Access-Authorizing and Privacy-Preserving Auditing with Group Dynamic for Shared Cloud Data

Wenting Shen[1], Jia Yu[1,2,3], Guangyang Yang[1], Yue Zhang[1], Zhangjie Fu[3], and Rong Hao[1]
[1]College of Information Engineering, Qingdao University
Qingdao, 266071, China
[2] Institute of Big Data Technology and Smart City, Qingdao University
Qingdao, 266071, China
[3]School of Computer and Software, Nanjing University of Information Science & Technology,
Nanjing, 210044, China
[e-mail: qduyujia@gmail.com]
*Corresponding author: Jia Yu

Received July 24, 2015; revised November 5, 2015; accepted June 9, 2016; published July 31, 2016

Abstract

Cloud storage is becoming more and more popular because of its elasticity and pay-as-you-go storage service manner. In some cloud storage scenarios, the data that are stored in the cloud may be shared by a group of users. To verify the integrity of cloud data in this kind of applications, many auditing schemes for shared cloud data have been proposed. However, all of these schemes do not consider the access authorization problem for users, which makes the revoked users still able to access the shared cloud data belonging to the group. In order to deal with this problem, we propose a novel public auditing scheme for shared cloud data in this paper. Different from previous work, in our scheme, the user in a group cannot any longer access the shared cloud data belonging to this group once this user is revoked. In addition, we propose a new random masking technique to make our scheme preserve both data privacy and identity privacy. Furthermore, our scheme supports to enroll a new user in a group and revoke an old user from a group. We analyze the security of the proposed scheme and justify its performance by concrete implementations.

Keywords: Cloud Storage, Public Auditing, Data Privacy, Batch Verification

http://dx.doi.org/10.3837/tiis.2016.07.025 **ISSN : 1976-7277**

1. Introduction

Cloud storage is one of the most important online storage models. It provides users with easy data access and pay-as-you-go storage service. Outsourcing data to the cloud helps users avoid investing a lot of money in maintaining the local hardware/software and data. Although cloud storage brings many benefits to users, it also raises several security concerns. The data stored in the cloud might be lost or corrupted due to the inevitable hardware/software failures and human errors [1-4]. What is more serious, the cloud might deliberately delete the data which are rarely used for saving storage space, and hide the fact that data are lost for maintaining its reputation [5]. So it is very necessary to check the integrity of data stored in the cloud.

In order to verify the integrity of data in the cloud, the notion of the cloud storage auditing has been proposed [6-23]. In a cloud storage auditing scheme, auditing task can be carried out by a data owner or a third party auditor (TPA). The cloud storage auditing scheme allows to verify the integrity of data efficiently without downloading the entire data from the cloud. Many cloud storage auditing schemes focusing on different aspects have been proposed [10-23]. If TPA challenges the same data blocks several times during the data auditing, he/she might derive the contents of user data. In order to deal with this problem, Wang et al. [10] utilized homomorphic linear authenticators and random masking technique to guarantee that TPA would not know anything about data content; that is to say, the users' data were not leaked during the process of auditing. Solomon et al. [11] further proposed a privacy-preserving auditing scheme which had the same security level as the scheme in [10] but had better efficiency. In addition, the data stored in the cloud might be updated frequently by users for various applications. In order to satisfy this requirement, Erway et al. [12] proposed the first data auditing scheme supporting dynamic data updates based on skip list structure. Zhu et al. [13] used index hash tables to construct a data auditing scheme supporting data dynamic operations. Wang et al. [14] proposed a data auditing scheme supporting dynamic data updates based on Merkle hash tree. Mo et al. [15] further improved the scheme in [14], and proposed a data auditing scheme supporting dynamic data updates based on Merkle hash tree and B+ tree, which had better efficiency than the scheme in [14]. Ranked Merkle hash tree [16] and balanced update tree [17] can be used to improve efficiency of dynamic data updates. If the secret key for cloud storage auditing is exposed, it can cause the serious security problem. In order to deal with this problem, Yu et al. [18] proposed the first practical auditing protocol with built-in key-exposure resilience for cloud storage based on binary tree in [19].

Sometimes, the data are not only stored in the cloud, but also shared across multiple users in some cloud data storage applications, such as iCloud, Google Drive and Dropbox. These applications allow a number of users to work together as a group by sharing data each other.

Once one of these users in the group uploads shared data to the cloud, the rest users of the group can access these shared cloud data. In above scenario, data auditing is still necessary to assure the integrity of shared cloud data. The auditing mechanism in [24] is built to support shared cloud data dynamic by using leverage index hash tables. With the technique of proxy re-signature, the auditing scheme for shared cloud data proposed in [25] not only supports group dynamic (user enrollment and user revocation) but also supports identity privacy. Yuan et al. [26] designed a polynomial-based authentication tags and a proxy tag update technique to implement an efficient and scalable public data checking scheme with multi-user revocation. Wang et al. [27] designed an auditing scheme for shared cloud data based on ring signatures, in which TPA checks the integrity of shared cloud data but he/she cannot know the identity of actual signer in auditing process.

Observing all previous auditing schemes for shared cloud data mentioned above, there exist the following problems:

(1) The above mentioned auditing schemes [24-27] for shared cloud data do not consider the problem of user access authorization. If a user is revoked from a group, although the signatures generated by this revoked user are not valid any more, the cloud does not know which user in the group is revoked [24-27]. It leads to this revoked user still being able to access shared cloud data. Therefore, how to achieve user access authorization is a further worth research.

(2) The above mentioned auditing schemes [24-27] for shared cloud data do not realize the necessity of the auditing authentication for TPA when shared cloud data are challenged. As Liu et al. [16] mentioned, it was necessary to add auditing authentication process between the auditor and the cloud. Adding auditing authentication for TPA is to eliminate the threat of unauthorized auditing challenges from malicious or pretended TPA. Malicious or pretended TPA can challenge data stored in the cloud without the group's permission, which might cause the cloud spending a lot of computing resources in responding to these auditing challenges. In the auditing schemes for shared cloud data, there also exists above problem. Therefore, it is essential to verify whether the TPA is authorized in the auditing scheme for shared cloud data.

(3) Very few auditing schemes for shared cloud data supporting data privacy and identity privacy simultaneously. Both identity and data content are confidential information for users, so they might be unwilling to reveal their identity and data content to TPA. For example, in electronic medical, a patient may agree medical researchers to analyze his/her health record stored in the cloud for research purpose, but this patient is unwilling to disclose his/her identity and data content to other people. Therefore, supporting both data privacy and identity privacy is very vital in many applications.

In order to deal with above problems, we design a novel auditing scheme for shared cloud data in this paper. The contributions of this paper can be summarized as follows:

(1) We firstly consider the user access authorization in the auditing for shared cloud data. To

efficiently achieve user access authorization, we design an efficient mechanism in which the cloud keeps an access authorization credential (one hash value of group private key and group public key) from group manager. Only the users in the group can provide the valid access authorization credential to access the shared data in the cloud. When a user is revoked from the group, the access authorization credential would be updated. Because this revoked user does not know this new access authorization credential, he/she cannot access shared cloud data any longer. So it solves the problem that the revoked user can still access shared cloud data.

(2) We add TPA auditing authorization mechanism in our scheme for avoiding the cloud responding to the unauthorized auditing challenges from malicious TPA. In our scheme, we can ensure that only the TPA authorized by group manager can receive the auditing proof from the cloud. That is to say, if a TPA who sends an auditing challenge to the cloud has a valid auditing authorization from group manager, the cloud will generate an auditing proof as the response; otherwise, will not.

(3) In addition, our scheme not only supports the group dynamic, but also preserves data privacy and identity privacy simultaneously. In order to preserve data privacy, our scheme utilizes a new random masking technique, which makes TPA unable to derive the content of data from the cloud's response. In order to preserve identity privacy, our scheme uses a common group privacy key to calculate signatures on all data blocks, which makes TPA cannot know the identity of actual signer.

(4) Finally, we also extend the proposed scheme to support batch auditing, which can efficiently perform multiple auditing tasks simultaneously from different groups.

Organization. The rest of this paper is organized as follows: Section 2 presents the system model and the design goals. We introduce some simple definitions in Section 3. The detailed description of the proposed scheme and the scheme supporting batch auditing are introduced in Section 4 and Section 5, respectively. In Section 6, we provide the security analysis of the proposed scheme. The evaluation of performance is shown in Section 7. Finally, we conclude our paper in Section 8.

2. System Model and Design Goals

2.1. System Model

The system model involves three kinds of different entities: the cloud, users and the third party auditor (TPA), as shown in **Fig. 1**. The cloud provides data storage and data sharing service to users. In a group, there are multiple users. One of these users is regarded as a special one, who is named as group manager. The differences between the group manager and the other group users are as follows: Firstly, the group manager is in charge of generating the group public/private key and distributing group private key to every user in the group. Secondly, he is responsible for computing and sending the access authorization

credential to the cloud. The cloud verifies whether the user who accesses the shared cloud data is legal or not by this access authorization credential. Thirdly, he is also in charge of generating and sending the group auditing authorization to TPA. When a user in the group who provides the valid access authorization credential, he/she can create and share data with other group users in the cloud, and moreover, can access other shared cloud data. TPA is a public verifier who is delegated by group manager to audit the integrity of shared cloud data.

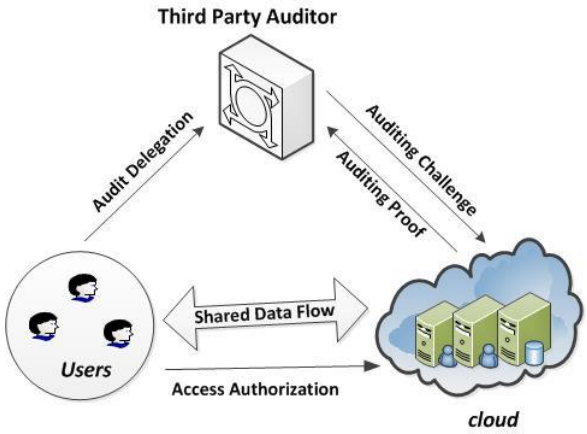

Fig. 1. The system model

In our paper, only the users able to provide the valid access authorization credential can access shared data in the cloud. When group manager wants to check the integrity of shared cloud data, he/she will give an auditing authorization to TPA. After TPA verifies the auditing authorization from group manager is valid, he/she sends an auditing challenge along with the auditing authorization to the cloud. After receiving these messages, the cloud needs to verify whether this TPA is indeed authorized by group manager. If it is, the cloud will respond to this TPA with a proof of shared cloud data possession; otherwise, will not. Finally, TPA will check the correctness of the proof to verify the integrity of shared cloud data.

2.2. Design Goals

To efficiently check the integrity of shared cloud data, our scheme should be designed to achieve the following properties:

1. The auditing authorization of TPA: to ensure that TPA who sends auditing challenge to the cloud is authorized by group manager.
2. Group dynamic: to allow users join or leave a group.
3. The access authorization of user: to determine that only the user with a valid access authorization credential can upload or access shared data in the cloud.
4. Privacy preserving: to assure that user identity and data content are not leaked to TPA during the auditing process.

5. Public auditing: to allow TPA to audit the integrity of shared cloud data on demand without retrieving the entire data.
6. Batch auditing: to enable TPA to perform multiple auditing tasks simultaneously from different groups.

3. Definition

3.1 Bilinear Maps

Let G_1, G_2 be two multiplicative cyclic groups of prime order p, and g be a generator of G_1. A bilinear map e is a map $e: G_1 \times G_1 \to G_2$ with the following properties:

1) *Computability*: there exists an efficiently computable algorithm for computing map $e: G_1 \times G_1 \to G_2$.
2) *Bilinearity*: for all $u, v \in G_1$ and $a, b \in Z_p^*$, $e(u^a, v^b) = e(u,v)^{ab}$.
3) *Non-degeneracy*: $e(g,g) \neq 1$.

3.2 Computational Diffie-Hellman (CDH) Problem

For x, $y \in Z_p^*$, given g, g^x and $g^y \in G_1$ as input, outputs $g^{xy} \in G_1$. The CDH assumption in G_1 holds if it is computationally infeasible to solve the CDH problem in G_1.

3.3 Discrete Logarithm (DL) Problem

For $x \in Z_p^*$, given g, $g^x \in G_1$ as input, outputs x. The DL assumption in G_1 holds if it is computationally infeasible to solve the DL problem in G_1.

3.4 Access-Authorizing and Privacy-Preserving Auditing Scheme with Group Dynamic for Shared Cloud Data

An access-authorizing and privacy-preserving auditing scheme with group dynamic for shared cloud data includes seven algorithms: *KeyGen*, *SigGen*, *Join*, *Revoke*, *Resign*, *ProofGen*, and *ProofVerify*:

1. $KeyGen(1^k)$: This algorithm is run by group manager. It takes as input a security parameter k, and generates a public-private key pair (pk, sk).
2. $SigGen(sk, name, F)$: This algorithm is run by user to process shared cloud data. It takes as input the private key sk, the file identifier $name$ and an ordered collection of data blocks $\{m_i\}_{i\in[1,n]}$, and generates a signature set Φ, which is an ordered collection of signatures $\{\sigma_i\}_{1\le i\le n}$ on blocks $\{m_i\}_{1\le i\le n}$ in shared cloud data.
3. $Join(U)$: This algorithm is operated by group manager. It takes as input a new user U, then group manager sends the private key sk to this new user U.
4. $Revoke(U)$: This algorithm is executed by group manager when a user is revoked from the group. It takes as input the revoked user U, then group manager delivers an update

key to each user in the group except the revoked user U. The update key is used to recalculate the data blocks in algorithm *Resign*.

5. *Resign*(uk): This algorithm is operated by the cloud. It takes as input an update key uk, and generates a new signature set Φ' on all data blocks. Once a user is revoked from the group, the previous signatures stored in the cloud need to be recomputed and updated by the update key uk.
6. *ProofGen*($F, \Phi, chal$): This algorithm is executed by the cloud. It takes as input the shared cloud data file F, the corresponding signature set Φ and the auditing challenge $chal$, and generates a proof P which can demonstrate that the cloud truly possesses shared cloud data.
7. *VerifyProof* ($pk, chal, P$): This algorithm is run by TPA. It takes as input the public key pk, the auditing challenge $chal$ and the proof P. TPA verifies whether the proof P is valid or not.

4. The Proposed Scheme

4.1 Notation

Let G_1 and G_2 be two multiplicative cyclic groups of prime order p, g be a generator of G_1, $e: G_1 \times G_1 \to G_2$ be a bilinear map, and u be a random generator of G_1. Let $H: \{0,1\}^* \to G_1$, $h_1(\cdot): Z_p^* \times G_1 \to Z_p^*$ and $h_2(\cdot): G_1 \to Z_p^*$ be three cryptographic hash functions. The global parameters are $(G_1, G_2, p, e, g, u, H, h_1, h_2)$. Shared cloud data M are divided into n blocks $(m_1, ..., m_n)$. Assume the total number of users in the group is d.

4.2 Description of the Scheme

(1) Algorithm *KeyGen*(1^k)

① The group manager generates a random signing key pair $\{ssk, spk\}$. Then he/she chooses a random value $x \in Z_p^*$ as the group private key, and computes $v = g^x$ as the group public key, and computes $h_1(x, v)$ as the access authorization credential. We define $\{x, ssk\}$ as the private key sk, and $\{v, spk\}$ as the public key pk.

② The group manager distributes the private key sk to all the users in the group, and sends access authorization credential $h_1(x, v)$ to the cloud through a secure channel. The user who wants to access shared cloud data needs to compute access authorization credential $h_1(x, v)$ according to the group private key x. The cloud will verify whether the user provides a valid access authorization credential $h_1(x, v)$ when a user asks to access the shared cloud data.

(2) Algorithm *SigGen*($F, sk, name$)

① For each block $m_i \in Z_p^*$ ($i \in [1, n]$), user generates signature σ_i on block m_i with

the group private key x as follows: $\sigma_i = (H(i \| name) \cdot u^{m_i})^x$, where $name \in Z_p^*$ is a random value which is chosen as the identifier of the file by the group. Let $\Phi = \{\sigma_i\}_{1 \le i \le n}$ be a set of signatures.

② The user calculates the tag of file by computing $t = name \| Sig_{ssk}(name)$, where $Sig_{ssk}(name)$ is the signature on $name$ under the signing private key ssk.

③ The user sends $\{h_1(x,v), F, \Phi, t\}$ to the cloud, and deletes the file F and its corresponding set of signatures from local storage.

(3) Algorithm $Join(U)$

When a new user U joins a group, the group manager sends the private key sk to this new user U. Then this new user U calculates the access authorization credential $h_1(x,v)$ according to the private key sk.

(4) Algorithm $Revoke(U)$

① The group manager chooses a random value $y \in Z_p^*$ as the new group private key, and sets the new private key $sk' = (y, ssk)$.

② The group manager computes update key $uk = y / x \in Z_p^*$, where x is the previous group private key, and then distributes this update key y / x to each user in the group except the revoked user U.

③ After receiving this update key y / x, users in the group calculates the value of y according to the previous group private key x and the update key y / x, and then calculates $h_1(y, g^y)$ according to the value of y, where $h_1(y, g^y)$ is the new access authorization credential for users to access shared cloud data.

④ The group manager sends the new access authorization credential $h_1(y, g^y)$ and the update key y / x to the cloud through a secure channel. The update key y / x is used to recalculate the data blocks in algorithm $ReSign$. Both users and the cloud store the new access authorization credential $h_1(y, g^y)$, then delete the previous access authorization credential $h_1(x,v)$ from their local storage. The cloud will verify whether the user provides a new valid access authorization credential $h_1(y, g^y)$ when a user asks to access the shared cloud data. This revoked user cannot know the new access authorization credential $h_1(y, g^y)$, thus cannot provide it to the cloud and access the shared cloud data any longer.

(5) Algorithm $Resign(uk)$

The cloud recalculates the signatures of cloud data blocks by the update key $uk = y / x$. The new signature of each block in shared cloud data is computed by the cloud as follows: $\sigma_i' = \sigma_i^{uk} = ((H(i \| name) \cdot u^{m_i})^x)^{y/x} = (H(i \| name) \cdot u^{m_i})^y$.

(6) Algorithm $ProofGen(F, \Phi, chal)$

① The group manager sends $AUTH$ to the cloud, where $AUTH$ is random value generated by group manager. Then, group manager computes $sig_{AUTH} = SSig_{ssk}(t \| VID \| AUTH)$ (VID is the identity of TPA), and sends sig_{AUTH}

and t (the tag of file) to the TPA that is authorized by group manager.

② The TPA verifies the validity of the file tag t using spk. TPA will not execute auditing task if the file tag on t is invalid; otherwise, TPA constructs and sends an auditing challenge $chal$ to the cloud. TPA constructs the auditing challenge as follows:

1. Randomly chooses a set of I with c elements, $I \subseteq [1,n]$.
2. Generates a random value $v_i \in Z_p^*$, for each $i \in I$.
3. Outputs an auditing challenge $\{\{i, v_i\}_{i\in I}, \{VID\}_{pk_{cloud}}, sig_{AUTH}\}$, and sends it to the cloud, where $\{VID\}_{pk_{cloud}}$ denotes the encrypted VID using the cloud public key pk_{cloud}.

③ After receiving an auditing challenge from TPA, the cloud decrypts $\{VID\}_{pk_{cloud}}$ with the corresponding cloud secret key, and then uses $AUTH$, t, VID and user signing public key spk to verify whether this TPA is indeed authorized by group manager. If it is, the cloud will respond to this TPA with a proof of shared cloud data possession; otherwise, will not. The cloud generates a proof of shared cloud data possession as follows:

1. Computes $\mu' = \sum_{i\in I} m_i v_i$. For blinding the value of μ', the cloud chooses a random element $r \in Z_p^*$, and then sets $\mu = h_2(R)\mu' - r$, where $R = u^r$.
2. Calculates an aggregated signature $\sigma = \prod_{i\in I} \sigma_i^{v_i}$.
3. Outputs an auditing proof $\{R, \mu, \sigma\}$ to TPA.

(7) Algorithm $VerifyProof(pk, chal, P)$

① Parse $pk = \{v, spk\}$.

② The TPA computes $h_2(R)$ according to R, and then verifies the correctness of auditing proof as:

$$e(\sigma^{h_2(R)}, g) = e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{\mu} \cdot R, v\right) \quad (1)$$

The correctness of the above verification equation can be shown as follows:

$$\begin{aligned}
e(\sigma^{h_2(R)}, g) &= e\left(\prod_{i\in I} \sigma_i^{h_2(R)v_i}, g\right) \\
&= e\left(\prod_{i\in I} (H(i \| name) \cdot u^{m_i})^{x \cdot h_2(R)v_i}, g\right) \\
&= e\left(\prod_{i\in I} \left(H(i \| name)^{h_2(R)v_i} \cdot u^{h_2(R)v_i m_i}\right), g^x\right) \\
&= e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{h_2(R)\sum_{i\in I} m_i v_i}, v\right) \\
&= e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{h_2(R)\mu'}, v\right) \\
&= e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{\mu + r}, v\right) \\
&= e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{\mu} \cdot u^{r}, v\right) \\
&= e\left(\prod_{i\in I} (H(i \| name)^{h_2(R)v_i} \cdot u^{\mu} \cdot R, v\right)
\end{aligned}$$

In *SigGen* algorithm, user calculates the signatures on data blocks which are created by this user with the group private key x, and calculates the tag of file t with the signing private key ssk. When a user is revoked from a group, the group manager generates an update key y/x in *Revoke* algorithm. In *Resign* algorithm, the cloud recomputes and updates previous signatures on all data blocks with this update key y/x. In *ProofGen* algorithm, to verify the integrity of shared cloud data and to avoid the cloud responding to the unauthorized auditing challenges from malicious TPA, we add auditing authorization to TPA. When a group manager wants to check the integrity of shared cloud data, he/she will give an auditing authorization to a specified TPA. Then this specified TPA checks the validity of the file tag t by using spk. Only when the file tag t is valid, can the TPA send auditing challenge to the cloud. The cloud verifies whether the auditing authorization of TPA is valid. If it is, the cloud will respond an auditing proof P to this TPA; otherwise, will not. In *ProofVerify* algorithm, TPA can verify the correctness of the auditing proof P.

In a cloud storage system, the shared data stored in the cloud may be updated for various application purposes [14]. The data dynamic operation includes block level operations of insertion, deletion and modification. Thus, supporting data dynamic operation is very important in public auditing. Now, we present how our scheme supports data dynamic and achieves data privacy protection based on the technique in [14].

In [14], data dynamic is achieved by replacing $H(i\|name)$ with $H(m_i)$ as the tag for block m_i in computing the data signatures and utilizing Merkle hash tree (MHT) to perform block dynamic operation. Therefore, we can use the similar technique to achieve the data dynamic in our scheme. Specifically, every data block signature will be changed into $\sigma_i=(H(m_i)\cdot u^{m_i})^x$. In *SignGen* algorithm, user needs to generate a tree root Ω based on MHT and sends it to TPA for auditing task. In MHT, the leave nodes are an ordered set of hashes of "block tags" $H(m_i)(i\in[1,n])$. In *ProofGen* algorithm, the auditing proof P generated by the cloud not only includes $\{R,\mu,\sigma\}$, but also includes $\{H(m_i)\}_{i\in I}$ and the corresponding auxiliary information $\{\Omega_i\}_{i\in I}$ in the MHT. The cloud's auditing proof P is set as $\{R,\mu,\sigma,\{H(m_i),\Omega_i\}_{i\in I}\}$. When TPA receives the auditing proof from the cloud, he/she computes a tree root Ω based on $\{H(m_i),\Omega_i\}_{i\in I}$ and verifies whether it equals to the tree root he/she has stored. If the both tree roots are same, TPA will verify the correctness of the auditing proof P via equation (1), where $\prod_{i\in I}(H(i\|name)^{v_i}$ is replaced by $\prod_{i\in I}H(m_i)^{v_i}$. All these changes above mentioned have no influence on the random making technique in our scheme. Thus, data privacy is still protected. When a data block is performed for dynamic operation, user needs to generate a new tree root and sends it to TPA for auditing task. The details of performing data dynamic operations are similar to [14]. We do not describe the details of process here. If we use the balanced Merkle hash tree to support data dynamic operation, the efficiency of the scheme is $O(\log n)$.

5. Batch Auditing

In this section, we show how to extend our scheme to support batch auditing, which can handle multiple auditing delegations simultaneously from various group managers of different groups. Specifically, we assume that TPA takes K auditing delegations from K different groups with K diverse data files. The batch auditing scheme achieves the aggregation of K verification equations into a single one with K auditing delegations [10]. The details are described as follows:

(1) Algorithm $KeyGen(1^k)$

Each group manager k ($k \in [1,K]$) generates $sk_k = (x_k, ssk_k)$ as his/her private key and $pk_k = (v_k = g^{x_k}, spk_k)$ as his/her public key. Then, each group manager k ($k \in [1,K]$) distributes the private key sk_k to all users in his/her group, and sends access authorization credential $h_1(x_k, v_k)$ to the cloud through a secure channel. One user belonging to group k ($k \in [1,K]$) who wants to access shared cloud data needs to compute access authorization credential $h_1(x_k, v_k)$ according the group private key x_k. The cloud will verify whether the user provides a valid access authorization credential $h_1(x_k, v_k)$ when a user asks to access the shared cloud data.

(2) Algorithm $SigGen(F = \{F_k\}_{1 \le k \le K}, sk = \{sk_k\}_{1 \le k \le K}, name = \{name_k\}_{1 \le k \le K})$

We assume a group k ($k \in [1,K]$) has a data file $F_k = (m_{k,1}, \cdots, m_{k,n})$ to be outsourced to the cloud. He/She calculates the tag of data file as $t_k = name_k \| Sig_{ssk_k}(name_k)$, where $name_k$ is a random value as the identifier of k data file. He/She chooses $u_k \in G_1$ randomly**,** then computes the signature for every file block $m_{k,i}$ as follows: $\sigma_{k,i} = (H(i \| name_k) \cdot u_k^{m_{k,i}})^{x_k}$, where $i \in [1,n]$ and $k \in [1,K]$. Finally, each group k ($k \in [1,K]$) sends $\{h_1(x_k, v_k), F_k, \Phi_k = \{\sigma_{k,i}\}_{1 \le i \le n}, t_k\}$ to the cloud, and deletes the file F_k and its corresponding set of signatures from local storage.

(3) Algorithm $ProofGen(F = \{F_k\}_{1 \le k \le K}, \Phi = \{\Phi_k\}_{1 \le k \le K}, chal)$

Each group manager k ($k \in [1,K]$) sends $AUTH_k$ to the cloud, where $AUTH_k$ is randomly generated by group manager k. Next, he/she computes $sig_{AUTH_k} = SSig_{ssk_k}(t_k \| VID \| AUTH_k)$ (VID is the identity of TPA), then sends sig_{AUTH_k} and t_k (the tag of file k) to TPA which is authorized. After that, this TPA verifies the validity of the file tag t_k using spk_k, he/she will not execute this auditing task if the file tag t_k is invalid; otherwise, he/she constructs and sends an auditing challenge $chal$ $\{\{i, v_i\}_{i \in I}, \{VID\}_{pk_{cloud}}, sig_{AUTH_k}\}$ to the cloud. The computation of challenge parameters is the same as single user scheme. After receiving an auditing challenge from TPA, the cloud will verify whether this TPA is indeed authorized by group manager k. If it is, the cloud will respond to this TPA with a proof of shared cloud data possession; otherwise, will not. Before generating the proof, the cloud computes $\mu_k' = \sum_{i \in I} m_{k,i} v_i$ and $\sigma_k = \prod_{i \in I} \sigma_{k,i}^{v_i}$ where $k \in [1,K]$, and then randomly chooses $r_k \in Z_p^*$ for each group and sets $\mu_k = h_2(R_k)\mu_k' - r_k$ where $R_k = u_k^{r_k}$. Finally, the cloud outputs an auditing proof

$\{\{R_k\}_{1\le k\le K},\{\mu_k\}_{1\le k\le K},\{\sigma_k\}_{1\le k\le K}\}$ to this TPA.

(4) Algorithm $VerifyProof(pk,chal,P)$

After receiving the proof $\{\{R_k\}_{1\le k\le K},\{\mu_k\}_{1\le k\le K},\{\sigma_k\}_{1\le k\le K}\}$, this TPA computes $h_2(R_k)$, and then verifies the integrity of shared cloud data by checking the following equation :

$$e(\prod_{k\in K}\sigma_k^{h_2(R_k)},g)=\prod_{k\in K}e\left(\prod_{i\in I}H(i\|name_k)^{h_2(R_k)v_i}\cdot u_k^{\mu_k}\cdot R_k,v_k\right)$$

$$\begin{aligned}
e(\prod_{k\in K}\sigma_k^{h_2(R_k)},g)&=e\left(\prod_{k\in K}\left(\prod_{i\in I}\sigma_{k,i}^{v_i}\right)^{h_2(R_k)},g\right)\\
&=\prod_{k\in K}e\left(\prod_{i\in I}(H(i\|name_k)\cdot u_k^{m_{k,i}})^{x_k\cdot h_2(R_k)v_i},g\right)\\
&=\prod_{k\in K}e\left(\prod_{i\in I}(H(i\|name_k)^{h_2(R_k)v_i}\cdot u_k^{h_2(R_k)\sum_{i\in I}m_{k,i}v_i},v_k\right)\\
&=\prod_{k\in K}e\left(\prod_{i\in I}(H(i\|name_k)^{h_2(R_k)v_i}\cdot u_k^{h_2(R_k)\mu_k'},v_k\right)\\
&=\prod_{k\in K}e\left(\prod_{i\in I}(H(i\|name_k)^{h_2(R_k)v_i}\cdot u_k^{\mu_k+r_k},v_k\right)\\
&=\prod_{k\in K}e\left(\prod_{i\in I}H(i\|name_k)^{h_2(R_k)v_i}\cdot u_k^{\mu_k}\cdot R_k,v_k\right)
\end{aligned}$$

(2)

If the above verification equation holds, this TPA believes that all the shared cloud data are correct; otherwise, does not.

The correctness of the verification equation (2) can be shown as follows:

6. Security Analysis

Theorem 1: A *user who is not in a group cannot access shared cloud data belonging to this group in the proposed scheme. The cloud cannot know the group private key* x *from the knowledge of access authorization credential* $h_1(x,v)$.

Proof: In our scheme, only the users in the group can calculate access authorization credential $h_1(x,v)$ by the group private key x. When a user is revoked from the group, group manager will generate an update key y/x, and send it to all the users in the group except the revoked user. These users can compute a new access authorization credential $h_1(y,g^y)$ according to this update key y/x. However, this revoked user cannot forge this new access authorization credential $h_1(y,g^y)$ because he/she does not know the information of the update key y/x. Because the hash function is one-way, the cloud cannot know the group private key x to forge the data authenticators from the knowledge of access authorization credential $h_1(x,v)$. □

Theorem 2: *In the auditing process, the cloud would not generate a proof to respond to*

the unauthorized auditing challenge from malicious TPA, unless this TPA obtains the auditing authorization from group manager.

Proof: TPA can not forge auditing authorization from group manager in our scheme. Because TPA does not know the information of signing private key ssk, he/she can not forge the signature of auditing authorization sig_{AUTH}. If the TPA sends auditing challenge to the cloud, but he/she does not obtain auditing authorization from group manager, the cloud would not generate a proof to respond to this TPA. □

Theorem 3: *From the cloud's response* $\{R, \mu, \sigma\}$, TPA cannot derive the content of users' data μ'.

Proof: In our scheme, the privacy of μ' is guaranteed from μ. We use random masking technique to make μ be blinded by r as $\mu = h_2(R)\mu' - r$, where r is chosen randomly by the cloud and its value is hidden from TPA. Given $u \in G_1$, $R = u^r \in G_1$, computing r is hard, due to the hardness of computational Discrete Logarithm (DL) problem, so the value of r is unknown to TPA. No information of μ' can be known from σ.

From our scheme, it follows that

$$\begin{aligned}\sigma = \prod_{i\in I}\sigma_i^{v_i} &= \prod_{i\in I}(H(i\|name)\cdot u^{m_i})^{xv_i} \\ &= \prod_{i\in I}(H(i\|name)^{v_i})^x \cdot \left(u^{\sum_{i\in I} m_i v_i}\right)^x \\ &= \prod_{i\in I}(H(i\|name)^{v_i})^x \cdot \left(u^{\mu'}\right)^x .\end{aligned}$$

From the above equations, we can see that $\left(u^{\mu'}\right)^x$ is blinded by $\prod_{i\in I}(H(i\|name)^{v_i})^x$. Given $(H(i\|name))^{v_i}$ and g^x, computing $\prod_{i\in I}(H(i\|name)^{v_i})^x$ is hard, due to the hardness of Computational Diffie-Hellman Problem. It tells us that TPA cannot derive the value of $\left(u^{\mu'}\right)^x$, let alone μ'. □

Theorem 4: *For the cloud, it is computationally infeasible to generate a forgery of an auditing proof in our scheme. The cloud passes the verification only if it truly possesses the challenged blocks.*

Proof: Following the security game defined in [7,11], we can prove that, if the cloud could win the following security game, named *Game 1*, by forging an auditing proof on corrupted shared cloud data, then we can solve the Discrete Logarithm (DL) problem in G_1. *Game 1* is described as follows:

Game 1: When TPA sends an auditing challenge $\{\{i, v_i\}_{i\in I}, \{VID\}_{pk_{cloud}}, sig_{AUTH}\}$ to the cloud, the auditing proof $\{R, \mu, \sigma\}$ on correct shared cloud data M is generated, which is able to pass the verification with equation (1). The cloud generates a proof $\{R, \mu^*, \sigma\}$ on

incorrect shared cloud data $M*$. Define $\Delta\mu = \mu* - \mu$. If this invalid proof based on the incorrect shared cloud data $M*$ can successfully pass the verification, then the untrusted cloud wins; otherwise, it fails.

We assume that the cloud wins the game. Then according to equation (1), we have:

$$e(\sigma^{h_2(R)}, g) = e(\prod_{i\in I} H(i \| name)^{h_2(R)v_i} \cdot u^{\mu} \cdot R, v).$$

Because $\{R, \mu^*, \sigma\}$ is a valid auditing proof, we have

$$e(\sigma^{h_2(R)}, g) = e(\prod_{i\in I} H(i \| name)^{h_2(R)v_i} \cdot u^{\mu^*} \cdot R, v).$$

Then, we can learn that

$$u^{\mu} = u^{\mu^*}, \quad u^{\Delta\mu} = 1.$$

For two random elements g, $h \in G_1$, there exists $x \in Z_P^*$ and $h = g^x$ because G_1 is a cyclic group. Without loss of generality, given g, h, set $u = g^{\alpha} h^{\beta} \in G_1$, where α and β are random values of Z_p. Then, we find the solution for the discrete logarithm problem that is,

$$1 = u^{\Delta\mu} = (g^{\alpha} h^{\beta})^{\Delta\mu} = g^{\alpha\cdot\Delta\mu} \cdot h^{\beta\cdot\Delta\mu}.$$

Then the solution to the DL problem is,

$$h = g^{-\frac{\alpha\cdot\Delta\mu}{\beta\cdot\Delta\mu}} = g^{-\frac{\alpha}{\beta}}, \quad x = -\frac{\alpha}{\beta}.$$

Note that β is zero only with the probability $1/p$, which is negligible because p is a large prime. Then, we can find a solution to the DL problem with a probability of $1-1/p$, which contradicts the assumption that the DL problem in G_1 is hard. Therefore, for an untrusted cloud, it is computationally infeasible to generate a forgery of an auditing proof.

7. Performance Evaluation

In this section, we analyze the cost of computation and communication in our scheme, and evaluate the performance in experiments.

7.1 Performance Analysis

When TPA obtains the auditing authorization from group manager, he/she will construct and send an auditing challenge to the cloud. Then, after receiving this auditing challenge, the cloud needs to calculate a proof of shared cloud data possession and sends it to TPA. The cost of computing this auditing proof is $(c-1)Mul_{G_1} + (c+1)Exp_{G_1} + Hash_{z_p^*} + (c+1)Mul_{z_p^*}$

$+(c-1)Add_{z_p^*}+Sub_{z_p^*}$, where Mul_{G_1} denotes the cost of computing one multiplication in G_1, Exp_{G_1} denotes the cost of computing one exponentiation in G_1, $Hash_{Z_p^*}$, $Mul_{Z_p^*}$, $Sub_{Z_p^*}$ and $Add_{Z_p^*}$ denote the cost of computing one hashing operation, computing one multiplication, computing one subtraction and computing one addition in Z_p^*, respectively. When TPA receives the auditing proof from the cloud, he/she will verify the correctness of the proof based on equation (1). The cost of checking this auditing proof is $(c+3)Exp_{G_1}+cHash_{G_1}+(c+1)Mul_{G_1}+2Hash_{Z_p^*}+cMul_{Z_p^*}+2Pair$, where $Hash_{G_1}$ denotes the cost of computing one hashing operation in G_1, $Pair$ denotes the cost of computing one pairing operation in $e:G_1\times G_1\to G_2$.

The cost of communication in our scheme is divided into auditing challenge and auditing proof. The size of an auditing challenge $\{\{i,v_i\}_{i\in I},\{VID\}_{pk_{cloud}},sig_{AUTH}\}$ is $c\cdot(|n|+|p|)+2|p|$ bits, where c is the number of selected blocks, $|n|$ is the size of an element of set $[1,n]$ and $|p|$ is the size of an element of Z_p^*. The size of an auditing proof $\{R,\mu,\sigma\}$ is $2|q|+|p|$ bits, where $|q|$ is the size of an element of G_1. Therefore, the cost of communication is $c\cdot(|n|+|p|)+3|p|+2|q|$ bits in total for an auditing task.

In scheme [10], the computing method of data masking is $\mu=r+h(R)\cdot\mu'$, where $R=e(u,v)^r$. In our scheme, the computing method of data masking is $\mu=h_2(R)\cdot\mu'-r$, where $R=u^r$. Thus, we can know that the scheme in [10] performs one more time-consuming pairing operation which costs more computation overhead in generating proof than our scheme. In scheme [10], the extra cost of user data privacy protection is $Pair+Mul_{z_p^*}+Exp_{G_1}+Hash_{z_p^*}+Add_{z_p^*}$. In our scheme, the extra cost of user data privacy protection is $Mul_{z_p^*}+Exp_{G_1}+Hash_{z_p^*}+Sub_{z_p^*}$. Therefore, our scheme is more efficient.

7.2 Experimental Results

In this section, we evaluate the performance of our scheme in experiments. In our experiments, we utilize the GNU Multiple Precision Arithmetic (GMP) and Pairing-Based Cryptography (PBC) library. All the following experiments are based on C language and tested in Linux OS with an Intel Pentium 2.70GHz processor and 4GB memory. In the experiments, we set the size of the base field to be 512 bits and the size of an element in Z_p^* to be $|p|$=160 bits. The size of shared cloud data we choose is 20MB.

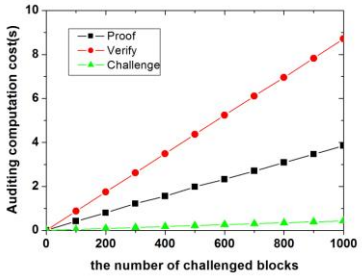

Fig. 2. Computation cost of each algorithm in auditing phase

Performance of Auditing. We analyze the time spent on three phases (challenge, proof, verification) in order to evaluate the overhead of auditing computation in our scheme which is presented in **Fig. 2**. For considering efficiency, we choose to challenge different blocks from 0 to 1000 increased by an interval of 100 in our experiments. In **Fig. 2**, we can see that the running time of challenge is the shortest among the three phases, ranging from 0.0434s to 0.432s. So we can conclude that it spends the least computational cost for generating challenge. The running time of generating the proof ranges from 0.419s to 3.857s. Verifying the correctness of proof costs the largest computation. And it is affected by challenged block number largely, ranging from 0.876s to 8.721s. We can see that the auditing computation time of each phase linearly increases with the number of challenged blocks. So we can infer that when the number of challenged blocks is larger, the cost of computation increases more dramatically in verification phase. Thus, we should find a trade-off between auditing computational cost and integrity guarantee.

Performance of Protecting Data Privacy. In order to demonstrate the process of data privacy protection costs only a little extra computational cost, here, we compare our scheme with Wang et al. 's scheme in [25] which does not support to protect data privacy. In **Fig. 3** and **Fig. 4**, we can see that the communicational cost of proof generation and proof verification in our scheme is only slightly higher than [25]. Thus, we can conclude that our scheme achieves data privacy protection only with acceptable computing overhead.

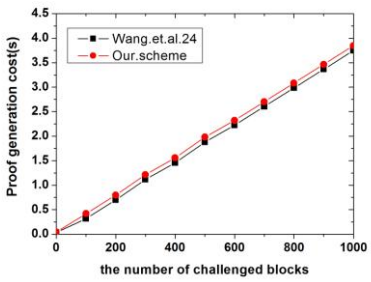

Fig. 3. Comparison of the cost of proof generation between our scheme and scheme[25]

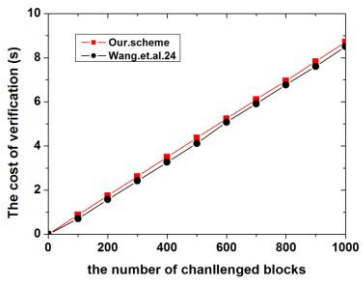

Fig. 4. Comparison of the cost of verification between our scheme and scheme[25]

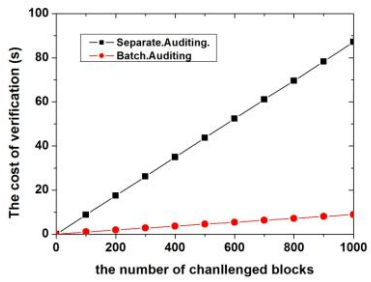

Fig. 5. Comparison of the cost of verification between separated auditing and batch auditing (K=10)

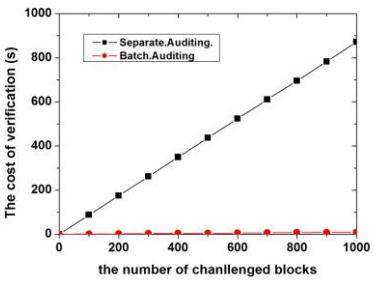

Fig. 6. Comparison of the cost of verification between separated auditing and batch auditing (K=100)

Performance of Batch Auditing. In our experiments, we set the number of auditing delegations to be $K=10$ and $K=100$, which means TPA performs ten and one hundred auditing tasks. In this experiment, we compare separated auditing (TPA performs ten and one hundred auditing tasks separately) with batch auditing (TPA performs ten and one hundred auditing tasks simultaneously). Comparing **Fig. 5** with **Fig. 6**, we can see that batching auditing can save a lot of time used in generating verification than separated auditing. With the increase of auditing tasks, this effect is more obvious. Thus, we can conclude that batch auditing is much more efficient than separated auditing.

8. Conclusion

In this paper, we proposed a new public auditing scheme for shared cloud data, which adds the user access authorization and the TPA auditing authorization, and supports group dynamic. With the user access authorization, only the user in the group can upload and access shared cloud data. When a user is revoked from a group, he/she cannot access the shared cloud data. The auditing authorization of TPA avoids the cloud responding to the unauthorized auditing challenges from malicious TPA. Moreover, our scheme utilizes a new random masking technique to preserve data privacy from TPA, and uses a common group privacy key to calculate the signatures on all data blocks to preserve users' identity privacy. To improve the efficiency of verifying multiple auditing tasks, we extend our scheme to support batch auditing. The experimental results show the high efficiency of our scheme.

Acknowledgements

This research is supported by National Natural Science Foundation of China (61572267, 61272425, 61402245), National Cryptography Development Fund of China (MMJJ201301011), Qingdao Construction Project of Science and Technology Development (JK2015-26), Minsheng Project of Huangdao District in Qingdao(2014-2-31), PAPD, and CICAEET.

Reference

[1] K. Ren, C. Wang and Q. Wang, "Security Challenges for the Public Cloud," *IEEE Internet Computing*, vol. 16, pp. 69-73, 2012. Article (CrossRef Link)

[2] D. Song, E. Shi, I. Fischer and U. Shankar, "Cloud Data Protection for the Masses," *IEEE Computer*, vol. 45, no. 1, pp. 39-45, 2012. Article (CrossRef Link)

[3] M. Arrington, "Gmail Disaster: Reports of Mass Email Deletions," Online at http://techcrunch.com/2006/12/28/gmail-disaster-reports-of-mass-email-deletions/, 2006. Article (CrossRef Link)

[4] Amazon S3 Team. Amazon S3 Availability Event: July 20, 2008. Online at http://status.aws.amazon.com/s3-20080720.html, 2008. Article (CrossRef Link)

[5] K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, vol. 15, no. 4, pp. 409-428, 2012. Article (CrossRef Link)

[6] 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 2007*, pp. 598-610, 2007. Article (CrossRef Link)

[7] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *Proc. of ASIACRYPT 2008*, Springer-Verlag, pp. 90-107, 2008. Article (CrossRef Link)

[8] C. Wang, Q. Wang, K. Ren and W. Lou, "Ensuring Data Storage Security in Cloud Computing," in *Proc. of ACM/ IEEE IWQoS. 2009*, pp. 1-9, 2009. Article (CrossRef Link)

[9] N. Cao, S. Yu, Z. Yang, W. Lou and Y. T. Hou, "LT Codes-based Secure and Reliable Cloud Storage Service," in *Proc. of IEEE INFOCOM* 2012, pp. 693-701, March 25-30, 2012. Article (CrossRef Link)

[10] C. Wang, S. Chow, Q. Wang, K. Ren and W. Lou, "Privacy Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362-375, 2013. Article (CrossRef Link)

[11] S. G. Worku, C. Xu, J. Zhao and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Computers and Electrical Engineering*, vol. 40, no. 5, pp. 1703-1713, 2014. Article (CrossRef Link)

[12] C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic Provable Data Possession," in *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS'09)*, pp. 213-222, 2009. Article (CrossRef Link)

[13] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu and S. S. Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds," in *Proc. of ACM SAC 2011*, pp. 1550-1557, 2011. Article (CrossRef Link)

[14] Q. Wang, C. Wang and K. Ren, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847-859, 2011. Article (CrossRef Link)

[15] Z. Mo，Y. Zhou and S. Chen, "A Dynamic Proof of Retrievability (PoR) Scheme with O(logn) Complexity," in *Proc. of Communication(ICC), 2012 IEEE Information Conference on IEEE*, pp. 912-916, June 10-15, 2012. Article (CrossRef Link)

[16] C. Liu, J. Chen and T. Yang, "Authorized Public Auditing of Dynamic Big Data Storage on Cloud with Efficient Verifiable Fine-grained Updates," *IEEE Transactions on Parallel and Distributed Systems*, vol.25, no.9, pp. 2234-2244, 2014. Article (CrossRef Link)

[17] Y. Zhang and M. Blanton, "Efficient Dynamic Provable Possession of Remote Data via Balanced Update Trees," in *Proc. of Department of the 8th ACM SIGSAC symposium on Information, computer and communications security. ACM*, pp.183-194, 2013. Article (CrossRef Link)

[18] J. Yu, K. Ren, C. Wang and V. Varadharajan, "Enabling Cloud Storage Auditing with Key-Exposure Resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp.1167-1179, 2015. Article (CrossRef Link)

[19] J. Yu, R. Hao, H. Zhao, M. Shu, and J. Fan, "IRIBE: Intrusion-Resilient Identity-Based Encryption," *Information Sciences*, Vol. 329, pp. 90-104, 2016. Article (CrossRef Link)

[20] J. Yu, K. Ren, and C. Wang, "Enabling Cloud Storage Auditing with Verifiable Outsourcing of Key Updates," *IEEE Transactions on Information Forensics and Security*, Vol. 11, No. 5, pp. 1362 – 1375, 2016. Article (CrossRef Link)

[21] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling Public Auditing for Shared Data in Cloud Storage Supporting Identity Privacy and Traceability," *Journal of Systems and Software*, Vol. 113, pp. 130-139, 2016. Article (CrossRef Link)

[22] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, Vol. 16, No. 2, pp. 317-323, 2015.

Article (CrossRef Link)
[23] H. Wang, Q. Wu, B. Qin and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *Information Security*, *IET*, vol.8, no.2, pp.114-121, 2014. Article (CrossRef Link)
[24] B. Wang, B. Li and H. Li, "Public Auditing for Shared Data with Efficient User Revocation in the Cloud," in *Proc. of IEEE INFOCOM 2013*, pp. 2904-2912, 2013. Article (CrossRef Link)
[25] B. Wang, H. Li and M. Li, "Privacy-Preserving Public Auditing for Shared Cloud Data Supporting Group Dynamics," in *Proc. of Communications (ICC), 2013 IEEE International Conference on IEEE*, pp. 1946-1950, June 9-13, 2013.Article (CrossRef Link)
[26] J. Yuan and S. Yu, "Efficient Public Integrity Checking for Cloud Data Sharing with Multi-user Modification," in *Proc. of IEEE INFOCOM 2014*, pp. 2121-2129, April 27-May 2, 2014. Article (CrossRef Link)
[27] B. Wang, B. Li and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," in *Proc. of IEEE Cloud 2012*, pp. 295-302, June 24-29, 2012. Article (CrossRef Link)

Wenting Shen is currently a master's degree candidate in college of Information Engineering, Qingdao University, China. Her research interests include cloud security and big data security

Jia Yu received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University, China, in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, China, in 2006. Since 2012, he has been a full professor and the department director of information security, at Qingdao University, China. He was a visiting professor with the Department of Computer Science and Engineering, the State University of New York at Buffalo from 2013 to 2014. His research interests include cloud computing security, key evolving cryptography, digital signature, and network security. He has published over 100 academic papers. He is the reviewer of more than 30 international academic journals.

Guangyang Yang is currently a master's degree candidate in college of Information Engineering, Qingdao University, China. His research interests include cloud security and big data security.

Yue Zhang is currently a master's degree candidate in college of Information Engineering, Qingdao University, China. Her research interests include cloud security and big data security.

Zhangjie Fu received his PhD in computer science from the College of Computer, Hunan University, China, in 2012. Currently, he works as an assistant professor from 2012 in Department of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include Cloud & Outsourcing Security, Digital Forensics, Network and Information Security. His research has been supported by NSFC, PAPD, and GYHY. Zhangjie is a member of IEEE, and a member of ACM.

Rong Hao received Master degree in Institute of Network Security from Shandong University. She is working in the College of Information Engineering, Qingdao University. Her research interest is information security.