


Privacy-Preserving Public Auditing for Non-manager Group Shared Data

Longxia Huang¹ · Gongxuan Zhang¹ · Anmin Fu¹ 

Published online: 28 March 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract By the widespread use of cloud storage service, users get a lot of conveniences such as low-price file remote storage and flexible file sharing. The research points in cloud computing include the verification of data integrity, the protection of data privacy and flexible data access. The integrity of data is ensured by a challenge-and-response protocol based on the signatures generated by group users. Many existing schemes use group signatures to make sure that the data stored in cloud is intact for the purpose of privacy and anonymity. However, group signatures do not consider user equality and the problem of frameability caused by group managers. Therefore, we propose a data sharing scheme PSFS to support user equality and traceability meanwhile based on our previous work HA-DGSP. PSFS has some secure properties such as correctness, traceability, homomorphic authentication and practical data sharing. The practical data sharing ensures that the data owner won't loss the control of the file data during the sharing and the data owner will get effective incentive of data sharing. The effective incentive is realized by the technology of blockchain. The experimental results show that the communication overhead and computational overhead of PSFS is acceptable.

Keywords File sharing · Non-manager group · Privacy protection · Homomorphic authentication · Blockchain

1 Introduction

By the widespread use of cloud computing [1], it gives users various capabilities such as storing and accessing their data in the centres, which greatly reduce users' pressure on storage overhead and calculation. What's more, this technology also makes it easier for

✉ Anmin Fu
fuam@njust.edu.cn

¹ School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

data owners to share data with group members in the cloud than before [2]. The major popular research points in this area include data integrity [3], data privacy [4] and data access [5]. The integrity of files stored in the cloud must be guaranteed [6], as data owners may delete the files they stored physically after uploading the files to the cloud to save storage space. However, the data stored in the cloud may be invalid because of hardware failures or error behaviours of users. What's worse, to keep reputation or avoid losing its profits, the cloud may even conceal accidents about data errors [7]. Besides security threats mentioned forward, attacks outside may also put data at risk [8]. Therefore, it is an urgent security demand to ensure the integrity of shared data stored in the cloud.

So far, various integrity checking schemes [9–14] have been proposed based on different signatures. Most of the existing public auditing schemes are proposed for groups with managers and only a few schemes [11–13] support data sharing for non-manager groups. In a group with manager, the manager has the ability to trace user identity and manager data, which may causes the problem of frameability [14]. Once a group manager discloses information, serious threats to data integrity and identity privacy will appear. On the other hand, such centralized control is undesirable in many applications. For example, if users want to be in a group where the group membership is managed jointly by all group members, none of these schemes is suitable. Non-manager groups provide users an equal environment for data sharing, which is in line with the needs of modern people. To avoid the participation of unqualified data owners, it's essential to trace the identity of data signer in non-manager groups [13]. It is necessary for all members to share the responsibility of data management and identity tracing [15]. Therefore, a public auditing scheme which supports non-manager groups is practical in a practical cloud data sharing.

In a secure public auditing scheme, the problem of privacy disclosure during the integrity verification should be avoided. To support privacy protection, Wang et al. [6] used the technology of random masking and public key-based homomorphic linear authenticator to prevent TPA from learning any message of the data stored in the cloud server during the process of auditing. Yu et al. [16] also applied “zero-knowledge privacy” to ensure that verifier cannot obtain any information from publicly available data.

The practicality and security during the data sharing among group members should be taken into consideration. The existing public auditing schemes which support data sharing mainly focus on the updating of the outsourced data [17] and the authentication of users [18]. However, none scheme considers the fact that the data owner will lose the control of the data once the data is copied. What's more, the data owners of the shared data should get reward according to the access of the data. In this paper, we construct a practical and secure file sharing scheme for non-manager groups based on public auditing and blockchain [19, 20].

To address these above problems, our work distributes trapdoor of the signer identity in non-manager groups by utilizing publicly verifiable secret sharing (PVSS) [21], which enables equal group members work together to trace the malicious signers. To handle the problem in HA-DGSP [13], we propose a practical data sharing scheme for non-manager groups. By using random masking, our protocol protects the data content stored in the cloud server against the verifier during the process of integrity verification. In addition, we introduce a novel technology blockchain in this paper to realize effective incentive for data sharing.

1.1 Related Work

Ateniese et al. [22] are the first researchers who proposed the provable data possession (PDP) model, which is the base of integrity checking schemes which allow a user who stores data in an untrusted server to check that whether the server indeed stores the original data without retrieving the whole data. However, it cannot guarantee that the data can be retrieved. Then, Juels and Kaliski [23] proposed proofs of retrievability (POR) scheme which enables a back-up or archive service to produce a proof that the data can be retrieved by the verifier. Based on PDP and PoR models, many extended schemes [6–18, 24] have been proposed to solve different problems. To release users' burden on calculation, public auditing [6–8, 10–18] was proposed to make a third party auditor (TPA) verify the integrity of data instead of users. In this process, the TPA may get some important message and carry on statistical analysis to get privacy information. Therefore, the property of privacy-preserving [6] should be considered seriously.

The aforementioned researches [6–18] studied data integrity in different groups, but none of them considers the group without manager but with the ability of traceability. To realize integrity checking in data sharing groups, Wang et al. [9] proposed Knox to construct homomorphic authenticators based on group signatures. Thus, the TPA verifies the integrity of the shared data without retrieving the whole data. In Knox, group managers have the ability of tracing the identity of signer. But Knox cannot support public auditing, and it can't be used in a group with equal members because of high authority of group manager. Oruta [11] was the first privacy protection mechanism which supports public auditing for the shared data stored in the cloud. Oruta exploited ring signatures to generate the information for the integrity verification of shared data. However, Oruta cannot against the attack of group member changing. Then, Wang et al. [12] proposed IAID to handle the attack and introduced the concept of incentive in public auditing.

Oruta and IAID both use ring signature to construct the public auditing scheme. This kind of public auditing provides users with an equal environment, but the identity of the real signer cannot be traced by other users, which may cause malicious data uploading. HA-DGAP claimed that it realizes traceability and equal powder. But it used the public key of signers to verify the correctness, which runs counter to the aims of identity privacy. In this paper, we construct a secure and practical file sharing scheme with the methods in IAID and HA-DGSP. The incentive in IAID is proposed for data signers, not for data sharing groups. To construct an efficient incentive in data sharing group, we introduce blockchain in this paper.

1.2 Our Contributions

The main contributions of this paper can be concluded as follows.

1. We present a practical and secure file sharing scheme PSFS in non-manager group with the properties of traceability and privacy-preserving. In PSFS, each member is given equal power and the identity of the malicious user can be traced.
2. PSFS makes the data file controllable during the sharing by providing visitors with access interface instead of sharing the data file.
3. Blockchain technology is used in this paper to ensure effective incentive for data owners, which has never been proposed before in data sharing schemes.
4. We prove the security of PSFS and evaluate the performance to show that the overhead is acceptable.

Table 1 Function comparison

Function	Oruta [11]	HA-DGSP [12]	IAID [13]	PSFS
Identity privacy	Yes	No	Yes	Yes
Public auditing	Yes	Yes	Yes	Yes
Traceability	No	Yes	No	Yes
Non-manager	Yes	Yes	Yes	Yes
Secure file sharing	No	No	No	Yes
Incentive	No	No	Yes	Yes

We describe a high-level comparison between PSFS and the existing work [11–13] is shown in Table 1. We can find that PSFS supports identity privacy, public auditing, traceability, non-manager, secure file sharing, incentive and it exists no designated authority. The identity of signer can be traced only by legal members, and the signer's anonymity is against all non-members.

1.3 Organization

Section 2 introduces some preliminaries used in this paper. Then, the security problems, the design objectives and the system models are described in Sect. 3. In Sect. 4, we show the details of the proposed scheme and present the security analysis, followed by Sect. 5 that the performance of PSFS is evaluated. Finally, Sect. 6 gives the concluding remark of this paper.

2 Preliminaries

In this section, we introduce some preliminaries, including bilinear maps, homomorphic verifiable tags and blockchain technology.

2.1 Bilinear Map

Let G_1, G_2 be two multiplicative cyclic groups of prime order q , g be a generator of group G_1 . A bilinear map $e : G_1 \times G_1 \rightarrow G_2$ has the following properties.

1. For all $u, v \in G_1$ and all $a, b \in \mathbb{Z}_q$, $e(u^a, v^b) = e(u, v)^{ab} = e(u^b, v^a)$.
2. For any $u_1, u_2, v \in G_1$, $e(u_1 \cdot u_2, v) = e(u_1, v) \cdot e(u_2, v)$.
3. There exists an efficiently computable algorithm for computing e and the map should be nontrivial, i.e., e is nondegenerate: $e(g, g) \neq 1$.

In this paper, we utilize two problems in bilinear map which make the map secure against attackers.

- *Discrete Logarithm (DL) Problem* Given g and $Y = g^x \in G_1$, it is computational infeasible to compute $x \in \mathbb{Z}_q$.
- *Computational Diffie-Hellman (CDH) Problem* Given $(g, g^\alpha, g^\beta) \in G_1$, it is computational infeasible to compute $g^{\alpha\beta} \in G_1$ without knowing $\alpha, \beta \in \mathbb{Z}_q$.

2.2 Homomorphic Verifiable Tags

Homomorphic Verifiable Tags [11] enable a public verifier to audit the integrity of remote data without downloading the whole data. Besides unforgeability, a homomorphic verifiable signature scheme should also satisfies blockless verification and non-malleability:

Assume that (sk, pk) denotes the private/public key pair of a signer, σ_1 and σ_2 denote the signatures on block $m_1, m_2 \in Z_q$ respectively.

Unforgeability Only a party with the right private key can generate a valid signature, this property relies on the hardness of CDH.

Blockless verification A verifier only needs to verify the correctness of the data stored in the cloud with a linear combination of all blocks via a challenge-and-response protocol without knowing the content of blocks. Specifically, given two signatures σ_1 and σ_2 on blocks m_1, m_2 respectively, random values $y_1, y_2 \in Z_q$ and a linear combined block $m' = y_1 m_1 + m_2$, a verifier can check the correctness of the combined block m' without knowing blocks m_1, m_2 .

Non-malleability The one who does not have the valid private keys is unable to generate valid signatures on the combined blocks only by combining the existing signatures. Specifically, with two signatures σ_1 and σ_2 on blocks m_1, m_2 respectively, random values $y_1, y_2 \in Z_q$ and a linear combined block $m' = y_1 m_1 + m_2$, a user who does not obtain the secret key sk cannot generate a valid signature σ' on block m' by combining σ_1 and σ_2 .

2.3 Blockchain Technology

Blockchain, a well-known technology nowadays comes out with the development of Bitcoin, provides users with a distributed peer-to-peer network where untrusted members are able to interact with each other and no trusted intermediary is needed, in a verifiable manner [19]. Zyskind et al. [20] ensured users to own and control their data by using blockchain technology in a decentralized personal data management system. In the protocol, they combine blockchain and off-blockchain storage to realize a data management platform, which can be further used in cloud storage. In this paper, the storage and access mechanisms are based on blockchain technology, which improves the security level of the proposed scheme.

3 Problem Statements

In this section, we illustrate the security problems in data sharing schemes, the design objectives of our proposed mechanism and the system models of the proposed scheme including data sharing model, public auditing model and file management model.

3.1 Security Problems

In public auditing schemes for data sharing groups, the integrity of files stored in the cloud is the most important issue as the file owner may no longer store the files after uploading it to the cloud. However, the cloud may conceal accidents about data errors. Thus, it's necessary to ensure the validity of shared files. What's more, the file owner loses the control of the file once it has been copied, which leads to the impairment of benefit.

Therefore, the uncontrollable of shared files should be taken into consideration, which is the first time proposed in public auditing schemes for data sharing.

Integrity Threat If users do not store their data after outsourcing it to the cloud, they will no longer possess the outsourced data physically. When cloud server inadvertently modifies or even removes data due to hardware or software errors or attackers from outside, the integrity of data will be threaten. What's worse, the cloud service will conceal the fact that the data has been damaged in order to maintain their interests and reputation. Then users will no longer get the right data.

Identity and Data Privacy Threat During the auditing, a semi-trusted verifier may try to get identity information or data information from verification information. For example, the block which has been modified frequently may be more important, and after several auditing performances, some private and sensitive information may reveal to the verifier.

Uncontrollable file sharing Due to the property of digital files, the file owner cannot control the ownership of the file once it has been copied, which seriously damages the interests of the file owner. What's worse, this problem has never been considered in public auditing schemes for data sharing.

3.2 Design Goals

To enable a privacy-preserving public integrity verification for non-manager groups, our protocol should achieve the following security and performance guarantees.

1. Homomorphic verifiable: A public verifier is allowed to check the integrity of data stored in the cloud without knowing the whole data.
2. Public auditing: A verifier has the ability to audit the integrity of the data stored in the cloud without knowing the secret keys of data owners.
3. Identity and data privacy: The public verifier couldn't retrieve any identity information or data information from the message he get during the auditing.
4. Traceability: The identity of signers can be figured out when some group members work together.
5. Unforgeability: Only the data owner has the ability to sign for the file data.
6. Practical file access: When a user applies file access, the system gives the visitor the access interface instead of sending the file data. In addition, the data owner will get rewards according to the unchangeable access history of the files he shared in the group.

3.3 The System Model

The system consists of three models such as data sharing model, public auditing model and file management model. First, we describe the new data sharing model for group users. As shown in Fig. 1, the data sharing model is consisted of two entities: a user group and the cloud server, where the user group includes file uploader and other normal users who may want to access the shared file. The file uploaders sign the files with their secret keys and the public message of other members to protect their identity and the files. Then, they upload the files and signature to the cloud for data sharing. When other group members ask for file access, the cloud sends the file access interface to the group members, which ensures the benefits of the file owners.

The public auditing model has three kinds of parties: a user group, a verifier and the cloud server. Besides checking the integrity of files stored in the cloud, a group member

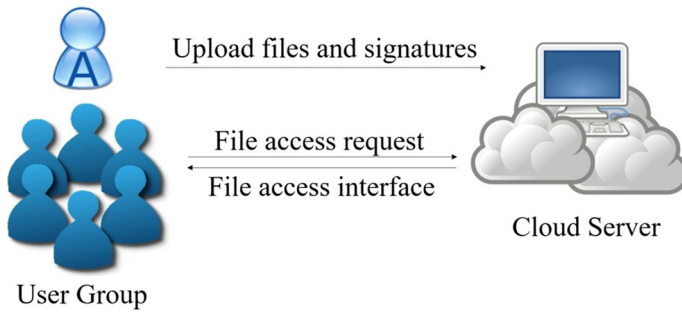


Fig. 1 Data sharing model

may send a verification request when he wants to ensure the validity of files stored in the cloud as each group user has validity. The files and signatures are useless when the validity of a file uploader is no longer in force.

As shown in Fig. 2, a group member sends a verification request to the verifier, and then the verifier asks the cloud server to generate the proof message by sending a challenge message. At last, the verifier returns the result of verification according to the challenge message and the proof message.

The storage based on blockchain technology is also a new point in this paper. Blockchain has the advantage of tamper resistance which provides users with correct and trusted recording. In Fig. 3, the blockchain of file uploading lists the file identity of the file, the identity of the uploader and the validity of the uploader. When a user wants to access some file block, the user needs to send access time and his identity to the trusted service provider, and these information will be recorded in the access blockchain. The file uploader gets rewards according to these blockchains. As blockchain records in chronological order, it gives a suitable solution for awarding of duplicated files. For example, if two different users upload the same file or a user upload the same file twice, we only admin the one who upload earlier.

4 Practical and Secure File Sharing Scheme

In this section, we design a practical and secure file sharing scheme PSFS for non-manager groups in the cloud based on secret share technology [21] and Wang et al.'s public auditing scheme IAID-PDP [12]. In PSFS, each member in the group is able to upload file to get reward if other group members access the upload file. Other group members need to pay for his access. To ensure the integrity of the accessed files, each group member has the ability to initiate integrity verification. If the file is not valid, in other words, the file does not pass the verification, t group members can work together to trace the file uploader. Noting that the initiator of integrity checking can delegate a public auditor to verify and he also can check data integrity himself.

4.1 Construction of Scheme

PSFS includes nine algorithms: **Setup**, **KeyGen**, **ShareGen**, **SignGen**, **ChallGen**, **ProofGen**, **ProofVerify**, **Access** and **Trace**, where algorithms **Setup**, **KeyGen** and

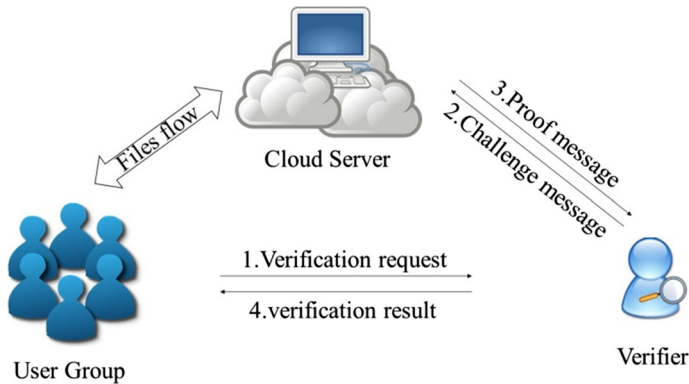
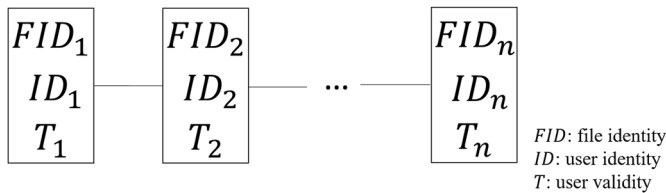


Fig. 2 Public auditing model

The blockchain of file uploading .



The access blockchain of file block m_s .



Fig. 3 File management model

SignGen provide file uploading and storing, algorithms **ChallGen**, **ProofGen** and **ProofVerify** realize public auditing, algorithms **ShareGen** and **Trace** make other group members work together to trace the file signer, and algorithm **Access** answers file access requests from group members. The details of these algorithms are shown as follows.

Setup This algorithm runs by a trusted centre. Input a security parameter λ , this algorithm establishes the public parameters $(G_1, G_2, q, e, g, h, H)$, where q is a large prime number, G_1 and G_2 are two cyclic multiplicative groups with order q which satisfies a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, g is a generator of group G_1 and h, H are two hash functions with $h : \{0, 1\}^* \rightarrow Z_q$ and $H : \{0, 1\}^* \rightarrow G_1$ respectively. This algorithm also outputs the secret key and the public key of PKG. PKG picks a random number $x \in Z_q$, where Z_q denotes a prime finite field, as the secret key and calculates $P = g^x \in G_1$ as the public key, where q is a large prime number, G_1 is a cyclic multiplicative group with order q . Noting that PKG keeps x secretly and publishes the public key g^x .

KeyGen This algorithm runs by the PKG. Input a user's identity ID_i , validity T_i the public parameters, the secret key and public key of PKG, it outputs the private key of the user with identity ID_i . In this step, the user first sends the identity ID_i to PKG. Then, PKG

randomly chooses a number $r_i \in Z_q$, and calculates $R_i = g^{r_i}$, $sk_i = r_i + xh(ID_i, T_i)$ and $pk_i = g^{sk_i}$. Finally, PKG returns (R_i, sk_i) to the user with identity ID_i . Upon received (R_i, sk_i) , the user verifies the correctness of it by checking $g^{sk_i} = R_i P^{h(ID_i, T_i)}$. If the equality holds, the user accepts the key pair and publishes R_i as user public key.

ShareGen This algorithm runs by a signer to distribute his public key to other members. Input the public key R_k of signer with identity ID_k where $1 \leq k \leq n$, it outputs the signer's public key sharing *share*. To realize the property of identity traceability, the signer needs to computer secret sharing *share* of his identity. Firstly, the signer with identity ID_k randomly chooses $\alpha_j \in Z_q, 0 \leq j \leq t-1$ to formulate a random polynomial $p(z) = \alpha_0 + \alpha_1 z + \dots + \alpha_{t-1} z^{t-1}$. The signer computes $\tau_j = g^{\alpha_j}, 0 \leq j \leq t-1$ and $\chi_i = \prod_{j=0}^{t-1} \tau_j^{ij}, i \in [1, n]$. The signer will keep the polynomial secretly and share $\tau_j = g^{\alpha_j}, 0 \leq j \leq t-1$ with other group members. Secondly, the signer calculates $\eta_i = pk_i^{p(i)}, 1 \leq i \leq n$ and publishes to all group users. Finally, the signer computes $s = H(\chi_1, \dots, \chi_n, \eta_1, \dots, \eta_n)$ and sets *share* = $(\tau_0, \dots, \tau_{t-1}, \eta_1, \dots, \eta_n, s)$.

SignGen This algorithm runs by a user who uploads files to the cloud for sharing. Input the set of all users' identities $ID = \{ID_1, ID_2, \dots, ID_n\}$, the key pair of signer u_k , and the file block m_y with name FID_y and index y , it outputs the signature of the file. For each block m_y , the signer first picks $a_{y,i} \in Z_q$, and calculates $\rho_{y,i} = g^{a_{y,i}} \in G_1$ for all $i = \{1, 2, \dots, k-1, k+1, \dots, n\}$. The signer also computes $\rho_{y,k} =$

$\left(\frac{H(y, FID_y) g^{m_y}}{\prod_{i \neq k} (R_i P^{h(ID_i, T_i)})^{a_{y,i}}} \right)^{1/sk_k} \in G_1$ and $c_y = g^{a_0} pk_k$. The signature of block m_y is $\rho_y = (\rho_{y,1}, \dots, \rho_{y,n}, c_y)$. At last, the signer uploads block-sign pairs (FID_y, m_y, ρ_y) to the cloud for file sharing and records (FID_y, ID_k, T_k) in the trusted service provider.

ChallGen This algorithm runs by a public verifier. Input a checking request of any user T_i , it outputs challenge message or reject. The public verifier first checks the validity T_i . If T_i is useless, the verifier ignores the request; otherwise the verifier randomly picks a c -element subset C of set $[1, d]$ to locate the c selected random blocks that will be checked in this auditing task. Then the verifier chooses c random values $v_j \rightarrow Z_q, j \in C$ and sends challenge message *chall* = $\{(j, v_j)\}_{j \in C}$ to the cloud.

ProofGen This algorithm runs by the cloud. Input the challenge message, it outputs the proof message. After receiving the challenge message *chall* = $\{(j, v_j)\}_{j \in C}$, the cloud calculates $\mu' = \sum_{j \in C} v_j m_j$ and aggregates all the blocks' tags $o_i = \prod_{j \in C} \rho_{j,u}^{v_j} \in G_1, 1 \leq i \leq n$. Then the cloud randomly picks $\kappa \rightarrow Z_q$, calculates $K = g^\kappa$, $\mu = \kappa + \mu'$, and sends audit proof $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{I_j\}_{j \in C}\}$ to the verifier.

ProofVerify This algorithm runs by a public verifier. Input the audit proof message, it outputs *success* or *fail*. Upon receiving audit proof $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{FID_j\}_{j \in C}\}$ from the cloud, the public verifier checks the correctness of the proof message by verifying the following equation.

$$e\left(g^\mu \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) = e(K, g) \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, o_i\right) \quad (1)$$

Access This algorithm runs by the trusted service. Input a user's identity ID_i and the file identity FID_β which the user wants to access, it outputs the access interface of file block

whose file identity is FID_β . The trusted service provider records IDA_e as the user's identity ID_i and the access time t_e in the access blockchain of the file block whose file identity is FID_β . The user who uploaded and signed the file block whose file identity is FID_β will get award according the valuation standard. For example, reward the file owner according to the times of access or total access time. If the data owner is rewarded according to the times of access, the trusted service provider outputs the length of the access blockchain of the data owner's file block. Otherwise, the trusted service provider computes the sum of the access time stored in the access blockchain of the data owner's file block and outputs the sum.

Trace This algorithm runs by t valid group users. Input the signer's public key sharing $share$ and a signature $\rho_y = \{\rho_{y,1}, \dots, \rho_{y,n}, c_y\}$, it outputs the public key of the signer. With public key sharing information $share = (\tau_0, \dots, \tau_{t-1}, \eta_1, \dots, \eta_n, s)$, any valid group user calculates $\chi_i = \prod_{j=0}^{t-1} \tau_j^{ij}, i \in [1, n]$, and checks the correctness of $share$ by verify the equality $s = H(\chi_1, \dots, \chi_n, \eta_1, \dots, \eta_n)$. Only when the above equality holds, t valid group members trace the identity of signer as follow. Each one of t valid group members uses his secret key sk_i calculates $\zeta_i = \eta_i^{sk_i^{-1}}, \lambda_i = \prod_{j=1, \dots, t, j \neq i} \frac{i}{j-i}$, and gets $\theta = \prod_{i=1}^t \zeta_i^{\lambda_i}$. Finally, the public key of the signer is calculated by $pk_k = c_y \theta^{-1}$.

4.2 Security Analysis of PSFS

In this section, we prove the security of the proposed scheme PSFS from the following five parts: homomorphic authentication, public auditing, identity privacy and data privacy, traceability and unforgeability.

4.2.1 Homomorphic Authentication

We need to prove that the signature algorithm in PSFS supports homomorphic authentication, the basic theory used in public auditing. As a homomorphic verifiable signature scheme should satisfy blockless verification and non-malleability, we need to prove that the signature algorithm satisfy these two properties.

Blockless Verifiability Given the public keys of all group users message $\{(ID_i, R_i, T_i)\}_{1 \leq i \leq n}$, signatures $sig_1 = (FID_1, \rho_1)$, $sig_2 = (FID_2, \rho_2)$ on m_1 and m_2 respectively, and $m' = b_1 m_1 + b_2 m_2 \in Z_q$, where $b_1, b_2 \in Z_q$, $\rho_1 = \{\rho_{1,1}, \dots, \rho_{1,n}, c_1\}$ and $\rho_2 = \{\rho_{2,1}, \dots, \rho_{2,n}, c_2\}$, a verifier is able to check the correctness of the block $m' = b_1 m_1 + b_2 m_2$ by verifying

$$e\left(g^{m'} \prod_{j=1}^2 H(j, FID_j)^{b_j}, g\right) = \prod_{i=1}^n e\left(R_i P^{h(ID_i, T_i)}, \rho_{1,i}^{b_1} \rho_{2,i}^{b_2}\right) \quad (2)$$

without knowing m_1, m_2 . Based on properties of bilinear maps, correctness of Eq. 2 can be proved as follow.

$$\begin{aligned}
 & e\left(g^{m'} \prod_{j=1}^2 H(j, FID_j)^{b_j}, g\right) \\
 &= e\left(g^{b_1 m_1 + b_2 m_2} \prod_{j=1}^2 H(j, FID_j)^{b_j}, g\right) \\
 &= e\left(g^{b_1 m_1} H(1, FID_1)^{b_1} \cdot g^{b_2 m_2} H(2, FID_2)^{b_2}, g\right) \\
 &= e\left((g^{m_1} H(1, FID_1))^{b_1} \cdot (g^{m_2} \cdot H(2, FID_2))^{b_2}, g\right) \\
 &= e\left((g^{m_1} H(1, FID_1))^{b_1}, g\right) \cdot e\left((g^{m_2} \cdot H(2, FID_2))^{b_2}, g\right) \\
 &= e(g^{m_1} H(1, FID_1), g)^{b_1} \cdot e(g^{m_2} \cdot H(2, FID_2), g)^{b_2} \\
 &= \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, \rho_{1,i}^{b_1}\right) \cdot \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, \rho_{2,i}^{b_2}\right) \\
 &= \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, \rho_{1,i}^{b_1} \cdot \rho_{2,i}^{b_2}\right)
 \end{aligned}$$

It is clear that the signature algorithm in PSFS supports blockless verifiability.

Non-malleability An attacker who does not own the valid private key, which is used to sign the message, cannot generate a valid signature sig' of a combined block $m' = b_1 m_1 + b_2 m_2$ only by combining sig_1 and sig_2 with b_1 and b_2 . The hardness of this issue depends on the reality that the hash function H we use in the signature algorithm is a one-way hash function. First assume that sig_1 and sig_2 are signed by the same user u_k and the combined signature of user u_k is

$$\begin{aligned}
 \rho'_k &= \rho_{1,k}^{b_1} \cdot \rho_{2,k}^{b_2} = \left(\frac{H(1, FID_1) g^{m_1}}{\prod_{i \neq k} (R_i P^{h(ID_i, T_i)})^{a_{1,i}}} \right)^{\frac{b_1}{sk_k}} \cdot \left(\frac{H(2, FID_2) g^{m_2}}{\prod_{i \neq k} (R_i P^{h(ID_i, T_i)})^{a_{2,i}}} \right)^{\frac{b_2}{sk_k}} \\
 &= \left(\left(\frac{H(1, FID_1) g^{m_1}}{\prod_{i \neq k} (R_i P^{h(ID_i, T_i)})^{a_{1,i}}} \right)^{b_1} \cdot \left(\frac{H(2, FID_2) g^{m_2}}{\prod_{i \neq k} (R_i P^{h(ID_i, T_i)})^{a_{2,i}}} \right)^{b_2} \right)^{\frac{1}{sk_k}}
 \end{aligned}$$

For other group members $u_i, i \in [1, n], i \neq k$, the signatures are $\rho'_i = \rho_{1,i}^{b_1} \cdot \rho_{2,i}^{b_2} = (g^{a_{1,i}})^{b_1} \cdot (g^{a_{2,i}})^{b_2} = g^{a_{1,i} b_1 + a_{2,i} b_2}$, where $a_{1,i}$ and $a_{2,i}$ are randomly selected. Thus the combined signature for block m' is $\rho' = (\rho'_1, \dots, \rho'_n)$. If the combined signature is correct, we have

$$\begin{aligned}
 \prod_{i=1}^n e\left(R_i P^{h(ID_i, T_i)}, \rho'_i\right) &= e\left((g^{m_1} H(1, FID_1))^{b_1} \cdot (g^{m_2} H(2, FID_2))^{b_2}, g\right) \\
 &= e\left(g^{b_1 m_1 + b_2 m_2} \prod_{j=1}^2 H(j, FID_j)^{b_j}, g\right) = e(g^{m'} H(X, FID_X), g).
 \end{aligned}$$

Thus, we have $(H(1, FID_1))^{b_1} \cdot (H(2, FID_2))^{b_2} = H(X, FID_X)$, in other words, the attacker find a (X, FID_X) which makes $\prod_{i=1}^n e(R_i P^{h(ID_i, T_i)}, \rho'_i) = e(R_i P^{h(ID_i, T_i)}, H(X, FID_X) g^{m'})$, he successes in the attack. However, it contradicts to the assumption that H is a one-way hash function. If sig_1 and sig_2 are generated by two users, the attacker certainly can not generate a correct signature either, which can be proved similarly. Thus, it is clear that the signature algorithm in PSFS can support non-malleability.

Therefore, the signature algorithm in PSFS supports homomorphic authenticable, which meets the requirement of public auditing.

4.2.2 Public Auditing

Public auditing means that the verifier is able to check the correctness of the proof message without knowing the content of files stored in the cloud and the secret key of the signer. Specifically, given a valid proof message $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{FID_j\}_{j \in C}\}$ generated by the cloud with the valid files stored in the cloud, a public verifier is able to verify the integrity of files stored in the cloud. Based on properties of homomorphic authentication, the correctness of Eq. 1 can be proved as follow.

$$\begin{aligned}
 e\left(g^\mu \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) &= e\left(g^{\kappa+\mu'} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) \\
 &= e\left(g^{\kappa+\sum_{j \in C} v_j m_j} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) = e\left(g^{\kappa} \cdot g^{\sum_{j \in C} v_j m_j} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) \\
 &= e(g^\kappa, g) \cdot e\left(g^{\sum_{j \in C} v_j m_j} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) = e(K, g) e\left(\prod_{j \in C} g^{v_j m_j} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) \\
 &= e(K, g) \prod_{j \in C} \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, \rho_{j,i}^{v_j}\right) = e(K, g) \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, \prod_{j \in C} \rho_{j,i}^{v_j}\right) \\
 &= e(K, g) \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, o_i\right)
 \end{aligned}$$

It's clear that a correct proof message can pass the verification and the verifier does not need to use the secret key of signer.

4.2.3 Identity Privacy and Data Privacy

Identity privacy is guaranteed if no one can guess the identity of the party who generates the signature, and data privacy make the verifier learns no knowledge from the proof message he received.

Identity privacy Given a signature $\rho_y = \{\rho_{y,1}, \dots, \rho_{y,n}, c_y\}$, the probability of getting the block signed by the signer from $\rho_{y,1}, \dots, \rho_{y,n}$, is $1/n$ and $1/(n-1)$ for group member and non-member respectively. In addition, c signatures are randomly chosen in the integrity auditing, so the probability becomes about $1/n^c$. What's more, even an attacker finds the right signature block generated with the signer's secret key, he cannot get the identity message. Details of this proof can be found in [11]. In addition, an attacker can get identity information from c_y . Due to the technology of secret share [21], he cannot recover the public key of the signer even he obtains all public key sharings. However, when t members work together, they can trace the signer's public key, which will be proven later. Otherwise, retrieving the identity of the signer on each block during the auditing process is as hard as solving DL problem in group G_1 . Therefore, signer's identity privacy is protected in PSFS.

Data privacy An attacker can get the proof message $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{FID_j\}_{j \in C}\}$ generated by the cloud, where $\mu = \kappa + \mu'$ has been blinded by random number. If a public verifier get the combined message $\sum_{j \in C} v_j m_j$, he can get the content of data by collecting a sufficient number of linear combinations [11]. However, PSFS only transmits $\mu = \kappa + \mu'$, which masks $\sum_{j \in C} v_j m_j$ with a random element κ by random masking technology. In order to solve bilinear equations, the public verifier must get the value of the random κ . Given $K = g^\kappa$, the public verifier cannot obtain κ as obtaining κ from K is as hard as solving the DL problem in G_1 , which is computationally infeasible. In other words, the attacker cannot get any m_j by solving linear equations by collecting a sufficient number of linear combinations. Therefore, PSFS protects data privacy.

4.2.4 Traceability

Due to the technology of secret share [21], no less than t members work together to reconstruct some information for the revealing of the signer identity. The details of tracing method can be found in 4.1. Unlike other group signature public auditing scheme [9], our scheme's traceability has (t, n) -threshold property, which means that even $t - 1$ group members have been corrupted by an adversary, the scheme is remain secure. And any subset of more than t members can jointly reveal the identity of the signer. Details of this proof can be found in (t, n) threshold mechanism [25].

4.2.5 Unforgeability

In a public auditing scheme, unforgeability should be guaranteed in two parts. First, an attack without the correct secret key cannot generate a valid signature. Second, the cloud is unable to provide a proof message without using the corresponding file blocks and signatures. As we consulted the IAID [12] and made some necessary changes in the construction of the proposed public auditing scheme, the unforgeability of signature scheme is guaranteed. Thus, we only prove that the cloud cannot forge a proof message without using the corresponding file blocks. Assume that the cloud forge a valid proof message $\{\{o_i\}_{1 \leq i \leq n}, K, \underline{\mu}, \{FID_j\}_{j \in C}\}$ according to the challenge message $chall = \{(j, v_j)\}_{j \in C}$. Then, we have

$$e\left(g^{\underline{\mu}} \prod_{j \in C} H(j, FID_j)^{v_j}, g\right) = e(K, g) \prod_{1 \leq i \leq n} e\left(R_i P^{h(ID_i, T_i)}, o_i\right).$$

Therefore, we have $e(g^{m_i} H(j, FID_j), g) = \prod_{1 \leq i \leq n} e(R_i P^{h(ID_i, T_i)}, \rho_{j,i})$, which contradicts the unforgeability in the signature scheme. In other words, it is infeasible to retrieve the valid block-tag pairs if the cloud has modified the message in $\mu = \kappa + \mu'$.

5 Performance Analysis

In this section, we evaluate the performance of the proposed public auditing scheme PSFS by calculating the overhead of communication and computation. Then we show the overhead comparisons between PSFS and schemes Oruta [11] and IAID [12]. To make

readers understand easily, the notations of operations used in this section are listed in Table 2.

5.1 Computation Cost

During an auditing task in PSFS, the total computation overhead is about $(n + 2)Pair + cHash + nhash + cMul_{Z_q} + (c + 1)Exp_{Z_q} + (nc - n + c - 1)Mul_{G_1} + (nc + n + 1)Exp_{G_1} + nMul_{G_2}$.

Specifically, in **ProofGen**, with the challenge message $chall = \{(j, v_j)\}_{j \in C}$ generated by a public verifier, the cloud generates the corresponding audit proof message $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{I_j\}_{j \in C}\}$ and sends it to the verifier. In this step, the cloud needs to calculate $\mu' = \sum_{j \in C} v_j m_j$, all the blocks' tags $o_i = \prod_{j \in C} \rho_{j,i}^{v_j} \in G_1, 1 \leq i \leq n$, $K = g^\kappa$ and $\mu = \kappa + \mu$. In this process, the computation overhead of generating audit proof is $cMul_{Z_q} + n(c - 1)Mul_{G_1} + (nc + 1)Exp_{G_1}$. Upon receiving the proof message, the verifier checks the correctness with computation overhead $(n + 2)Pair + cHash + nhash + (c + 1)Exp_{Z_q} + (c - 1)Mul_{G_1} + nExp_{G_1} + nMul_{G_2}$. The detailed descriptions of symbols before have been listed in Table 2. We exhibit the computation cost of schemes Oruta [11], IAID [12] and PSFS in Table 3. Noting that Oruta [11] further splits a file block into d blocks.

5.2 Communication Cost

In **ChallGen**, a verifier needs to send a challenge message $chall = \{(j, v_j)\}_{j \in C}$ to the cloud. For each auditing challenge, the cost is $2c|Z_q|$ bits, where $||$ represents the length of the content in $||$ and c is the number of challenged blocks in shared data. Then, in **ProofGen**, the cloud needs to send audit proof $\{\{o_i\}_{1 \leq i \leq n}, K, \mu, \{I_j\}_{j \in C}\}$ to the verifier and the communication cost is $(n + 1)|G_1| + (c + 1)|Z_q|$ bits. Therefore, the total communication overhead is $(n + 1)|G_1| + (3c + 1)|Z_q|$ bits during an auditing task in PSFS. We compare it with schemes Oruta [11], IAID [12] in Table 4.

From Table 4, we can find that communication cost in these three schemes are all linearly related to the size of group, where Oruta and PSFS needs to transmit more

Table 2 Notation of cryptographic operations

Notation	Descriptions
Exp_{G_1}, Exp_{Z_q}	Exponentiation in G_1, Z_q
$Mul_{G_1}, Mul_{Z_q}, Mul_{G_2}$	Multiplication in G_1, Z_q, G_2
$Hash$	Hash operation in G_1
$hash$	Hash operation in Z_q
$Pair$	Pair operation
f	Pseudo-random function
π	Pseudo-random permutation
c	Numbers of auditing blocks
n	Number of group users

Table 3 Computation cost comparison

Scheme	Computation cost
Oruta [11]	$(n + 2)Pair + cHash + dhash + d(c + 1)Mul_{Z_q} + (c + d)Exp_{Z_q}$ $+ (n + 2d - 2)Mul_{G_1}$ $+ (2d + n)Exp_{G_1} + nMul_{G_2}$
IAID [12]	$2cf + 2c\pi + (n + 1)Pair + cHash + nhash$ $+ cMul_{Z_q} + (nc + c - 1)Mul_{G_1} + (nc + n + c + 1)Exp_{G_1}$ $+ (n - 1)Mul_{G_2}$
PSFS	$(n + 2)Pair + cHash + nhash + cMul_{Z_q}$ $+ (c + 1)Exp_{Z_q} + (nc - n + c - 1)Mul_{G_1}$ $+ (nc + n + 1)Exp_{G_1} + nMul_{G_2}$

elements than IAID. The reason is that IAID uses a pseudo-random function and a pseudo-random permutation to compute challenge message sets instead of selecting two sets and then sending them. Noting that the communication cost of Oruta and PSFS is the same if $d = 1$, when Oruta does not further split the file block.

5.3 Experimental Results

In the following experiments, we utilize the Pairing Based Cryptography (PBC) [26] library to simulate the cryptographic operations in HA-DGSP, and all experiments are tested on Ubuntu system with Intel Core i7 3.40 GHz over 1000 times. We assume $|Z_q| = 160\text{bit}$, $|G_1| = 160\text{ bit}$, the number of users in the group is $n = 100$. Every block has d elements where $d = 10$. According to previous work [27], to keep detection probability greater than 99%, we set $c = 460$, the number of selected blocks in an auditing task.

Table 5 presents the comparison of experimental results between schemes Oruta, IAID and PSFS. It is clear to find that in Oruta, the overheads are linear correlated with d , the number of blocks. But in IAID and PSFS, the overhead of the auditing is independent of d . The computation overhead of IAID is similar to that of PSFS but is larger than that in Oruta. However, IAID pointed that Oruta cannot against group member changing attack. The communication overhead of IAID slightly smaller because it uses pseudo-random function to generate random numbers instead of transmitting the random number set. Compare to the storage of file, the communication overhead is acceptable. The computation overhead of these three schemes are similar to each other. However, neither Oruta nor IAID supports identity traceability and secure data sharing which are the feature properties of PSFS.

6 Conclusions

In this paper, we proposed a practical and secure file sharing scheme for non-manager group based on public auditing, PSFS for short, which solves the disadvantage of signer identity privacy in HA-DGSP [13], and supports non-manager groups to share data in the cloud securely. PSFS protects the privacy of signer identity and file, which makes verifier cannot get any information of the signer or file during the verification. PSFS also provides identity traceability but it is independent of any centralized control as each member is provided with equal power and t group members can work together to trace the identity of the signer if there is something wrong with the file. Noting that PSFS is the first practical

Table 4 Communication cost comparison

Scheme	Communication cost (KB)
Oruta [11]	$(n + d) G_1 + (3c + d) Z_q $
IAID [12]	$n G_1 + (c + 5) Z_q $
PSFS	$(n + 1) G_1 + (3c + 1) Z_q $

Table 5 Experiment results comparison

Scheme	Computation cost (s)		Communication cost (KB)	
	$d = 1$	$d = 10$	$d = 1$	$d = 10$
Oruta [11]	3.371	3.454	28.945	29.297
IAID [12]	4.092	4.092	11.035	11.035
PSFS	4.100	4.100	28.945	28.945

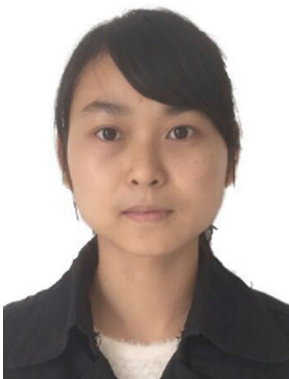
data sharing scheme in cloud storage, which only provides access interface to protect file, and effective incentive for data sharing based on the technology of blockchain. From performance analysis, we can see that the communication overhead and computational overhead of PSFS is acceptable. However, PSFS does not support dynamic group operations, which is a disadvantage in public auditing schemes based on ring signature. Thus, we will further discuss group dynamic and data processing in public auditing based on blockchain in our future research.

Acknowledgements This work is supported by National Science Foundation of China (61572255), Six talent peaks project of Jiangsu Province, China (XYDXXJS-032), CERNET Innovation Project (NGII20170205). We would like to appreciate the anonymous referees for their helpful comments.

References

1. Yang, H. S., & Yoo, S. J. (2015). A study on smartwork security technology based on cloud computing environment. *Wireless Personal Communications*, 94(3), 1–10.
2. Yuan, J., & Yu, S. (2015). Public integrity auditing for dynamic data sharing with multiuser modification. *IEEE Transactions on Information Forensics and Security*, 10(8), 1717–1726.
3. Huang, L., Zhang, G., & Fu, A. (2016). Privacy-preserving public auditing for dynamic group based on hierarchical tree. *Journal of Computer Research and Development*, 53(10), 2334–2342.
4. Yu, S. (2017). Big privacy: Challenges and opportunities of privacy study in the age of big data. *IEEE Access*, 4, 2751–2763.
5. Li, X., Kumari, S., Shen, J., Wu, F., & Chen, C. (2017). Secure data access and sharing scheme for cloud storage. *Wireless Personal Communications*, 96(4), 5295–5314.
6. Wang, C., Chow, S. S. M., Wang, Q., Ren, K., & Lou, W. (2013). Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers*, 62(2), 362–375.
7. Huang, L., Zhang, G., & Fu, A. (2017). Certificateless public verification scheme with privacy-preserving and message recovery for dynamic group. In *Australasian computer science week multiconference* (p. 76). ACM.
8. Li, J., Zhang, L., Liu, J. K., Qian, H., & Dong, Z. (2016). Privacy-preserving public auditing protocol for low-performance end devices in cloud. *IEEE Transactions on Information Forensics and Security*, 11(11), 2572–2583.
9. Wang, B., Li, B., & Li, H. (2012). Knox: Privacy-preserving auditing for shared data with large groups in the cloud. In *International conference on applied cryptography and network security* (pp. 507–525). Springer.

10. Li, H., Sun, W., Li, F., & Wang, B. (2014). Secure and privacy-preserving data storage service in public cloud. *Journal of Computer Research & Development*, 51(7), 1397–1409.
11. Wang, B., Li, B., & Li, H. (2014). Oruta: Privacy-preserving public auditing for shared data in the cloud. *IEEE Transactions on Cloud Computing*, 2(1), 43–56.
12. Wang, H., He, D., Yu, J., & Wang, Z. (2016). Incentive and unconditionally anonymous identity-based public provable data possession. *IEEE Transactions on Services Computing*, PP(99), 1.
13. Huang, L., Zhang, G., & Fu, A. (2017). Privacy-preserving public auditing for non-manager group. In *IEEE international conference on communications* (pp. 1–6). IEEE.
14. Fu, A., Yu, S., Zhang, Y., Wang, H., & Huang, C. (2017). NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users. *IEEE Transactions on Big Data*, PP(99), 1.
15. Yang, G., Yu, J., Shen, W., Su, Q., Fu, Z., & Hao, R. (2016). Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability. *Journal of Systems & Software*, 113(C), 130–139.
16. Yu, Y., Man, H. A., Mu, Y., Tang, S., & Ren, J. (2015). Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage. *International Journal of Information Security*, 14(4), 307–318.
17. Zhang, J., Li, P., & Mao, J. (2015). An oriented-group supporting multi-user public auditing for data sharing. In *IEEE international conference on smart city* (pp. 996–1002). IEEE.
18. Achhra, A., Vaswani, P., Agale, R., & Chheda, M. (2015). Public auditing for the shared data in the cloud. *International Journal of Advance Foundation and Research in Computer*, 2(4), 125–129.
19. Christidis, K., & Devetsikiotis, M. (2016). Blockchains and smart contracts for the internet of things. *IEEE Access*, 4, 2292–2303.
20. Zyskind, G., Nathan, O., Pentland, A. (2015). Decentralizing privacy: Using blockchain to protect personal data. *IEEE security and privacy workshops* (pp. 180–184). IEEE Computer Society.
21. Blömer, J. (2011). How to share a secret. *Communications of the ACM*, 22(11), 612–613.
22. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., et al. (2007). Provable data possession at untrusted stores. In *ACM conference on computer and communications security* (pp. 598–609). ACM.
23. Juels, A., & Kaliski, B. S. (2007). Pors: Proofs of retrievability for large files. In *ACM conference on computer and communications security* (pp. 584–597). ACM.
24. Fu, A., Li, Y., Yu, S., Yu, Y., & Zhang, G. (2018). DIPOR: An IDA-based dynamic proof of retrievability scheme for cloud storage systems. *Journal of Network & Computer Applications*, 104, 97–106.
25. Li, X., Qian, H., & Li, J. (2011). Democratic group signatures with threshold traceability. *Journal of Shanghai Jiaotong University*, 16(5), 530–532.
26. Lynn, B. (2012). *The pairing-based cryptography (pbc) library*. <http://crypto.stanford.edu/pbc>.
27. Huang, L., Zhang, G., Yu, S., Fu, A., & Yearwood, J. (2017). SeShare: Secure cloud data sharing based on blockchain and public auditing. *Concurrency & Computation Practice & Experience*. <https://doi.org/10.1002/cpe.4359>.



Longxia Huang is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, School of Computer Science and Engineering, Nanjing University of Science and Technology, China. She received her B.S. degree in Communication Engineering from the same university, in 2013. Her research interests include blockchain technology, privacy protection and public auditing in cloud storage.



Gongxuan Zhang received the B.S. degree in electronic computer from Tianjin University in 1983 and the M.S. and Ph.D. degrees in Computer Application from the Nanjing University of Science and Technology in 1991 and 2005, respectively. He was a Senior Visiting Scholar in Royal Melbourne Institute of Technology from 2001.9 to 2002.3 and University of Notre Dame from 2017.7 to 2017.10. Since 1991, he has been with the Nanjing University of Science and Technology, where he is currently a professor in the School of Computer Science and Engineering. His current research interests include multicore and parallel processing and distributed computing, and cyber space security. He has served on the program committees in a couple of conferences. He has over 80 publications and has led or participated in 40 research projects supported by the National Science Foundation of China, and the Provincial Science Foundation of Jiangsu.



Anmin Fu is currently an associate professor and supervisor of Ph.D. students of Nanjing University of Science and Technology, China. He received his B.S. degree in Communication Engineering from Lanzhou University of Technology, China, in 2005. He received his M.S. and Ph.D. degrees in Cryptography and Information Security from Xidian University in 2008 and 2011, respectively. His research interests include wireless security and cryptography.