Public Auditing Scheme for Shared Data in the Cloud Storage

Xueyan Liu, Xiaomei He, Tingting Lu and Yukun Luo

School of Mathematics and Statistics, Northwest Normal University, 967 Anning East Road, Lanzhou, Gansu, China.
Email: liuxy@nwnu.edu.cn; 1532678921@qq.com

Abstract. Cloud storage is widely used by both individual users and organizational users due to its efficient and scalable data storage services. Because of the vulnerabilities of hardware/software, human error and malicious attacks, the audit of data integrity in cloud has become the focus of research. However, how to efficiently check the integrity of cloud data and protect the confidentiality of shared data and anonymity of user identity, is still a sharp problem. In this paper, we propose a privacy-preserving mechanism to reduce the security risks in existing schemes, which enables public auditing on shared cloud data. Specifically, identity-based signature is introduced in our scheme, which effectively prevents collusion attack from forgery of signatures. At the same time, the auditors cannot obtain the identity of the signer on shared data. In addition, multiple auditing tasks can be accomplished at one time rather than one by one in our scheme, thus, the efficiency of verification is greatly improved. The theoretical analyses shows that our scheme is correct, secure and privacy protection, and our experiments show that our scheme can efficiently achieve integrity audit of cloud data.

1. Introduction

Cloud data storage has rapid developed, unlike storing data in local storage systems, individuals and businesses store large amounts of data to remote cloud servers, thus the burden of data maintenance and management can greatly be reduced. However, because of the vulnerabilities of hardware/software, human errors and malicious attacks, data stored in cloud is easily lost or damaged, the integrity of data stored in cloud storage is questioned and reviewed. Cloud service providers may be reluctant to inform users about these data errors in order to maintain the reputation of their service and avoid losing profits, but the situation is getting worse, some sensitive data that users do not frequently access may be deleted by CSP. Because losing physical control of share data, users do not know whether their outsourced data is properly stored on cloud servers. Therefore, how to ensure the integrity of data on a remote cloud server becomes a urgent issue for cloud storage services. Ateniese etc. [1] proposed the first auditing scheme. In the scheme, it is not necessary to downloading entire documents from a remote cloud server while the auditor audit the integrity of data. In order to ensure the integrity of data stored on cloud servers, a series of solutions have been proposed based on various techniques [5, 6, 7, 8]. In their scheme, homomorphic technique is introduced to aggregate the sample blocks and corresponding signatures to reduce communication costs, and the auditing process is executed by data owner, but it didn’t take public audit of group user into consideration.

However, public auditing maybe leak data owner identity, especially the third party auditor (TPA) and other data users may obtain the identity information of the data owner. Thus, the frequently audited data may experience many attacks and attract numerous attackers. Therefore, Wang etc. proposed privacy-preserving public auditing schemes, such as Oruta [2] and Knox [3], Oruta is the first scheme for privacy-preserving, ring signature and homomorphic authentication technique have been exploited to support public auditing and protect the anonymity of signer. But in Oruta,
communication and computation cost linearly increase with the number of group users, it cannot
against the attack of group member changing. To further improve its efficiency, Knox was proposed,
which implements group signature to assure user identity anonymity from the TPA. Unfortunately, the
communication efficiency and computational efficiency of Knox are relatively low, and it cannot
support public auditing. Therefore, Yuan etc. [4] proposed a scheme supporting anti-collusion, public
auditing and user update, but it cannot support multi-data blocks batch auditing. To solve the problem,
Wu et al. [9] proposed a public auditing scheme based on homomorphic message authentication code,
which supports multi-block auditing, but the communication cost and computational costs linearly
increase relative to the number of group users. To enrich public auditing, some schemes [10-12] have
been proposed relying on signature or support dynamic groups of public auditing [13], however, their
common shortcomings are leakage identity privacy and low auditing efficiency.

In this work, we aim at the problems mentioned above and propose a dynamic group privacy
protection public batch auditing scheme based on identity signature, introducing users identity
information into user public-private key pairs and adding random values selected by users to prevent
attackers from forging signatures by using false information, and support group user revocation and
data dynamic operation. Moreover, our scheme offers batch checking operation for multiple
documents based on aggregation signature. At the same time, our scheme is achievable batch auditing
tasks, and to have improved the efficiency of auditing.

The paper is organized as follows. The system model and threat model have been presented in
Section 2. In Section 3, cryptography primitives have been introduced. In Section 4, the concrete
construction has been provided, and security analysis in Section 5. The performance evaluate has been
showed in Section 6. In Section 7, we make a conclusion about our work.

2. Problem Statement

2.1. System Model

We consider a cloud system composed of four major entities as shown in Figure1: a trust authority
(TA), the third party auditor (TPA), the cloud server provider (CSP) and group users.

TA: TA is a full-trust entity, it generates system master key, public parameters, users enroll and
users private/public key pairs.

TPA: TPA is a full-trust entity, it checks the integrity of data stored in cloud, but it can’t learn any
data during the auditing.

CSP: CSP is a semi-trust entity, it provides storage and retrieve service, and in a variety of motives
to return the wrong data, resulting in the disclosure of confidential data.

Users: Group users consist of one manage user and group users, the manage user initially creates
shared data in cloud and enroll group users, group users can access data.

![Figure 1. System model diagram](image-url)
2.2. Threat Model
In our paper, CSP is semi-trust entity, it obey rules, but it maybe return the wrong message, concealing the fact that data has been attacked or deleted may or may not occur many times. Data loss is mostly claimed by users not the CSP. Therefore, in this work we consider the following factors that may impact data integrity: (1) system administrator hardware/software failures and operational errors, (2) corrupt to external adversaries that data stored in the cloud, (3) revocation of users who no longer have data access privilege but attempt to illegally modify data.

2.3. Design Objectives
Our mechanism should implement following features: (1) Public Auditing: a public auditor can publicly verify the integrity of shared data without the need to retrieve the entire data from the cloud. (2) Correctness: a public auditor can verify the integrity of the shared data correctly. (3) Unforgeable: only users in the group can generate valid verification on the shared data. (4) Identity Privacy: during the audit process, the public auditor cannot distinguish the identity of the signer on each block in shared data. (5) Batch Auditing: public auditing can satisfy one-time validation of multiple blocks.

3. Preliminaries

3.1. Bilinear Maps [14]:
Definition: Let \( G_1, G_2 \) be two multiplicative cyclic groups of prime order \( p \), \( g \) be generator of \( G_1 \). A bilinear map is a map \( e : G_1 \times G_2 \rightarrow G_2 \) with the following properties:

- **Bilinearity:** for all \( \forall a, b \in Z_p^* \) and \( g_1, g_2 \in G_1 \), \( e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} \).
- **Non-degeneracy:** \( e(g_1, g_2) \neq 1 \).
- **Computability:** there exists an efficiently computable algorithm for computing map \( e \).

3.2. Security Assumptions
The security of the proposed scheme depends on the following assumptions:

Definition1: CDH (Computational Diffie-Hellman) assumption, given three elements \( g, g^a, g^b \in G \), there is no polynomial algorithm to calculate the value of \( g^{ab} \) that cannot be ignored in probability, where \( a, b \) are random selection from group \( Z_p \).

Definition2: DL (Discrete Logarithm) assumption, given two element \( g, g^a \in G(a \in Z_p) \), it is computationally infeasible to compute the value of \( a \) and \( g \) is a random point in group \( G \).

4. The Proposed Scheme
In this section, we describe our approach.

**Setup.** Input the security parameter \( k \), \( G_1 \) and \( G_2 \) are multiplicative cyclic groups of prime order \( p \), \( g \) is generator of \( G_1 \), randomly pick \( g_1 \in G_1 \). Let \( e : G_1 \times G_2 \rightarrow G_2 \) be a bilinear map, there is a hash function \( H : \{0,1\}^* \rightarrow \{0,1\}^l \in G_1 \). There are \( d \) group users registered through group manager, \( I = \{1, \ldots, d\} \). Shared data \( M \) is divided into \( n \) blocks as \( M = \{m_1, \ldots, m_n\} \), \( j = 1, \ldots, n \). TA randomly picks \( \theta_j \), \( \alpha \in Z_q^* \), the master keys \( MK = \{\alpha, \{\alpha_i\}_{i=1}^d\} \), computes the system public keys are \( T = g^n \), \( u_j = g^{\frac{1}{m_j}} \), \( v_j = g^{m_j} \), \( Z = e(g, g)^n \), \( PK = \{T, \{u_i\}_{i=1}^d, \{v_i\}_{i=1}^d, Z\} \). Finally, the system publishes these public parameters \( pp = \{q, g, g_1, \{u_i\}_{i=1}^d, \{v_i\}_{i=1}^d, G_1, G_2, H, T, Z\} \).

**KeyGen.** TA generates the private key and part of public key for user based on the user identity information ID, \( pk = H(ID) \), \( sk = H(ID)^n \), user randomly chooses \( \beta_i \in Z_q^* \), generates the other part public key \( h = H(ID)^h \).
SignGen. Given \( sk_i \) and a block data \( m_j \), signer randomly chooses \( \beta_i \in Z_q^* \) and \( u_i \), computes signature value: 
\[
\sigma_{i,j} = g_i(H(ID_i)^{\beta_i} u_i)^{m_j} = g_i H(ID_i)^{\omega_{i,j} m_j} \] 
and signature on multi-blocks is
\[
\sigma = \prod_{j=1}^n \sigma_{i,j} = \prod_{j=1}^n g_i H(ID_i)^{\omega_{i,j} \beta_i u_i} \] 
m. Later, signer sends signature \( \sigma \) to CSP.

SignVerify. CSP accepted signature \( \sigma \) after verification. If the equation holds, the signature is valid. Verifying the signature value of multiple blocks.
\[
\frac{e(\sigma, T)}{e(g, T)^{\frac{1}{n}} e(u, T)^{\frac{1}{\frac{1}{n} \sum m_j}}} = e(g_i, T) \]  

Challenge. After receiving the auditing query, TPA generates challenge message, randomly chooses a \( c \) element subset \( c \) from set \([1,n]\) and generates an element \( \rho \in Z_q^* \), the TPA sends \( \text{chal} = \{\rho, c\} \) to the CSP.

ProofGen. On receiving the challenging message \( \text{chal} = \{\rho, c\} \), the CSP generates the corresponding proof. Choose a random element \( j \in c \), calculates \( y_j = \rho^j \mod q \), \( m_j \cdot y_j = E \), \( y_j \sum_{j=1}^n m_j = Y \), \( \sigma = \prod_{j=1}^n \sigma_{i,j} \). The CSP returns an auditing \( \text{proof} = \{E, Y, \sigma_{i,j}, \sigma\} \) to the TPA.

ProofVerify. Give an auditing \( \text{proof} = \{E, Y, \sigma_{i,j}, \sigma\} \) to TPA. The TPA checks the correctness of this proof by checking the following equations:
\[
\frac{e(\sigma_{i,j}^j, T)}{e(g, T)^j e(h, T)^j} = e(u, T)^j \]  

Update. Suppose a group user modifies a data block \( m_j \) to \( m'_j \), however, the number of shared data block total is \( n \). He/she computes the signature for \( m'_j \):
\[
\sigma = \prod_{j=1}^n \sigma_{i,j}' = \prod_{j=1}^n g_i H(ID_i)^{\omega_{i,j} m'_j} \] 

Of course, user uploads \( m_j ' \) and \( \sigma_{i,j}', \sigma' \) to CSP and calculates the proof:
\[
m_j ' y_j = E', \quad \sum_{j=1}^n m_j ' y_j = Y', \quad \sigma' = \prod_{j=1}^n \sigma_{i,j}' \] 

Signature validation equation (6) is still valid after updating data.
User Revoke. If a user has malicious access, the data owner can delete the register and send the deletion information to the CSP, and then delete the signature stored in the cloud.

5. Correctness and Security Analysis
In this section, we analyze the proposed scheme in detail according to the security requirements mentioned in Design objectives. According to the properties of bilinear mapping, the correctness of the scheme can be verified through a straightforward calculation. The following theorems support the security analysis of the proposed scheme.

5.1. Correctness Analysis
The proposed scheme satisfies the correctness. Given share data blocks and the corresponding signature, the TPA can check the integrity of data during ProofVerify procedure.

Proof: we analyze the correctness of our verification based on Equation (7)-Equation (9) as following:

\[
\frac{e(\sigma, T)}{e(h, v) e(u, T)^{\sum_{j=1}^{n_y}}} = e\left(g, T \right)
\]  \( (6) \)

\[
\frac{e\left(g, T \right)}{e(g, T)^{\gamma_i} e(h, v)^{\gamma_i}} = \frac{e\left(g, T \right)}{e\left(g, T \right)^{\gamma_i} e\left(h, v \right)^{\gamma_i}} = e\left(g, T \right)
\]  \( (7) \)

\[
\frac{e\left(g, T \right)^{\gamma_i} e\left(h, v \right)^{\gamma_i}}{e\left(g, T \right)^{\gamma_i} e\left(h, v \right)^{\gamma_i}} = \frac{e\left(g, T \right)^{\gamma_i} e\left(h, v \right)^{\gamma_i}}{e\left(g, T \right)^{\gamma_i} e\left(h, v \right)^{\gamma_i}} = e\left(u, T \right)^{\gamma_i}
\]  \( (8) \)
### 5.2. Security Analysis [2]

Theorem 1 For any adversary $A$, as long as the CDH assumption on $G_1$ holds, it is computationally infeasible to forge a signature of our scheme.

Proof: In order to prove the non-forgery signature, we first suppose that adversary $A$ can adaptively choose data blocks, and can generate a forged signature with the advantage of $\varepsilon$ in polynomial time $t$. Assume in the coin toss game, an adversary $A$ is granted public key $(pk_1, \ldots, pk_d)$ of all the users, and can access the hash oracle and the signing oracle. The goal of adversary $A$ is to output a valid signature on a pair of block/identity $(m, ID_j)$, $(m, ID_j)$ has never been presented to the signing oracle. If adversary $A$ reaches this goal, then it wins the game, we can find an algorithm $B$ to solve the CDH problem on $G_1$ with a non-negligible advantage, which contradicts to the CDH assumption we introduced in Section 4. So, we first assume that adversary $A$ can generate a forgery the security game simulated by the following algorithm $B$.

Firstly, given $g^a \in G_1, g^{ab} \in G_1$, algorithm $B$ randomly chooses $\varepsilon_1, \ldots, \varepsilon_n \in Z_p$ and sets $\varepsilon_1 = 1$. Then, it sets $pk_i = H(ID_i)$. The adversary $A$ is given the public keys $(pk_1, \ldots, pk_d) = (H(ID_1), \ldots, H(ID_d))$. We assume $A$ can submit distinct queries and $A$ has previously issued a hash query on block $m$ and its signature.

**Hash Query:** On a hash query issued by $A$, $B$ flips a coin with probability $p$, of otherwise 1. Then $B$ randomly choose $r \in Z_p$, if the coin shows 0, $B$ return $H(ID_j)^{br}$ to $A$, otherwise it returns $H(ID_j)^{br}$. Since $r$ is randomly selected from $Z_p$, $(H(ID_j)^{br}$ and $(H(ID_j)^{br}$ are both elements of cyclic group $G_1$, therefore, the distribution of $H(ID_j)^{br}$ is identical to the distribution of $H(ID_j)^{br}$, which means that the result of flipped coin and the result of the hash query $A$ cannot be distinguished.

**Signing Query:** Assume $A$ that a signature query is issued for a block $m$ and its identifier $ID_j$. Based on the assumption of the game, a hash query has been issued by $B$ on this pair of block/identifier $(m, ID_j)$. If the coin $B$ flipped of this hash query appears to be 0, it fails and exits. Otherwise the signature is returned.

Clearly, given $g^a \in G_1, g^{ab} \in G_1$, algorithm $B$ can output $g^b \in G_1$ via the above security game with an advantage of $\varepsilon = (1 - p)^t$, if $A$ is able to generation a forgery with an advantage of $\varepsilon$. This advantage of algorithm $B$ is maximized as $\varepsilon = \frac{\varepsilon}{qH4_4}$. Therefore, if the advantage of adversary $A$ of generating a forgery is non-negligible, then the advantage of solving the CDH problem by running two examples of algorithm $B$ can’t be ignored.
which contradicts the assumption of CDH on $G_1$. Therefore, it is computationally infeasible for adversary $A$ to generate a forgery.

Theorem 2 For an untrusted cloud, as long as the DL assumption holds, it is computationally infeasible to generate a forgery of an auditing proof in our scheme.

Proof: As proved by Theorem 1, if the CDH problem on $G_1$ for an untrusted cloud is hard, it is infeasible to generate a forgery of a signature in computation. In our scheme, in addition to trying to calculate the forgery of an auditing certificate, if the untrusted cloud could win the following security game, denoted as Game 1, it can generate a forgery of an auditing certificate for corrupted shared data. Security game as follow:

Game 1. A public verifier sends an auditing challenge $chal = \{\rho, c\}$ to the cloud server, the auditing proof generated from the correct shared data $M$ should be $proof = \{E, Y, \sigma_{i,j}, \sigma\}$, which can pass the verification with Equation (2) and Equation (3). The untrusted cloud generates a forgery of the $proof = \{E, Y, \sigma_{i,j}, \sigma\}$ based on the corrupted shared data $M'$, which is nonzero since $M' \neq M$. If this invalid proof based on the corrupted shared data $M'$ is successfully validated, then the untrusted cloud wins. Otherwise, it fails.

Now, we prove that, if the untrusted cloud can win Game 1, we can find a solution to solve the DL problem in $G_1$, which contracts to the DL assumption that the DL problem in $G_1$ is computationally infeasible. We first assume that the untrusted cloud can win Game 1. Then, based on Equation (10) and Equation (11), we have

$$\frac{e(\sigma_{i,j}, T)}{e(g_i, T)^y e(h, v)^y} = e(u, T)^x$$

(10)

$$\frac{e(\sigma, T)}{e(g_i, T)^y e(h, v)^y} = e(u, T)^y$$

(11)

Because $\{E, Y, \sigma_{i,j}, \sigma\}$ is a correct auditing proof.

Theorem 3 In our public auditing, the probability that the public verifier distinguishes the identifier of all the signers on the $c$ selected blocks in shared data is at most $1/d_c$.

Proof: For any algorithm $A$, the probability of displaying the signer on a block is $1/d_c$. Because the $c$ selected blocks are signed independently signed, where $c \in [1, n]$, the total probability that the public verifier can distinguish all the signers’ identities on the $c$ selected blocks in shared data is at most $1/d_c$.

Through our proposed mechanism, the public verifier knows every block signed by Alice or Bob, because it requires the public key of two users to verify the correctness of the entire shared data. However, it cannot distinguish the signer on each particular block. Therefore, the public verifier cannot have additional advantages in exposing private information, such as who always signs the maximum number of blocks in shared data or which specific blocks is often modified by different group members.

Theorem 4 Given an auditing $\{E, Y, \sigma_{i,j}, \sigma\}$, it is computationally infeasible for a public verifier to reveal any private information on shared data as long as the DL assumption holds.

Proof: If $\sigma_{i,j}, \sigma, m_j, y_j = E, y_j \sum_{j=1}^c m_j = Y$, linear combination on block $m_j$, is directly sent to a public verifier, a public verifier could learn the content of data by solving linear equation. To
preserve private data, the combined element is computed with random value. In order to solve linear equations, the verifier should get $\sigma_i, \sigma, Y, E$. However, given signer’s private key $sk_i = H(ID)^{\varepsilon_i}$, computing $\varepsilon_i$ is as hard as solving the DL problem in $G_1$, which is computationally infeasible. Therefore, the TPA cannot reveal any private information on shared data.

### 6. Performance

In this section, we first analyze the computation cost and communication costs of our scheme, and give an experiment evaluate. For the sake of simplicity, some notations of described in the Table 1.

### Table 1. The performance notations

| Nations  | Description                                      | Nations  | Description          |
|----------|--------------------------------------------------|----------|----------------------|
| $T_{exp}$ | one exponentiation operation time                | $c$      | the number of challenge data blocks |
| $T_{mul}$ | one multiplication operation time                | $t$      | the number of attributes |
| $T_{pair}$ | one bilinear pairing operation time              | $n$      | the number of data blocks |
| $T_{hash}$ | one hash-to-point operation time                 | $s$      | the number of sub-block |
| $d$      | the number of signing users                      |          |                      |

#### 6.1. Computational Overhead

In order to highlight the security benefits of our proposed scheme, we compare it with a few recently proposed schemes, such as Yuan etc.’s scheme [4] and Zha etc.’s scheme [15]. For sake of simplicity, the specific comparison is shown in the Table 2.

### Table 2. Comparison of the computational overhead of the scheme

| Scheme | Scheme [4] | Scheme [15] | Ours          |
|--------|------------|-------------|---------------|
| Setup  | $(s + 2)nT_{exp} + nsT_{exp}$ | $T_{pair} + (t + 1)T_{exp}$ | $(d + 1)T_{exp}$ |
| KeyGen | $d(s + 4)T_{exp}$ | $d(T_{hash} + T_{mul} + 2T_{exp})$ | $d(2T_{hash} + T_{exp})$ |
| SignGen| $d((s + 1)T_{mul} + 2(s + 1)T_{exp})$ | $d(3T_{hash} + 6T_{mul} + 8T_{exp})$ | $d(2T_{hash} + 2T_{mul} + 2T_{exp})$ |
| Verify | $3T_{pair} + 3T_{mul} + 6T_{exp}$ | $3T_{pair} + T_{mul} + 2T_{exp}$ | $3T_{pair} + 3T_{exp} + T_{mul}$ |
| Update | $2(s + 1)T_{exp} + dT_{pair} + (2s + d + 1)T_{mul}$ | $3T_{pair} + 3T_{hash} + 6T_{mul} + 5T_{exp}$ | $6T_{pair} + 4T_{hash} + 5T_{mul} + 6T_{exp}$ |
| Revocation | $d(T_{pair} + T_{exp})$ | $3T_{hash} + T_{mul} + 2T_{exp}$ | $2T_{hash} + 2T_{mul} + 4T_{exp}$ |

In general, computational overhead in scheme [4] increases with number of data blocks, computational overhead linearly increase with the number of revocation users in scheme [15], so the overhead is much greater than that in ours. The computational overhead in scheme [15] is equivalent to that in our scheme in verification phase, but our scheme supports signature verification and batch auditing of multi-data blocks and the computational overhead is same as that of single data block.

#### 6.2. Communication Overhead

About communication overhead, we mainly analyze the auditing phase. Among, $|\Gamma|$ is the size of random element, $|G_1|$ is the size of an element in group of $G_1$, $|I|$ is the size of a block index. As shown in the Table 3 of the particular comparison.
Table 3. Comparison of communication overhead during audit phase

| Scheme         | Communication Overhead |
|----------------|------------------------|
| Scheme[4]      | \( c|l|+(d+3)|G_1|+2|\nu| \) |
| Scheme[15]     | \( 6|G_1|+c|\nu| \) |
| Ours           | \( 5|G_1|+c|\nu| \) |

In our scheme, the communication cost mainly comes from the challenging message and the proof information. Our batch verification design allows users to aggregate multiple tasks into one proof, and which achieves the same communication cost as single file scenario. The communication in scheme [4] cost \( c|l|+(d+3)|G_1|+2|\nu| \), the communication cost in scheme [15] \( 6|G_1|+c|\nu| \). Therefore, the communication cost of our scheme is less than that in scheme [4] and scheme [15].

6.3. Functional Comparison

The table 4 shows a comparison on main functions between our scheme with scheme [4] and scheme. Our scheme is more functional.

Table 4. Function comparison

| Scheme          | Scheme[4] | Scheme[15] | Ours |
|-----------------|-----------|------------|------|
| Batch Auditing  | No        | No         | Yes  |
| Signature Verify| No        | No         | Yes  |
| Challenge/Proof | No        | Yes        | Yes  |
| Update          | No        | Yes        | Yes  |

6.4. Experimental Results

To evaluate the performance of our proposed scheme, we fully implemented our proposed scheme using C++ with C Pairing-Based Cryptography Library (C-PBC) [16]. We deploy nodes running Linux with 2.5GHz Intel i5-7200u CPU and 8GB memory, windows 10x64 Operating system. The specific experimental results are shown in Figure 2-5.

For simplicity and fair comparison, we still set the number of sector as \( s = 1 \). To demonstrate the efficiency of setup, key generation, signature generation and signature verification, we respectively evaluate the execution time of the Setup, KeyGen, SignGen and SignVerify phases in scheme [4,15] and our scheme. In this experiment, we adopt the elliptic curve with a based field size of 512 bits in our experiments. We choose \(|q|=160\) bits and \(|q|=80\) bits, and set the number of selected blocks as \( c=300 \) in the process of auditing. Compared to the size of entire shared data, the running time of each phase in schemes [4,15] and ours linearly increases with the number of users, but compared with the scheme [4,15], the running time of our scheme is less than the scheme [4,15], so our scheme is more efficient in terms of auditing process.

Through the above comparative analysis, we found that the communication overhead of our scheme in both the computational and auditing phase is more efficient, and the time of our scheme required in the specific operation phases, namely, Setup, KeyGen, SignGen, SignVerify, is less than the scheme [4, 15], our scheme has higher communication efficiency.
Figure 2. Setup time

Figure 3. Key generation time

Figure 4. Signature generate time
7. Conclusion
In this paper, we propose a dynamic group privacy protection public batch auditing scheme. We utilize identity-based signature to generate the tag of each data block, so that the TPA can efficiently check the integrity of shared data, but cannot obtain the identity information of data owner on the sample blocks to improve the audit efficiency. Moreover, the scheme can prevent attacker from forging signatures and it supports batch auditing. In order to protect privacy and resist the attack of the TPA, a masking technique with the random value has been utilized. The security analysis shows that our mechanism meets all the security requirements of public auditing, and the performance evaluation indicates that our scheme is efficient. Thus, we will further discuss dynamic group privacy protection public batch auditing scheme based on block chain in our future research.

8. Acknowledgment
Thank you very much for your valuable advice from anonymous reviewers and editors. The work was supported in part by the National Natural Science Foundation of China under (61662071, 61562077).

9. References
[1] Ateniese G, Burns R and Curtmola R 2007 Provable data possession at untrusted stores[C]// ACM Conf. on Computer and Communications Security. ACM pp 598-609
[2] Wang B, Li B and Li H 2012 Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud[C]// IEEE Fifth Int. Conf. on Cloud Computing. IEEE Computer Society pp 295-302
[3] Wang B, Li B and Li H 2012 Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud[M]// Applied Cryptography and Network Security. Springer Berlin Heidelberg pp 507-525
[4] Yuan J and Yu S 2014 Efficient public integrity checking for cloud data sharing with multi-user modification[C]// INFOCOM Proc. IEEE pp 2121-2129
[5] Shacham H and Waters B 2008 Compact Proofs of Retrievability. Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT ’08) pp 90-107
[6] Erway C C, Papamanthou C and Tamassia R 2015 Dynamic Provable Data Possession[J]. Acm Transactions on Information & System Security 17(4) pp 1-29
[7] Wang Q, Wang C and Li J 2009 Enabling public verifiability and data dynamics for storage security in cloud computing[C]//European Conference on Research in Computer Security. Springer-Verlag pp 355-370
[8] Wang C, Wang Q and Ren K 2013 Ensuring data storage security in Cloud Computing[J] 2009(12)
[9] Wu L, Wang J, Sherali Z and He D 2018 Privacy-preserving auditing scheme for shared data in public clouds[J] The Journal of Supercomputing
[10] Zhu Y, Wang H and Hu Z 2011 Dynamic audit services for integrity verification of outsourced storages in clouds[C]// ACM Symposium on Applied Computing, DBLP pp 1550-57
[11] Cao N, Yu S and Yang Z 2012 LT codes-based secure and reliable cloud storage service[J]. Proc. - IEEE INFOCOM 131(5) pp 693-701
[12] Wang B, Li B and Li H 2013 Certificateless public auditing for data integrity in the cloud[C]// Communications and Network Security. IEEE pp 136-144
[13] Yuan J and Yu S 2015 Public Integrity Auditing for Dynamic Data Sharing With Multiuser Modification[J]. IEEE Transactions on Information Forensics & Security 10(8) pp 1717-26
[14] Wang B, Li B and Li H 2015 Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud[J]. IEEE Transactions on Services Computing 8(1) pp 92-106
[15] Zha Y, Luo S, Li W and Bian J 2017 Dynamic group public auditing scheme for shared data on attribute-based threshold signature[J] Journal of Beijing University of Posts and Telecommunications 40(5) pp 43-49
[16] http://crypto.stanford.edu/pc/