m-1\}$, A will query the oracle on U_j. Upon receiving the query, B first computes the update key $uk_{j\to j+1} = g^{xj/x_{j+1}}$ using U_{j+1}'s secret key x_{j+1} and sends the result to A. Then A sets $\alpha_j = g^{x_j}$ and $\beta_{j+1} = uk_{j\to j+1}$, and adds them into $U_{j'}s$ verification metadata. Let $V_j = (\mu, \ \sigma, \ t, \ \alpha_1, \ \ldots, \ \alpha_{j-1}, \ \beta_1, \ \ldots, \ \beta_j)$ be $U_{j'}s$ verification metadata, then we know that U_{j+1}'s verification metadata is $V_{j+1} = (\mu, \ \sigma, \ t, \ \alpha_1, \ \ldots, \ \alpha_j, \ \beta_1, \ \ldots, \ \beta_{j+1})$.

Corrupt-Oracle. Let all revoked users at present be $U_1, \ \ldots, \ U_d$ for some $d \in \{1, \ \cdots, \ m-1\}$. If the adversary A queries the oracle on the user U_j where $j \in \{1, \ \cdots, \ d\}$, then B returns $U_{j'}s$ secret key x_j as response. □

Proof. If B wants to verify whether the data block m stored in the cloud remains intact or not, it will issue a random challenge $Challeng\,(Q, k_3)$, $Q = \{(i, v_i)\}$ to A, where $i \in \left\{1, \ \cdots, \ C_{j,k}\right\}$ and $v_i \in Z_p$. Let $\left(m_1, \ \ldots, \ m_{C_{j,k}}\right) \in (Z_p)^n$ and the current user be U_j. Upon receiving $Challeng(Q, k_3)$, A computes $r = f_{k_3}(challenge) \in Z_p$, $R = u^r \in G_1$, $\mu' = \sum_{i\in Q} v_i m_i$, $\mu = \mu' + rh(R) \in Z_p$, $\sigma = \prod_{i\in Q} \sigma_i^{v_i}$, $t = \{t_i\}_{i\in Q}$, and returns a valid proof $V_j = (\mu, \ \sigma, \ t, \ \alpha_1, \ \ldots, \ \alpha_{j-1}, \ \beta_1, \ \ldots, \ \beta_j)$ to B.

Forgery. A with non-negligible probability ϵ outputs a valid proof $(\mu^*, \ \sigma^*, \ t, \ \alpha_1, \ \ldots, \ \alpha_{j-1}, \ \beta_1, \ \ldots, \beta_j)$ of a $Challeng\,(Q, k_3)$ on a damaged file $\{m_i\}_{i\in[1,C_{j,k}]}$ with respect to user U_j, where $j \in \{1, \ \cdots, \ m\}$.

Let the proof of the $Challeng\,(Q, k_3)$ on the unbroken data blocks $\{m_i\}_{i\in[1,C_{j,k}]}$ with respect to user U_j be (μ, σ), then we know $\mu \neq \mu^*$. Let $\Delta\mu = \mu^* - \mu$ Since $e\,(\sigma, g) = \prod_{\ell\in[0,j]} e\left(\alpha_\ell, u^\mu \cdot R^{-h(R)} \cdot \prod_{i\in Q_{T_\ell}} H(W_i)^{v_i}\right)$ and $e\,(\sigma^*, g) = \prod_{\ell\in[0,j]} e\left(\alpha_0, u^\mu \cdot R^{-h(R)} \cdot \prod_{i\in Q_{T_\ell}} H(W_i)^{v_i}\right)$, (by Definition 2), we have $e(\sigma^* \cdot \sigma^{-1}, g) = e\left(g^{x_0}, u^{\Delta\mu}\right)$. As a result, we know $u^{x_0} = g^{ax_0} = \left(\sigma^* \cdot \sigma^{-1}\right)^{\frac{1}{\Delta\mu}}$. That is, B with probability ϵ solves a CDH problem: given $g, g^a, g^{x_\ell} \in G$,output g^{ax_0}. □

6. Performance Analysis

In this section, we analyze the communication and computation complexities of revocable third-party privacy-preserving auditing scheme for cloud storage. Particularly, we are only interested in the communication and computation costs of its frequent activities, and ignore the costs of the initial system setup that is the same as other conventional public auditing schemes.

NOTATION. Let *Pair* denote one pairing operation, *Exp* denote one exponentiation operation in G, and *MZ* and *MG* respectively denote one multiplication operation in Z_p and G. We denote the bit size of the element in $\left\{1, \ \cdots, \ C_{j,k}\right\}$, $\{1, \ \cdots, \ n\}$, Z_p and G by$|C|$, $|n|$, $|p|$ and $|G|$ respectively. The number of the data blocks selected by a challenge user is assumed to be a constant c.

6.1. Communication Cost

We can see that the communication overhead of our scheme depends on the communication complexity of algorithm Proof. According to the Proof algorithm, the user U_j in one auditing process would first send a challenge $Q = \{(i, v_i)\}$ with size $c\,(|C| + |p|)$ to the cloud and then the cloud would send a proof $(\mu, \ \sigma, \ t, \ \alpha_1, \ \ldots, \ \alpha_{j-1}, \ \beta_1, \ \ldots, \ \beta_j)$ with size $|p| + 2j|G| + c|C| + |G|$ to the user U_j if it's the user $U_{j'}s$ first auditing query; otherwise the cloud would just send (μ, σ, t) with size $|p| + |G| + c|C|$ to the user U_j. Therefore, the total communication cost of one audit process in our scheme is $|p| + |G| + c\,(2\,|C| + |p|)$ bits.

6.2. Computation Cost

The computation cost includes update time and audit time. To update a user U_j, the Update algorithm only needs to compute g^{x_{j-1}/x_j}. Hence the update time of our scheme is Exp. To complete one audit, the cloud should output a proof and the auditing user should verify its correctness. We know that the audit time of our scheme for user U_j depends on the generation and verification costs of (μ, σ, t). Therefore, the audit time for user U_j is $(c+2j)\,MZ + jMG + (2c+j)\,Exp + (j+1)\,Pair$ (here we ignore the simple addition and hash operations).

Additional Comparison. We also give a comparison between our scheme and the revised scheme of [16] for auditing owned cloud storage. Table 2 shows the details of the comparison. We know that the auditing scheme in [16] is insecure under collusion attacks but it's the most efficient revocable public cloud storage auditing scheme in the literature. When a user U_j executes the Proof algorithm of [16], it would send a challenge $Q = \{(i, v_i)\}$ with size $c\,(|n| + |q|)$ to the cloud and the cloud would send a proof $\{\alpha, \beta, \{id_l, s_l\}_{l\in L}\}$ with size $j \cdot (|p| + |G|) + c \cdot |id|$ to the user U_j. Therefore, the total communication cost of one audit process in that scheme is $j \cdot (|p| + |G|) + c \cdot (|id| + |n| + |p|)$ bits. As the Update algorithm of [16] needs to recalculate all the tags of $n$data blocks, we know the update time of [16] is $nExp$. To complete one audit, the scheme in [16] first requests the cloud to output a proof $nExp$ and then instructs the auditing user to verify its correctness. Therefore, we know that the audit time of [16] for any user is $(c+2j)\,MZ + jMG + (c+j)\,Exp + (j+1)\,Pair$ (here the simple addition and hash operations are also ignored). From Table 2, we can see that the communication cost of our scheme will has superior efficiency than the [16] in some cases. And audit time of our scheme is (almost) the same as those of [16], while the update time of [16] is larger than that of our scheme. Therefore we know our scheme is more computationally efficient than the scheme in [16].

Table 2. The comparison of two revocable public cloud storage auditing schemes.

Scheme	Communication Cost	Computation Cost		Collusion Resistance
		Update Time	Audit Time	
[16]	$j \cdot (\|p\| + \|G\|) + c \cdot (\|id\| + \|n\| + \|p\|)$	$nExp$	$(c+2j)\,MZ + jMG + (c+j)\,Exp + (j+1)\,Pair$	NO
Our scheme	$\|p\| + \|G\| + c\,(2\,\|C\| + \|p\|)$	Exp	$(c+2j)\,MZ + jMG + (2c+j)\,Exp + (j+1)\,Pair$	YES

6.3. Experimental Results

As we know, the comparison of computation cost is obvious. Our Update time is Exp , it is much lower than the update time of [16]: $nExp$. Our auditing time is approximately equal the scheme in [16], it is only a difference of $cExp$. So we only need compare the communication cost of our auditing scheme with the work of [16] in experiments. Our experiments are implemented on a windows 7 system with an Intel Core 2 i5 CPU running at 2.53 GHz, 2 GB DDR 3 of RAM (1.74 GB available). All algorithms are implemented by C language, and our code uses the MIRACL library version 5.6.1. The elliptic curve we use is an MNT curve, the base field size is 159 bits and the embedding degree is 6. The security level is chosen to be 80 bit, and $|p| = |q| = 160$. For simplicity, we also set $k = 20$, $c = 300$. All the results of experiments are represented as the average of 30 trials. As described in Figure 6, the experimental results show that, compared with the auditing scheme in [16], the communication cost of our auditing scheme are much light-weight than the scheme in [16].

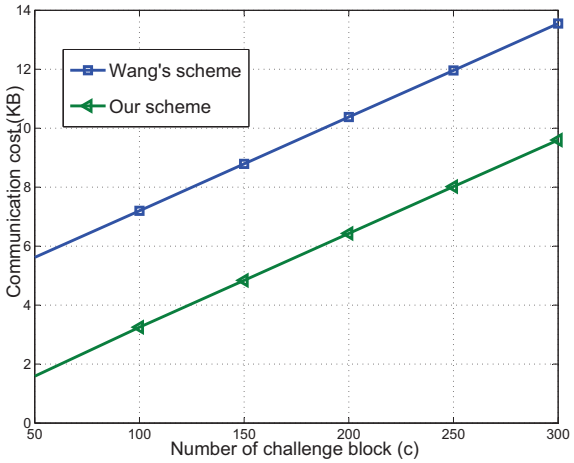

Figure 6. Comparison on the communication cost between our scheme and the scheme in [16].

7. Conclusions

In this paper, we have investigated the efficient user revocation problem in public cloud storage auditing systems and have proposed a dynamic revocable third-party privacy-preserving auditing scheme for cloud storage. We have proved that our scheme is secure against collusion attacks and have also demonstrated its effectiveness. In the light of the simplicity and extensibility of revocable third-party privacy-preserving auditing scheme for cloud storage, we believe the scheme would be much applicable in real-world cloud storage auditing systems.

Acknowledgments: This work is supported by the National Natural Science Foundation of China (No. 61370203) and the Science and Technology on Communication Security Laboratory Foundation(Grant No. 9140C110301110C1103).

Author Contributions: Theory: Xinpeng Zhang and Chunxiang Xu; Math analysis: Xinpeng Zhang and Xiaojun Zhang; Simulations: Xinpeng Zhang; Interpretation: Xinpeng Zhang and Xiaojun Zhang; Writing: Xinpeng Zhang, Taizong Gu, Zhi Geng and Guoping Liu. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. 9 worst cloud security threats. Available online: http://www.informationweek.com/byte/cloud/infrastructure-as-a-service/9-worst-cloud-security-threats/d/d-id/1114085 (accessed on 25 May 2016).
2. Cloud Security Alliance. Top Hreats to Cloud Computing. Available online: http://www.cloudsecurityalliance.org (accessed on 25 May 2016).
3. Kincaid, J. Mediamax/Helinkup Close Its Doors. Available online: http://techcrunch.com/2008/07/10/mediamaxthelinkup-closes-its-doors/ (accessed on 25 May 2016).
4. Cloud Computing Users Are Losing Data, Symantec Finds. Available online: http://news.investors.com/technology/011613-640851-cloud-computing-data-loss-high-in-symantec-study.htm (accessed on 25 May 2016).
5. Kher, V.; Kim, Y. Securing distributed storage: Challenges, techniques, and systems. In Proceedings of the 2005 ACM Workshop on Storage Security and Survivability, Fairfax, VA, USA, 11 November 2005; pp. 9–25.
6. Schroeder, B.; Gibson, G.A. Disk failures in the real world: What does an mttf of 1, 000, 000 hours mean to you? In Proceedings of the FAST '07: 5th USENIX Conference on File and Storage Technologies, San Jose, CA, USA, 14–16 Februery 2007; Volume 7, pp. 1–16.
7. Cloud Security Alliance. Cloud computing vulnerability incidents: a statistical overview. Available online: http://www.cert.uy/wps/wcm/connect/975494804fdf89eaabbdab1805790cc9/CloudComputing-Vulnerability-Incidents. pdf/?MOD=AJPERES (accessed on 25 May 2016).

8. Yu, S.; Wang, C.; Ren, K.; Lou, W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In Proceedings of the 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
9. Wang, C.; Chow, S.S.M.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375.
10. Wang, Q.; Wang, C.; Li, J.; Ren, K.; Lou, W. Enabling public verifiability and data dynamics for storage security in cloud computing. In *Computer Security—ESORICS 2009*; Springer: Medford, MA, USA, 2009; pp. 355–370.
11. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 598–609.
12. Juels, A.; Kaliski, B.S., Jr. Pors: Proofs of retrievability for large files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 584–597.
13. Shacham, H.; Waters, B. Compact proofs of retrievability. In *Advances in Cryptology-ASIACRYPT 2008*; Springer: Medford, MA, USA, 2008; pp. 90–107.
14. Li, M.; Yu, S.; Ren, K.; Lou, W. Securing personal health records in cloud computing: Patient-centric and fine-grained data access control in multi-owner settings. In *Security and Privacy in Communication Networks*; Springer: Medford, MA, USA, 2010; pp. 89–106.
15. Zhang, X.; Xu, C.; Zhang, X. Efficient pairing-free privacy-preserving auditing scheme for cloud storage in distributed sensor networks. *Int. J. Distrib. Sens. Netw.* **2015**, *501*, 593759,
16. Wang, B.; Li, B.; Li, H. Panda: Public auditing for shared data with efficient user revocation in the cloud. *IEEE Trans. Serv. Comput.* **2015**, *8*, 92–106.
17. Wang, B.; Li, H.; Li, M. Privacy-preserving public auditing for shared cloud data supporting group dynamics. In Proceedings of the 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 1946–1950.
18. Yuan, J.; Yu, S. Efficient public integrity checking for cloud data sharing with multi-user modification. In Proceedings of the IEEE Conference on Computer Communications (INFOCOM), Toronto, ON, Canada, 27 April–2 May 2014; pp. 2121–2129.
19. Delerablée, C.; Paillier, P.; Pointcheval, D. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing-Based Cryptography—Pairing 2007*; Springer: Medford, MA, USA, 2007; pp. 39–59.
20. Chaum, D.; Van Heyst, E. Group signatures. In *Advances in Cryptology EUROCRYPT 91*; Springer: Medford, MA, USA, 1991; pp. 257–265.
21. Wang, B.; Li, B.; Li, H. Knox: Privacy-preserving auditing for shared data with large groups in the cloud. In *Applied Cryptography and Network Security*; Springer: Medford, MA, USA, 2012; pp. 507–525.
22. Ateniese, G.; Hohenberger, S. Proxy re-signatures: New definitions, algorithms, and applications. In Proceedings of the 12th ACM conference on Computer and Communications Security, Alexandria, VA, USA, 7–10 November 2005; pp. 310–319.
23. Liu, Q.; Wang, G.; Wu, J. Efficient sharing of secure cloud storage services. In Proceedings of the 2010 IEEE 10th International Conference on Computer and Information Technology (CIT), Bradford, UK, 29 June–1 July 2010; pp. 922–929.
24. Liu, Q.; Wang, G.; Wu, J. Time-based proxy re-encryption scheme for secure data sharing in a cloud environment. *Inf. Sci.* **2014**, *258*, 355–370.