IEEE TRANSACTIONS ON COMPUTERS VOL: PP NO: 99 YEAR 2015

Public Integrity Auditing for Shared Dynamic Cloud Data with Group User Revocation

Tao Jiang, Xiaofeng Chen, and Jianfeng Ma

Abstract—The advent of the cloud computing makes storage outsourcing become a rising trend, which promotes the secure remote data auditing a hot topic that appeared in the research literature. Recently some research consider the problem of secure and efficient public data integrity auditing for shared dynamic data. However, these schemes are still not secure against the collusion of cloud storage server and revoked group users during user revocation in practical cloud storage system. In this paper, we figure out the collusion attack in the exiting scheme and provide an efficient public integrity auditing scheme with secure group user revocation based on vector commitment and verifier-local revocation group signature. We design a concrete scheme based on the our scheme definition. Our scheme supports the public checking and efficient user revocation and also some nice properties, such as confidently, efficiency, countability and traceability of secure group user revocation. Finally, the security and experimental analysis show that, compared with its relevant schemes our scheme is also secure and efficient.

Index Terms—Public integrity auditing, dynamic data, victor commitment, group signature, cloud computing.

1 Introduction

The development of cloud computing motivates enterprises and organizations to outsource their data to third-party cloud service providers (CSPs), which will improve the storage limitation of resource constrain local devices. Recently, some commercial cloud storage services, such as the simple storage service (S3) [1] on-line data backup services of Amazon and some practical cloud based software Google Drive [2], Dropbox [3], Mozy [4], Bitcasa [5], and Memopal [6], have been built for cloud application. Since the cloud servers may return an invalid result in some cases, such as server hardware/software failure, human maintenance and malicious attack [7], new forms of assurance of data integrity and accessibility are required to protect the security and privacy of cloud user's data.

To overcome the above critical security challenge of today's cloud storage services, simple replication and protocols like Rabin's data dispersion scheme [8] are far from practical application. The formers are not practical because a recent IDC report suggests that data-generation is outpacing storage availability [9]. The later protocols ensure the availability of data when a quorum of repositories, such as $k\text{-}out\text{-}of\text{-}n$ of shared data, is given. However, they do not provide assurances about the availability of each repositories, which will limit the assurance that the protocols can provide to relying parties.

For providing the integrity and availability of remote cloud store, some solutions [10], [11] and their variants [12], [13], [14], [15], [16], [17], [18] have been proposed. In these solutions, when a scheme supports data modification, we call it *dynamic* scheme, otherwise *static* one (or limited dynamic scheme, if a scheme could only efficiently support some specified operation, such as append). A scheme is *publicly verifiable* means that the data integrity check can be performed not only by data owners, but also by any third-party auditor. However, the dynamic schemes above focus on the cases where there is a data owner and only the data owner could modify the data.

Recently, the development of cloud computing boosted some applications [19], [20], [21], where the cloud service is used as a collaboration platform. In these software development environments, multiple users in a group need to share the source code, and they need to access, modify, compile and run the shared source code at any time and place. The new cooperation network model in cloud makes the remote data auditing schemes become infeasible, where only the data owner can update its data. Obviously, trivially extending a scheme with an online data owner to update the data for a group is inappropriate for the data owner. It will cause tremendous communication and computation overhead to data owner, which will result in the single point of data owner. To support multiple user data operation, Wang et al. [22] proposed a data integrity based on ring signature. In the scheme, the user revocation problem is not considered and the auditing cost is linear to the group size and data size. To further enhance the previous scheme and support group user revocation, Wang et al. [23] designed a scheme based on proxy re-signatures. However, the scheme assumed that the private and authenticated channels exist between each

• *Tao Jiang, Xiaofeng Chen and Jianfeng Ma are with the State Key Laboratory of Integrated Service Networks(ISN), Xidian University, P.R. China, e-mail: jiangt2009@gmail.com, xfchen@xidian.edu.cn, and jfma@mail.xidian.edu.cn.*

pare of entities and there is no collusion among them. Also, the auditing cost of the scheme is linear to the group size. Another attempt to improve the previous scheme and make the scheme efficient, scalable and collusion resistant is Yuan and Yu [24], who designed a dynamic public integrity auditing scheme with group user revocation. The authors designed polynomial authentication tags and adopt proxy tag update techniques in their scheme, which make their scheme support public checking and efficient user revocation. However, in their scheme, the authors do not consider the data secrecy of group users. It means that, their scheme could efficiently support plaintext data update and integrity auditing, while not ciphertext data. In their scheme, if the data owner trivially shares a group key among the group users, the defection or revocation any group user will force the group users to update their shared key. Also, the data owner does not take part in the user revocation phase, where the cloud itself could conduct the user revocation phase. In this case, the collusion of revoked user and the cloud server will give chance to malicious cloud server where the cloud server could update the data as many time as designed and provide a legal data finally. To the best of our knowledge, there is still no solution for the above problem in public integrity auditing with group user modification.

The deficiency of above schemes motivates us to explore how to design an efficient and reliable scheme, while achieving secure group user revocation. To the end, we propose a construction which not only supports group data encryption and decryption during the data modification processing, but also realizes efficient and secure user revocation. Our idea is to apply vector commitment scheme [25] over the database. Then we leverage the Asymmetric Group Key Agreement (AGKA) [26] and group signatures [27] to support ciphertext data base update among group users and efficient group user revocation respectively. Specifically, the group user use the AGKA protocol to encrypt/decrypt the share database, which will guarantee that a user in the group will be able to encrypt/decrypt a message from any other group users. The group signature will prevent the collusion of cloud and revoked group users, where the data owner will take part in the user revocation phase and the cloud could not revoke the data that last modified by the revoked user.

1.1 Our Contribution

In this paper, we further study the problem of construing public integrity auditing for shared dynamic data with group user revocation. Our contributions are three folds:

1) We explore on the secure and efficient shared data integrate auditing for multi-user operation for ciphertext database.

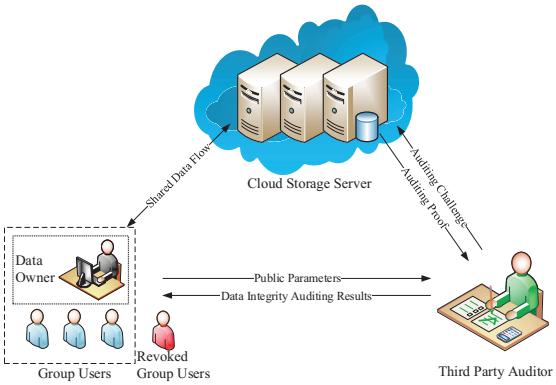

Figure 1. The cloud storage model

2) By incorporating the primitives of victor commitment, asymmetric group key agreement and group signature, we propose an efficient data auditing scheme while at the same time providing some new features, such as traceability and countability.
3) We provide the security and efficiency analysis of our scheme, and the analysis results show that our scheme is secure and efficient.

1.2 Organization

The rest of this paper is organized as follows: In section 2, we describe the problem formulation. In section 3, we present the used preliminaries. Then, we provide the detail of our scheme in section 4. We conduct the security and efficiency analysis in section 5 and section 6 and leave the related works in section 7. Finally, we show our conclusion in section 8.

2 Problem Formulation

In this section, we first describe the cloud storage model of our system. Then, we provide the threat model considered and security goals we want to achieve.

2.1 Cloud Storage Model

In the cloud storage model as shown in Figure 1, there are three entities, namely the cloud storage server, group users and a Third Part Auditor (TPA).

Group users consist of a data owner and a number of users who are authorized to access and modify the data by the data owner. The cloud storage server is semi-trusted, who provides data storage services for the group users. TPA could be any entity in the cloud, which will be able to conduct the data integrity of the shared data stored in the cloud server. In our system, the data owner could encrypt and upload its data to the remote cloud storage server. Also, he/she shares the privilege such as access and modify (compile and execute if necessary) to a number of group users. The TPA could efficiently verify the integrity of the

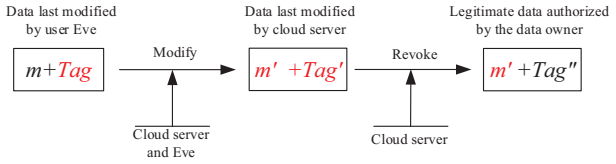

Figure 2. Security problem of server proxy group user revocation

data stored in the cloud storage server, even the data is frequently updated by the group users. The data owner is different from the other group users, he/she could securely revoke a group user when a group user is found malicious or the contract of the user is expired.

2.2 Threat Model and Security Goals

Our threat model considers two types of attack:

1) An attacker out side the group (include the revoked group user cloud storage server) may obtain some knowledge of the plaintext of the data. Actually, this kind of attacker has to at lease break the security of the adopted group data encryption scheme.
2) The cloud storage server colludes with the revoked group users, and they want to provide a illegal data without being detected.

Actually, in cloud environment, we assume that the cloud storage server is semi-trusted. Thus, it is reasonable that a revoked user will collude with the cloud server and share its secret group key to the cloud storage server. In this case, although the server proxy group user revocation way [24] brings much communication and computation cost saving, it will make the scheme insecure against a malicious cloud storage server who can get the secret key of revoked users during the user revocation phase. Thus, a malicious cloud server will be able to make data m, last modified by a user that needed to be revoked, into a malicious data m'. In the user revocation process, the cloud could make the malicious data m' become valid. To overcome the problems above, we aim to achieve the following security goals in our paper:

1) ***Security.*** A scheme is secure if for any database and any probabilistic polynomial time adversary, the adversary can not convince a verifier to accept an invalid output.
2) ***Correctness***. A scheme is correct if for any database and for any updated data m by a valid group user, the output of the verification by an honest cloud storage server is always the value m. Here, m is a ciphertext if the scheme could efficiently support encrypted database.
3) ***Efficiency***. A scheme is efficient if for any data, the computation and storage overhead invested by any client user must be independent of the size of the shared data.
4) ***Countability***. A scheme is countable, if for any data the TPA can provide a proof for this misbehavior, when the dishonest cloud storage server has tampered with the database.
5) ***Traceability***. We require that the data owner is able to trace the last user who update the data (data item), when the data is generated by the generation algorithm and every signature generated by the user is valid.

3 Preliminaries

Our scheme makes use of bilinear groups. The security of the scheme depends on the Strong Diffie-Hellman assumption and the Decision Linear assumption. In this section, we review the definitions of bilinear groups and the complexity assumption.

3.1 Bilinear Groups

We review a few concepts related to bilinear maps, which follow the notation of [28]. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p, g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$. ψ is an efficiently computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$ with $\psi(g_2) = g_1$, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map with the following properties:

1) **Computability:** there exits an efficiently computable algorithm for computing map e;
2) **Bilinearity:** for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$;
3) **Non-degeneracy:** $e(g_1, g_2) \neq 1$.

3.2 Complexity Assumption

The security of our scheme relies on the difficulty of some problems: the Strong Diffie-Hellman problem, the Decision Linear problem, and the Computational Diffie-Hellman problem. We describe these problems as follows.

Definition 1. q-Strong Diffie-Hellman problem. Let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic group of prime order p, where possibly $\mathbb{G}_1 = \mathbb{G}_2$. Let g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Given a $(q+2)-tuple(g1, g2, g_2^{\gamma}, g_2^{(\gamma^2)}, ..., g_2^{(\gamma^q)})$ as input, output a pair $(g_1^{1/(\gamma+x)}, x)$ where $x \in \mathbb{Z}_p^*$.

The assumption could be used to construct short signature scheme without random oracles [29]. The assumption has properties similar to the Strong-RSA assumption [30] and the properties are adopted for building short group signature in our scheme.

Definition 2. Decision Linear problem. Let g_1 be a generator of $\mathbb{G}_1$, and $\mathbb{G}_1$ be a cyclic group of prime order p. Given u, v, h, u^a, u^b,$u^c \in \mathbb{G}_1$ as input, output yes if $a + b = c$ and no otherwise.

Boneh et al. [31] introduced the Decision Linear assumption and they proved that the problem is intractable in generic bilinear groups.

Definition 3. Square Computational Diffie-Hellman (Square-CDH) problem. With $g \in \mathbb{G}_1$ as above, given (g, g^x) for $x \in_R \mathbb{Z}_p$ as input, output g^{x^2}.

It has been proved that the Square-CDH assumption is equivalent to the classical CDH assumption [32], [33].

3.3 Vector Commitment

Commitment is a fundamental primitive in cryptography and it plays an important role in security protocols such as voting, identification, zero-knowledge proof, etc. The hiding property of commitment requires that it should not reveal information of the committed message, and the binding property requires that the committing mechanism should not allow a sender to change his/her mind about the committed message.

Recently, Catalano and Fiore [25] put forward a new primitive called Vector Commitment. Vector Commitment satisfies position binding that an adversary should not be able to open a commitment to two different values at the same position, and the Vector Commitment is concise, which means that the size of the commitment string and its openings have to be independent of the vector length. We provide the formal definition of Vector Commitment [25] as follows.

Definition 4. (Vector Commitment) A vector commitment scheme is a collection of six polynomial-time algorithms (VC.KeyGen, VC.Com, VC.Open, VC.Ver, VC.Update, VC.ProofUpdate) such that:

$\mathsf{VC.KeyGen}(1^k, q)$. Given the security parameter k and the size q of the committed vector (with $q = poly(k)$), the key generation outputs some public parameters pp.

$\mathsf{VC.Com}_{\mathsf{pp}}(m_1, ..., m_q)$. On input a sequence of q messages $m_1, ..., m_q \in \mathcal{M}$ ($\mathcal{M}$ is the message space) and the public parameters pp, the committing algorithm outputs a commitment string C and an auxiliary information aux.

$\mathsf{VC.Open}_{\mathsf{pp}}(m, i, \mathsf{aux})$. This algorithm is run by the committer to produce a proof i that m is the i-th committed message. In particular, notice that in the case when some updates have occurred the auxiliary information aux can include the update information produced by these updates.

$\mathsf{VC.Ver}_{\mathsf{pp}}(C, m, i, \Lambda_i)$. The verification algorithm accepts (i.e., it outputs 1) only if Λ_i is a valid proof that C was created to a sequence $m_1, ..., m_q$ such that $m = m_i$.

$\mathsf{VC.Update}_{\mathsf{pp}}(C, m, m', i)$. This algorithm is run by the committer who produces C and wants to update it by changing the i-th message to m'. The algorithm takes as input the old message m, the new message m' and the position i. It outputs a new commitment C' together with an update information U.

$\mathsf{VC.ProofUpdate}_{\mathsf{pp}}(C, \Lambda_j, m', i, U)$. This algorithm can be run by any user who holds a proof Λ_j for some message at position j w.r.t. C, and it allows the user to compute an updated proof Λ'_j (and the updated commitment C') such that Λ'_j will be valid with regard to C' which contains m' as the new message at position i. Basically, the value U contains the update information which is needed to compute such values.

The primitive of verifiable database with efficient update based on vector commitment is useful to solve the problem of verifiable data outsourcing. Recently, Chen et al. [34], [35] figured out that the basic vector commitment scheme suffers from forward automatic update attack and backward substitution update attack. They also proposed a new framework for verifiable database with efficient update from vector commitment, which is not only public verifiable for dynamic outsourced data but also secure against the two attacks. The solution in their scheme is easy to apply in our scheme, which will overcome the attacks they figured out in our scheme.

3.4 Group Signature with User Revocation

We present the formal definition of group signatures with verifier-local revocation [27] as follows.

Definition 5. A verifier-local group signature scheme is a collection of three polynomial-time algorithms $(\mathsf{VLR.KeyGen}, \mathsf{VLR.Sign}, \mathsf{VLR.Verify})$, which behaves as follows:

$\mathsf{VLR.KeyGen}(n)$. This randomized algorithm takes as input a parameter n, the number of members of the group. It outputs a group public key gpk, an n-element vector of user keys $gsk = (gsk[1], gsk[2], ..., gsk[n])$, and an n-element vector of user revocation tokens grt, similarly indexed.

$\mathsf{VLR.Sign}(gpk, gsk[i], M)$. This randomized algorithm takes as input the group public key gpk, a private key $gsk[i]$, and a message $M \in \{0,1\}^*$, and returns a signature σ.

$\mathsf{VLR.Verify}(gpk, RL, \sigma, M)$. The verification algorithm takes as input the group public key gpk, a set of revocation tokens RL (whose elements form a subset of the elements of grt), and a purported signature σ on a message M. It returns either valid or invalid. The latter response can mean either that σ is not a valid signature, or that the user who generated it has been revoked.

4 Scheme Construction

In this section, we provide the formal definition of our scheme according to the definition in [23], [24]. Then, we design the concrete scheme based on our definition.

4.1 New Framework

We consider the database DB as a set of tuple (x, m_x), where x is an index and m_x is the corresponding value. Informally, a public integrity auditing scheme with updates allows a resource-constrained client to outsource the storage of a very large database to a remote server. Later, the client can retrieve and update the database records stored in the server and publicly audit the integrity of the updated data.

According to previous researches, the proposed framework of our public integrity auditing for shared dynamic cloud data with secure group user revocation is given as follows:

Setup($1^k, DB$):

Let the database be $DB = (i, m_i)$ for $1 \leq i \leq q$ and the database is shared by a group of n users with only one data owner.

1) The data owner run the key generation algorithm of vector commitment to obtain the public parameters $\mathsf{pp} \leftarrow \mathsf{VC.KeyGen}(1^k, q)$.
2) Run the key generation of verifier-local revocation to obtain the user keys and revocations (***gsk***, ***grt***) $\leftarrow \mathsf{VLR.KeyGen}(1^k, n)$, where ***gsk*** $= (gsk[1], gsk[2]...gsk[n])$ and an n-element vector of user revocation tokens ***grt***.
3) Run the computing algorithm to compute commitment and auxiliary information $(C, \mathsf{aux}) \leftarrow \mathsf{VC.Com_{pp}}(c_1, ..., c_q)$. Let the current database modifier be group user $s(0 \leq s \leq n-1)$, and $(gsk[s], gpk)$ be the secret/public key pair of the group user. Let $C^t = VC.Com_{\mathsf{PP}}(c_1^t, ..., c_q^t)$ be the commitment on the latest database vector, where t is a counter with 0 as its initial value.
4) Run the signing algorithm over the commitment C. Specially, for the t-th time the group user $s(0 \leq s \leq n-1)$, whose secret key is $gsk[s]$, compute and output a signature $\sigma^t \leftarrow \mathsf{VLR.Sign}(gpk, gsk[s], \{C(t-1), C^t, t\})$. Then, sends the signature σ^t to the cloud storage server. If σ^t is valid, then the server computes $C(t) = \sigma^t \cdot C^t$. Also, the cloud storage server adds the information of $\Sigma(t) = (C(t-1), C^t, t, \sigma^t)$ to aux.
5) Finally, set public key parameter $PK = (\mathsf{pp}, gpk, C(t-1), C(t))$.

Query($PK, \mathsf{PP}, \mathsf{aux}, DB, i$):

1) A group user run the opening algorithm to compute a proof $\Lambda_i \leftarrow \mathsf{VC.Open_{pp}}(c_i, i, \mathsf{aux})$, where Λ_i is the proof of the i-th committed message and return $\tau = (c_i, \Lambda_i, \Sigma(t))$.

Verify(PK, RL, i, τ):

1) Parse $\tau = (c_i, \Lambda_i, \Sigma(t))$. If the signature is valid after running the algorithm $\mathsf{VLR.Verify}(gpk, RL, \Sigma(T))$. Then, run the verification algorithm of vector commitment $\{0,1\} \leftarrow \mathsf{VC.Ver_{pp}}(C(t), \sigma^t, c_i, i, \Lambda_i)$. The algorithm accepts when it output 1, which means that Λ_i is a valid proof that C^t was created by a sequence $c_1, ...c_q$, such that $c = c_i$. Otherwise, return an error $\perp$.

Update(i, τ):

1) A group user first queries and verifies the database to make sure the current database is valid. More precisely, the group user obtain $\tau \leftarrow$**Query(**$PK, \mathsf{PP}, \mathsf{aux}, DB, i$**)** and check that **Verify**$(PK, i, \tau) = m_i$.
2) Run the update algorithm over the new data and output the updated commitment and the update information $(C', U) \leftarrow \mathsf{VC.Update}(C, m, m', i)$.

ProofUpdate(C, Λ_j, c_i', i, U):

1) A third part auditor can first verify that, compared with the stored counter t, the latest counter equals $t+1$. Then, run the proof of update algorithm of vector commitment to compute an update proof $\Lambda_j \leftarrow \mathsf{VC.ProofUpdate_{pp}}(C, \Lambda_j, m_i', i, U)$ for the message at position j, such that Λ_j is valid with respect to C' which contains m' as the new message at position j. Here, $U = (m, m', i)$ is the update information.
2) Verify the commitment C', and its corresponding proof Λ_i is also valid over message m_i'.

UserRevocation(PK, i, τ):

1) The third part auditor can run the verification algorithm of verifier-local revocation and return either valid or invalid $\{0,1\} \leftarrow \mathsf{VLR.Verify}(gpk, RL, \sigma, M)$. Here, RL are a set of revocation tokens.

4.2 A Concrete Scheme

In this section, we provide a concrete scheme from vector commitment [25] and verifier-local revocation group signature [27].

Setup($1^k, DB$):

Let k be a security parameter and $DB = (i, m_i)$ for $1 \leq i \leq q$ be the database. The database $DB = (i, m_i)$ is shared by a group of n users with only one data owner. The message space is $\mathcal{M} = \mathbb{Z}_p$.

1) Let $\mathbb{G}, \mathbb{G}_T$ be two bilinear groups of prime order p equipped with a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, and g be a random generator of $\mathbb{G}$. Randomly choose $z_1, ..., z_q \leftarrow_R \mathbb{Z}_p$. For all $i = 1, ..., q$, set $h_i = g^{z_i}$. For all $i, j = 1, ..., q, i \neq j$, set $h_{i,j} = g^{z_i z_j}$. The data owner runs the key generation algorithm of vector commitment $\mathsf{VC.KeyGen}(1^k, q)$ to obtain the public parameters $\mathsf{PP} = (p, q, G, G_T, \mathcal{H}, g, (\{h_i\})_{i \in [q]}, \{h_{i,j}\}_{i,j \in [q], i \neq j})$ and the message space $\mathcal{M} = \mathbb{Z}_p$. By using a collision-resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, our scheme can be easily extended to support arbitrary messages in $\{0,1\}^*$.
2) Run the key generation of verifier-local revocation $\mathsf{VLR.KeyGen}(1^k, n)$. Let $\mathbb{G}_1$, $\mathbb{G}_2$ be cyclic

group of prime order p, and g_1 be a generator of $\mathbb{G}_1$ and g_2 be a generator of $\mathbb{G}_2$. Consider bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ with isomorphism ψ, where $g_1 \leftarrow \psi(g_2)$. Select $\gamma \leftarrow_R \mathbb{Z}_p^*$ and set $w = g_2^{\gamma}$. For each user, generate an SDH tuple (A_i, x_i) by selecting $x_i \overset{R}{\leftarrow} \mathbb{Z}_p^*$ such that $\gamma + x_i \neq 0$, and setting $A_i \leftarrow g_1^{1/(\gamma+x_i)}$. Then, set the group public key $gpk = (g_1, g_2, w)$. The private key is a tuple $gsk[i] = (A_i, x_i)$. The revocation token corresponding to a user's secret key is $grt[i] = A_i$. Finally, the algorithm outputs (gpk, ***gsk***, ***grk***). γ is only known to the private-key issuer (the data owner).

3) Run the computing algorithm $\mathsf{VC.Com_{pp}}(m_1, ..., m_q)$ to compute commitment $C = h_1^{m_1}, h_2^{m_2}, ..., h_q^{m_q}$ and auxiliary information $\mathsf{aux} = (m_1, ..., m_q)$.
4) Employ hash functions H_0 and H as random oracles, with respective ranges $\mathbb{G}_2^2$ and $\mathbb{Z}_p$. For the t-th time data updating, run the signing algorithm $\mathsf{VLR.Sign}(gpk, gsk[i], \{C(t-1), C^t, t\})$ over the commitment. Assume that the input message is $\{C(t-1), C^t, t\} \in \{0,1\}^*$. Then, pick a random nonce $r \leftarrow_R \mathbb{Z}_p$ and obtain generators $(\hat{u}, \hat{v}) \leftarrow H_0(gpk, \{C(t-1), C^t, t\}, r) \in \mathbb{G}_2^2$ and compute their images in $\mathbb{G}_1$ with $u \leftarrow \psi(\hat{u})$ and $v \leftarrow \psi(\hat{v})$. Select an exponent $\alpha \leftarrow_R \mathbb{Z}_p$ and compute $T_1 \leftarrow u^{\alpha}$ and $T_2 \leftarrow A_i v^{\alpha}$. Set $\delta \leftarrow x_i \alpha \in \mathbb{Z}_p$. Pick blinding values r_α, r_x, and $r_\delta \leftarrow_R \mathbb{Z}_p$. Compute helper values $R_1 \leftarrow u^{r_\alpha}$, $R_2 \leftarrow e(T_2, g_2)^{r_x} \cdot e(v, w)^{-r_\alpha} \cdot e(v, g_2)^{-r_\delta}$ and $R_3 \leftarrow T_1^{r_x} \cdot u^{-r_\delta}$. Compute a challenge value $c \leftarrow H(gpk, (C(t-1), C^t, t), r, T_1, T_2, R_1, R_2, R_3) \in \mathbb{Z}_p$ using H. Compute $s_\alpha = r_\alpha + c\alpha$, $s_x = r_x + cx_i$, and $s_\delta = r_\delta + c\delta \in \mathbb{Z}_p$. Finally, output a signature $\sigma^t \leftarrow (r, T_1, T_2, c, s_\alpha, s_x, s_\delta)$. Then, sends the signature σ^t to the cloud storage server. If σ^t is valid, then the server computes $C(t) = \sigma^t \cdot C^t$. Also, the cloud storage server adds the information of $\Sigma(t) = (C(t-1), C^t, t, \sigma^t)$ to aux.
5) Set public key parameter $PK = (\mathsf{pp}, gpk, C(t-1), C(t))$.

Query($PK, \mathsf{pp}, \mathsf{aux}, DB, i$)**:**

1) We assume that the current public key is $PK = (\mathsf{PP}, gpk, C(t-1), C(t))$. A user runs the opening algorithm $\mathsf{VC.Open_{pp}}(c_i^t, i, \mathsf{aux})$ to compute a proof $\Lambda_i^t = \prod_{j-1, j\neq i}^{q} h_{i,j}^{m_j^t} = (\prod_{j=1, j\neq i}^{q} h_j^{m_j^t})^{z_i}$ of the i-th committed message and return $\tau = (m_i^t, \Lambda_i^t, \Sigma(t))$.

Verify(PK, i, τ)**:**

1) On input a group public key gpk, a purported signature σ^t, and the message $\{C(t-1), C^t, t\}$, the auditor first verify whether the signature is valid.
2) If $\tau = (m_i^t, \Lambda_i, \Sigma(t))$, run the verification algorithm of vector commitment $\mathsf{VC.Ver_{pp}}(C_i^t, c_i^t, i, \Lambda_i^t)$ to verify that the equation $e(C^t / h_i^{m_i^t}, h_i) \overset{?}{=} e(\Lambda_i^t, g)$ holds. The algorithm accepts when it outputs 1, which means that Λ_i^t is a valid proof that C^t was created to a sequence $m_1, ...m_q$, such that $m = m_i$.

Update(i, τ)**:**

1) A group user first queries and verifies the database to make sure the current database is valid.
2) If the user wanted to update m_i to m_i', the user runs the update algorithm $\mathsf{VC.Update}(C, m, m', i)$ and outputs the updated commitment $C' = C \cdot h_i^{m'-m}$ and the updated information $U = (m, m', i)$.

ProofUpdate(C, Λ_j, m_i', i, U)**:**

1) The third part auditor can run the proof of update algorithm of vector commitment to compute an update proof $\Lambda_j \leftarrow \mathsf{VC.ProofUpdate_{pp}}(C, \Lambda_j, m', i, U)$ for the message at position j, such that Λ_j is valid with respect to C' which contains m' as the new message at position j.
2) For the auditor who owns a proof Λj, the auditor uses the update information $U = (m, m', i)$ to generate the proof of update. If $i \neq j$, compute the updated commitment $C' = C \cdot h_i^{m'-m}$ and the updated proof is $\Lambda_j' = \Lambda_j \cdot (h_i^{m'-m})^{z_j} = \Lambda_j \cdot h_{j,i}^{m'-m}$; If $i = j$, compute the updated commitment $C' = C \cdot h_i^{m'-m}$ while do not change the proof Λ_i. Verify the commitment C' and its corresponding proof Λ_i is also valid over message m_i'.

UserRevocation(PK, i, τ)**:**

1) To verify the validity of the signature, the auditor need to conduct the signature check. The third part auditor runs the verification algorithm of verifier-local revocation $\mathsf{VLR.Verify}(gpk, RL, \sigma, M)$, $M = C(t-1), C^t, t$. More precisely, compute $\hat{u}$ and $\hat{v}$ and their image $u \leftarrow \psi(\hat{u})$ and $v \leftarrow \psi(\hat{v})$ in $\mathbb{G}_1$. Derive $\tilde{R}_1 \leftarrow u^{s_\alpha}/T_1^c$, $\tilde{R}_2 \leftarrow e(T_2, g_2)^{s_x} \cdot e(v, w)^{-s_\alpha} \cdot e(v, g_2)^{-s_\delta} \cdot (e(T_2, w)/e(g_1, g_2))^c$ and $\tilde{R}_3 \leftarrow T_1^{s_x} \cdot u^{-s_\delta}$. Check the challenge that $c \overset{?}{=} H(gpk, (C(t-1), C^t, t), r, T_1, T_2, \tilde{R}_1, \tilde{R}_2, \tilde{R}_3)$ and return either valid or invalid. Then, conduct the revocation check.
2) For each element $A \in RL$, check whether A is encoded in (T_1, T_2) by checking if $e(T_2/A, \hat{u}) \overset{?}{=} e(T_1, \hat{v})$. If no element of RL is encoded in (T_1, T_2), the signer of σ has not been revoked. Here, RL is a set of revocation tokens.

4.3 Supporting Ciphertext Database

In cloud storage outsourcing environment, the outsourced data is usually encrypted database, which is usually implicitly assumed in the exiting academic

research. Actually, our scheme could support the auditing of database of both plaintext and ciphertext database. However, it is not straightforward to extend a scheme to support encrypted database.

In order to achieve the confidentiality of the data record m_x, the client can use his/her secret key to encrypt each m_x using a encryption scheme. When there is only one user (data owner) in the group, the user only needs to choose a random secret key and encrypt the data using a secure symmetric encryption scheme. However, when the scheme needs to support multi-user data modification, while at the same time keeping the shared data encrypted, a shared secret key among group users will result in single point failure problem. It means that any group user (revoked or leave) leak the shared secret key will break the confidentiality guarantee of the data.

To overcome the above problem, we need to adopt a scheme, which could support group users data modification. Luckily, Wu et al. [26] designed an Asymmetric Group Key Agreement scheme (ASGKA). The scheme has a nice property that, instead of a common secret key, only a shared encryption key is negotiated in an ASGKA protocol. Also, in the scheme, the public key can be simultaneously used to verify signatures and encrypt messages while any signature can be used to decrypt ciphertext under this public key. Using the bilinear pairings, the authors instantiate a one-round ASGKA protocol tightly reduced to the decision Bilinear Diffie-Hellman Exponentiation (BDHE) assumption in the standard model. Thus, according to the ASGKA protocol, we consider the case of encrypted database (x, c_x), where x is an index and c_x is the corresponding cipher value.

We provide the detailed changes upon our scheme to support encrypted database.

1) In the **Setup** phase, the scheme has to run the key agreement of ASGKA for the group users. Then, the database $DB = (i, m_i)$ is encrypted by the group key gpk of data owner. Finally, the stored database is a ciphertext database $DB = (i, c_i)$.
2) In the second step of the **Update** phase, a group user firstly decrypts the record c_i using the ASGKA secret key $gsk[*]$ to get plaintext database $DB = (i, m_i)$. Then, update the data to m_i', and later encrypt the data with the public key gpk of ASGKA scheme to get the new encrypted database $DB = (i, c_i')$.

4.4 Probabilistic Detection

Actually, the position binding property of vector commitment of the scheme allows the cloud storage server to prove the data item correctness of certain position. Ateniese et. al. [10] figured out that the sampling ability greatly reduces the overhead on the server and provides high detection probability of server misbehavior. Then, among the q data items, we assume that the third part auditor randomly select x items out of the q-block item database as the target item. In the database, only y items of the database are incorrect. Then, if x, y and q satisfy the specific relationship, the third part auditor could provide a high possession detection ability over the database. The result is interesting that when y is a fraction of the total item number q, the detection probability of server misbehavior is a constant amount of item. For example, if y = 1% of q, then the third part auditor asks for 460 blocks and 300 blocks in order to achieve the detection probability of at least 99% and 95%, respectively.

5 Analysis of Our Scheme

Our scheme is designed to solve the security and efficiency problems of public data integrity auditing with multi-user modification, where the data has to be encrypted among a dynamic group and any group user can conduct secure and verifiable data update when necessary.

Some basic tools have been used to construct our scheme. Thus we assume that the underlying building blocks are secure, which include the vector commitment, group signature, and asymmetric group key agreement scheme. Based on this assumption, we show that our scheme is secure with respect to the following security analysis.

5.1 Security of Our Scheme

- **Security of Our Scheme.** Actually, Wu et al. [26] instantiated a one-round ASGKA scheme tightly reduced to the decision Bilinear Diffie-Hellman Exponentiation assumption in the standard model. Also, the security of adopted group signature scheme has been proved to be secure in the random oracle model. The security of the scheme is based on the strong Diffie-Hellman assumption and the Decision Linear assumption in bilinear groups as defined in Definition 1 and Definition 2. Thus, if we assume the two building blocks of our scheme is secure, then our scheme can be proven to be secure similar to [25].
  If we assume there exists a polynomial-time adversary $\mathcal{A}$ that has a non-negligible advantage ϵ in the experiment for some initial database. Then we can use the adversary to build an efficient algorithm to break the Squ-CDH assumption in Definition 3. It means that the algorithm takes a tuple g, g^a as input and output g^{a^2}. Trivially, suppose that $(\hat{i}, \hat{\tau})$ in a experiment where $\tau = (c_i^t, \Lambda_i^t, \Sigma(t))$, the verify query output a value $\hat{c} \neq \perp$, $\hat{c} \neq c_i^t$ and $e(C^t, h_{\hat{i}}) = e(h_{\hat{i}}^{c_{\hat{i}}^t}, h_{\hat{i}}) e(\Lambda_{\hat{i}}^t, g) = e(h_{\hat{i}}^{\hat{c}}, h_{\hat{i}}) e(\hat{\Lambda}, g)$. If the simulation does not fail, we have $h_{\hat{i}} = g^a$. Then, the adversary can compute

Table 1
Performance evaluation and comparison

Scheme	Scheme [23]	Scheme [24]	Our Scheme
Query	-	-	$(q-1)(\mathbf{Mul}+\mathbf{Exp})$
Verify	$2\mathbf{Exp}+\mathbf{Mul}+2\mathbf{Pair}+\mathbf{Hash}$	$\mathbf{Exp}+2\mathbf{Pair}r$	$7\mathbf{Pair}+\mathbf{Mul}+9\mathbf{Exp}+5\mathbf{Hash}$
Update	-	$(s+2)\mathbf{Exp}+(s+1)\mathbf{Mul}$	$s(\mathbf{Mul}+\mathbf{Exp})^*$
ProofUpdate	-	$s\mathbf{Exp}+(s+1)\mathbf{Mul}+c\mathbf{Pair}$	$2s(\mathbf{Mul}+\mathbf{Exp})ors(\mathbf{Mul}+\mathbf{Exp})^*$
UserRevocation	$(c+d)\mathbf{Exp}+(c+3d)\mathbf{Mul}+(d+1)\mathbf{Pair}+c\mathbf{Hash}$	$c(\mathbf{Pair}+\mathbf{Exp})$	$z(\mathbf{Mul}+2\mathbf{Pair})$

* In our scheme, we do not need to verify up to s elements each time. For comparison, we assume that our scheme conducts a verification of s data items here.

$g^{a^2} = (\frac{\hat{\Lambda}}{\Lambda_i^t})^{(c_i^t-\hat{c})^{-1}}$ and the success probability of the algorithm built by the adversary is ϵ/q.

- **Correctness of Our Scheme**. If the server is assumed to be honest, then the proof $\tau = (c_i^t, \Lambda_i^t, \Sigma(t))$, where $\Lambda_i^t = \prod_{i\neq j,1\leq j\leq q} h_{i,j}^{c_j^t}$. Since $C(t)/H_t h_i^{c_i^t} = C^t/h_i^{c_i^t} = \prod_{i\neq j,1\leq j\leq q} h_{i,j}^{c_j^t}$, we have $e(C_t/H_t h_i^{c_i^t}, h_i) = e(\Lambda_i^t, g)$. Thus, the verification algorithm always output c_i^t.
- **Efficiency of Our Scheme.** It is trivial that, except for the one time setup, the computational and storage overhead in our scheme invested by the group users are independent of the size of the data. More precisely, to verify the validity of the scheme, the verify algorithm run by the client requires only pairings and exponentiation in $\mathbb{G}$. Also, in the update algorithm, the computation overhead of the client is independent of the size of data. The storage overhead of a group user is also independent of the size of data. We will provide the detail efficiency and experiment analysis in the full version of this paper.
- **Countability of Our Scheme.** Sine the update counter t is a public parameter, given the proof with the counter t', the client will firstly compare it with the public latest counter. if $t' = t$, then the auditor verifies the corresponding signature $\sigma^{t'}$ over t'. Otherwise, if $t' \neq t$ the group auditor will reject the current result and report the malicious activity of the cloud storage server.
- **Traceability of Our Scheme.** The traceability of our scheme is based on the traceability of the adopted group signature. In the theorem 2 of reference [25], the authors provide the formal proof of the traceability of the group signature adopted. It means that if SDH is hard on ($\mathbb{G}_1$,$\mathbb{G}_2$), the group signature scheme is traceable.

6 Performance Evaluation

In this section, we provide both the numerical and the experimental analysis of our scheme and conduct the computation time cost comparison with [23] and [24].

6.1 Numerical Analysis

In this section, we conduct the numerical analysis of our scheme and compare the scheme with references [23] and [24].

First of all, all of the three schemes require one-time expensive computational effort in the **Setup** phase. Then, our scheme is secure against the collusion attack of the cloud storage server and the revoked users in the efficient scheme [24], and it is also efficient, since the computational resources invested by the client is independent on the size of the database. The reason is that, most of the expensive computation overhead is outsourced to the cloud storage server. Finally, the cloud storage server store all the database and its relevant materials. Thus, except some private key materials, the group users do not require to store any data locally.

We provide the time cost simulation for our scheme in different phases and the Table 1 presents the numerical analysis of computation of our scheme and two other schemes related. For the convenience of analysis, we denote by **Mul** a multiplication in $\mathbb{G}$ ($\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$), **Exp** an exponentiation in $\mathbb{G}$, **Pair** a computation of the pairing, and **Hash** a regular hashing operation. We omit other operations such as addition in $\mathbb{G}$ for all the schemes.

As shown in Table 1, in the **Query** algorithm of our scheme, the computation overhead increases with the database item q. However, we need to remark that the server does not need to compute the proof each time. The reason is that the proof is identical for the same data item and the server only need to compute once for the first query on each index. Thus, the server could adopt some storage overhead to reduce the computational cost in the **Query** algorithm. Compare with scheme [24], the **Verify** algorithm of our scheme bring much more computation overhead. The reason is that scheme [24] adopt the delegation technology for data updating. In our scheme, to prevent the attack against the collusion of the malicious and revoked group users, we adopt the group signature scheme with secure group user revocation. Although the **Verify** algorithm bring much more computational overhead than scheme [24], it is important that it is

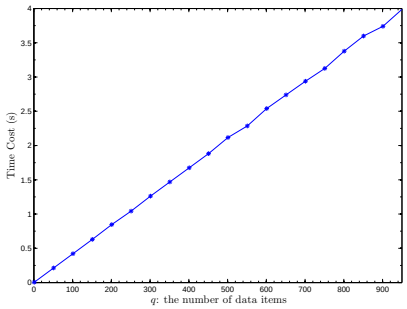

Figure 3. Query Time Cost

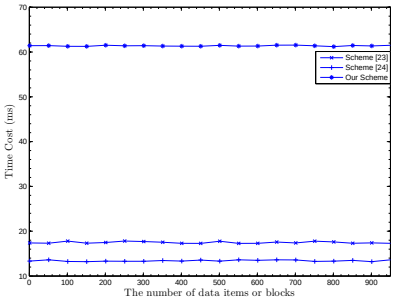

Figure 4. Verify Time Cost

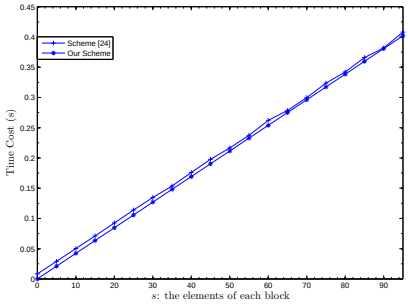

Figure 5. Update Time Cost

a constant part of our scheme. In the **Update** and **ProofUpdate** algorithms, the computation cost of our scheme and the scheme [24] grow with the increase of data elements (data items in our scheme). For the **UserRevocation** algorithm, all the computation computation time cost grow with the increase of the challenge blocks (items) number. The different is that, scheme [24] is efficient because the computation time cost will not grow with the increase of the group users. Thus, their scheme provide constant computation overhead with different group size. The computation time cost of our scheme grows with the revoked users number z, which is different from the scheme [23] growing with the increase of the group users number. Since it is reasonable that the revoked users number z is small than the group users d, we use $2z = d$ in our simulation.

6.2 Experimental evaluation

In this section, we evaluate the thorough experimental evaluation of our scheme. Our experiments are simulated with the pairing-based cryptography library (PBC)[36] on Linux Machine with Intel® Core™2 Duo Processor T9500 running at 2.60GHz and 3G memory. To precisely evaluate the computation complexity at different entities, we simulate all the entity on this machine.

As shown in Figure 3, the **Query** time cost of our scheme is linear with the data items number q, which will take approximately 4 seconds to query about 1000 data items. However, we need to emphasize that the computation cost is at the cloud storage server side, which is very powerful compare with the Linux system running on our laptop. More over, the server does not need to run the whole **Query** algorithm every time as analyzed in the previous section.

Actually, in the **Verify** algorithm, the computation overhead mostly comes from the group signature scheme. More precisely, to verify the validity of this phase, we need firstly to verify the integrity of the signature, which means that our scheme need to generate the time costed parameters such as R_1, R_2, and $R3$. Actually, the computation time cost of our scheme a constant number. Also, it is around 5 times that of the most efficient scheme [24]. In the Figure 5, we show the data update computation comparison with scheme [24], and both their computation overheads grow with the increase of the element number in each block.

In the **PoofUpdate** algorithm, as shown in Figure 6, the computation time cost grows with the increase of the elements number of each block and the number of selected challenging data blocks. Actually, the data blocks contain data elements in scheme [24] while not in our scheme. It is interesting that, if we do not consider the data elements in the scheme [24], the computation overhead of the two schemes in with the same challenging number are almost the same.

The **UserRevocation** algorithm simulation in Figure 7 shows that scheme [24] is the most efficient one. Compare with scheme [23] whose computation overhead grows rapidly with the increase of group users number and the selected challenging blocks number, scheme [24] and our scheme have a flatting growth. The reason is that, scheme [23] has to consider the whole group users number, while the computation time cost in our scheme is related to the revoked group users. It means that we need to verify $e(T_2/A, \hat{u}) \stackrel{?}{=} e(T_1, \hat{v})$ for each user in the revocation list. The best scheme is [24], whose computation overhead is irrelevant to the group users number. They achieve this by allowing the cloud storage server to recompute the authentication tag of blocks last modified by a revoked group user. We analyze this tag update delegation way in the previous section and point out that it is not secure against the cloud storage server and revoked group users collusion attack.

7 Related Work

Plenty of researchers have devoted considerable attention to the problems on how to securely outsource local store to remote cloud server. Among which, the problem of remote data integrity and availability auditing attacks the attestation of many researchers. The concepts and solution Provable Data Possession (PDP) and Proofs of Retrievability (PoR) were first proposed by Ateniese et al. [10] and Juels et al. [11]. In

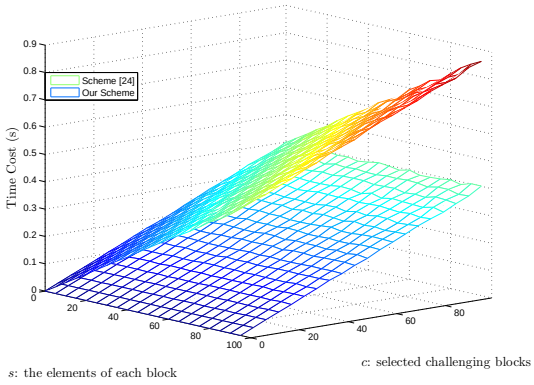

Figure 6. ProofUpdate Time Cost

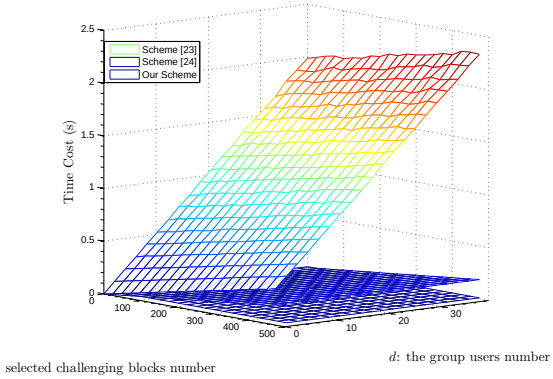

Figure 7. UserRevocation Time Cost

their scheme, the homomorphic authentication technique was adopted to reduce both the communication and computation cost. Later, a number of variants of PDP and PoR schemes are designed to improve the efficiency and enhance the function of basic schemes, such as allowing public auditing [16], [22], [17] and supporting data update [14], [15].

To enhance the previous works, Wang et al. [22] designed a scheme to support share data integrity auditing, whose scheme adopted ring signature to protect the privacy of users. The limitation of the scheme is that it does no support dynamic group and also suffers from a computational overhead linear to the group size and the number of data auditing. To further support user revocation, Wang et al. [23] designed another scheme based on the assumption that no collusion occurs between cloud servers and revoked user. As a matter of fact, they assumed that the private and authenticated channels exit between each pair of entities and collusion between invalid users and cloud servers will lead to the disclosure of secrets of all other valid users. Recently, Yuan and Yu [24] designed a dynamic public integrity auditing scheme with secure group user revocation. The scheme is based on polynomial authentication tags and adopts proxy tag update techniques, which makes their scheme support public checking and efficient user revocation. However, the authors do not consider the ciphertext store. Also, to make the scheme efficient, the data owner (the data owner's private key is not necessary) does not take part in the user revocation phase, where the cloud could conduct some malicious operation of user's data when it colludes with the revoked users.

Gennaro et al. [37] formalized the notion of verifiable computation which allows a client to outsource the computation of an arbitrary function. However, it is inefficient for practical applications due to the complicated fully homomorphic encryption techniques [38], [39]. Also, another disadvantage of the schemes based on fully homomorphic encryption is that, the client must repeat the expensive pre-processing stage if the malicious server tries to cheat and learn a bit of information. Benabbas et al. [40] proposed the first practical verifiable database scheme based on the hardness of the subgroup membership problem in bilinear groups with composite order. However, the scheme does not support the public verifiability property. Catalano and Fiore [25] proposed a practical solution to build verifiable database (VDB) from vector commitment that supports the public verifiability. Both of the schemes assume that the size of the outsourced database should be fixed and the client can know the outsourcing function in advance. Recently, Backes et al. [41] presented a flexible VDB scheme with two additional properties that eliminates the assumption.

Group signature is introduced by Chaum and Heyst [42]. It provides anonymity for signers, where each group member has a private key that enables the user to sign messages. However, the resulting signature keeps the identity of the signer secret. Usually, there is a third party that can conduct the signature anonymity using a special trapdoor. Some systems support revocation [43], [44], [45], [27], [46], [47], where group membership can be disabled without affecting the signing ability of unrevoked users. Boneh and Shacham [27] proposed an efficient group signature with verifier-local revocation. The scheme provides the properties of group signature such as selfless-anonymity and traceability. Also, the scheme is a short signature scheme where user revocation only requires sending revocation information to signature verifiers. Libert et al. [46] proposed a new scalable revocation method for group signature based on the broadcast encryption framework. However, the scheme introduces important storage overhead at group user side. Later, Libert et al. [47] designed a scheme to enhance the former scheme which could obtain private key of constant size. In their scheme, the unrevoked members still do not need to update their keys at each revocation.

8 Conclusion

The primitive of verifiable database with efficient updates is an important way to solve the problem of verifiable outsourcing of storage. We propose a scheme to realize efficient and secure data integrity auditing for share dynamic data with multi-user modification. The scheme vector commitment, Asymmetric Group Key Agreement (AGKA) and group signatures with user revocation are adopt to achieve the data integrity auditing of remote data. Beside the public data auditing, the combining of the three primitive enable our scheme to outsource ciphertext database to remote cloud and support secure group users revocation to shared dynamic data. We provide security analysis of our scheme, and it shows that our scheme provide data confidentiality for group users, and it is also secure against the collusion attack from the cloud storage server and revoked group users. Also, the performance analysis shows that, compared with its relevant schemes, our scheme is also efficient in different phases.

9 Acknowledgment

This work is supported by the National Natural Science Foundation of China (No. 61272455), China 111 Project (No. B08038), Doctoral Fund of Ministry of Education of China (No. 20130203110004), Program for New Century Excellent Talents in University (No. NCET-13-0946), and the Fundamental Research Funds for the Central Universities (Nos. BDY151402 and JB142001-14).

References

[1] Amazon. (2007) Amazon simple storage service (amazon s3). Amazon. [Online]. Available: http://aws.amazon.com/s3/
[2] Google. (2005) Google drive. Google. [Online]. Available: http://drive.google.com/
[3] Dropbox. (2007) A file-storage and sharing service. Dropbox. [Online]. Available: http://www.dropbox.com/
[4] Mozy. (2007) An online, data, and computer backup software. EMC. [Online]. Available: http://www.dropbox.com/
[5] Bitcasa. (2011) Inifinite storage. Bitcasa. [Online]. Available: http://www.bitcasa.com/
[6] Memopal. (2007) Online backup. Memopal. [Online]. Available: http://www.memopal.com/
[7] M. A. et al., "Above the clouds: A berkeley view of cloud computing," *Tech. Rep. UCBEECS*, vol. 28, pp. 1–23, Feb. 2009.
[8] M. Rabin, "Efficient dispersal of information for security," *Journal of the ACM (JACM)*, vol. 36(2), pp. 335–348, Apr. 1989.
[9] J. G. et al. (2006) The expanding digital universe: A forecast of worldwide information growth through 2010. IDC. [Online]. Available: Whitepaper
[10] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. of ACM CCS*, Virginia, USA, Oct. 2007, pp. 598–609.
[11] A. Juels and B. S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proc. of ACM CCS*, Virginia, USA, Oct. 2007, pp. 584–597.
[12] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: theory and implementation," in *Proc. of CCSW 2009*, llinois, USA, Nov. 2009, pp. 43–54.
[13] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. of TCC 2009*, CA, USA, Mar. 2009, pp. 109–127.
[14] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Proofs of retrievability via hardness amplification," in *Proc. of ESORICS 2009*, Saint-Malo, France, Sep. 2009, pp. 355–370.
[15] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. of ACM CCS*, Illinois, USA, Nov. 2009, pp. 213–222.
[16] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. of IEEE INFOCOM 2010*, CA, USA, Mar. 2010, pp. 525–533.
[17] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability and constant communication cost in cloud," in *Proc. of International Workshop on Security in Cloud Computing*, Hangzhou, China, May 2013, pp. 19–26.
[18] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in *Proc. of ACM CCS 2013*, Berlin, Germany, Nov. 2013, pp. 325–336.
[19] Cloud9. (2011) Your development environment, in the cloud. Cloud9. [Online]. Available: https://c9.io/
[20] Codeanywhere. (2011) Online code editor. Codeanywhere. [Online]. Available: https://codeanywhere.net/
[21] eXo Cloud IDE. (2002) Online code editor. Cloud IDE. [Online]. Available: https://codenvy.com/
[22] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Proc. of IEEE CLOUD 2012*, Hawaii, USA, Jun. 2012, pp. 295–302.
[23] B. Wang, L. Baochun, and L. Hui, "Public auditing for shared data with efficient user revocation in the cloud," in *Proc. of IEEE INFOCOM 2013*, Turin, Italy, Apr. 2013, pp. 2904–2912.
[24] J. Yuan and S. Yu, "Efficient public integrity checking for cloud data sharing with multi-user modification," in *Proc. of IEEE INFOCOM 2014*, Toronto, Canada, Apr. 2014, pp. 2121–2129.
[25] D. Catalano and D. Fiore, "Vector commitments and their applications," in *Public-Key Cryptography - PKC 2013*, Nara, Japan, Mar. 2013, pp. 55–72.
[26] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement," in *Proc. of EUROCRYPT 2009*, Cologne, Germany, Apr. 2009, pp. 153–170.
[27] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proc. of ACM CCS*, DC, USA, Oct. 2004, pp. 168–177.
[28] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. of Asiacrypt 2001*, Gold Coast, Australia, Dec. 2001, pp. 514–532.
[29] D. Boneh and X. Boyen, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. of EUROCRYPT 2004*, Interlaken, Switzerland, May 2004, pp. 56–73.
[30] N. Baric and B. Pfitzman, "Collision-free accumulators and fail-stop signature schemes without trees," in *Proc. of EUROCRYPT 1997*, Konstanz, Germany, May 1997, pp. 480–494.
[31] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Proc. of CRYPTO 2004*, CA, USA, Aug. 2004, pp. 41–55.
[32] U. M. Maurer and S. Wolf, "Diffie-hellman oracles," in *Proc. of CRYPTO 1996*, CA, USA, Aug. 1996, pp. 268–282.
[33] F. Bao, R. Deng, and H. Zhu, "Variations of diffie-hellman proble," in *Information and Communications Security*, Huhehaote, China, Oct. 2003, pp. 301–312.
[34] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," in *Proc. of ESORICS 2014*, Wroclaw, Poland, Sep. 2014, pp. 148–162.
[35] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," to appear in IEEE Transactions on Dependable and Secure Computing, Accepted.
[36] B. Lynn. (2006) The pairing-based cryptography library. [Online]. Available: http://crypto.stanford.edu/pbc/
[37] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. of CRYPTO 2010*, CA, USA, Sep. 2010, pp. 465–482.
[38] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. of ACM STOC 2009*, Washington DC, USA, May 2009, pp. 169–178.

[39] C. Gentry and S. Halevi, "Implementing gentrys fully-homomorphic encryption scheme," in *Proc. of EUROCRYPT 2011*, Tallinn, Estonia, May 2011, pp. 129–148.
[40] S. Benabbas, R. Gennaro, and Y. Vahlis, "Verifiable delegation of computation over large datasets," in *Proc. of CRYPTO 2011*, CA, USA, Aug. 2011, pp. 111–131.
[41] M. Backes, D. Fiore, and R. M. Reischuk, "Verifiable delegation of computation on outsourced data," in *Proc. of ACM CCS 2013*, Berlin, Germany, Nov. 2013, pp. 863–874.
[42] D. Chaum and E. van Heyst, "Group signatures," in *Proc. of EUROCRYPT 1991*, Brighton, UK, Apr. 1991, pp. 257–265.
[43] E. Bresson and J. Stern, "Efficient revocation in group signatures," in *Public-Key Cryptography - PKC 2001*, Cheju Island, Korea, Feb. 2001, pp. 190–206.
[44] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Proc. of CRYPTO 2002*, CA, USA, Aug. 2002, pp. 61–76.
[45] G. Ateniese, D. Song, and G. Tsudik, "Quasi-efficient revocation in group signatures," in *Proc. of FC 2002*, Soughamption, Bermuda, Mar. 2002, pp. 183–197.
[46] B. Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," in *Proc. of EUROCRYPT 2012*, CA, USA, Aug. 2002, pp. 61–76.
[47] ——, "Group signatures with almost-for-free revocation," in *Proc. of CRYPTO 2012*, CA, USA, Aug. 2012, pp. 571–589.

Tao Jiang received his B.S. (2009) in Network Engineering from Shandong Jianzhu University, China. He got his M.S. (2012) in Computer Application Technology from Jiangsu University, China. Currently, he is a Ph.D. student of Xidian University in Cryptography. His research interests include cryptography and cloud computing security.

Xiaofeng Chen received his B.S. and M.S. on Mathematics from Northwest University, China in 1998 and 2000, respectively. He got his Ph.D degree in Cryptography from Xidian University in 2003. Currently, he works at Xidian University as a professor. His research interests include applied cryptography and cloud computing security. He has published over 100 research papers in refereed international conferences and journals. His work has been cited more than 1800 times at Google Scholar. He is in the Editorial Board of Computing and Informatics (CAI), International Journal of Grid and Utility Computing (IJGUC), and International Journal of Embedded Systems (IJES) etc. He has served as the program/general chair or program committee member in over 30 international conferences.

Jianfeng Ma received his B.S. degree in mathematics from Shaanxi Normal University, China in 1985, and obtained his M.E. and Ph.D. degrees in computer software and communications engineering from Xidian University, China in 1988 and 1995, respectively. From 1999 to 2001, he was with Nanyang Technological University of Singapore as a research fellow. Now he is a Professor in School of Computer Science at Xidian University, China. His current research interests include distributed systems, computer networks, and information and network security.