Mona: Secure Multi-Owner Data Sharing for Dynamic Groups in the Cloud

Xuefeng Liu, Yuqing Zhang, *Member*, *IEEE*, Boyang Wang, and Jingbo Yan

Abstract—With the character of low maintenance, cloud computing provides an economical and efficient solution for sharing group resource among cloud users. Unfortunately, sharing data in a multi-owner manner while preserving data and identity privacy from an untrusted cloud is still a challenging issue, due to the frequent change of the membership. In this paper, we propose a secure multi-owner data sharing scheme, named Mona, for dynamic groups in the cloud. By leveraging group signature and dynamic broadcast encryption techniques, any cloud user can anonymously share data with others. Meanwhile, the storage overhead and encryption computation cost of our scheme are independent with the number of revoked users. In addition, we analyze the security of our scheme with rigorous proofs, and demonstrate the efficiency of our scheme in experiments.

Index Terms—Cloud computing, data sharing, privacy-preserving, access control, dynamic groups

1 Introduction

Cloud computing is recognized as an alternative to traditional information technology [1] due to its intrinsic resource-sharing and low-maintenance characteristics. In cloud computing, the cloud service providers (CSPs), such as Amazon, are able to deliver various services to cloud users with the help of powerful datacenters. By migrating the local data management systems into cloud servers, users can enjoy high-quality services and save significant investments on their local infrastructures.

One of the most fundamental services offered by cloud providers is data storage. Let us consider a practical data application. A company allows its staffs in the same group or department to store and share files in the cloud. By utilizing the cloud, the staffs can be completely released from the troublesome local data storage and maintenance. However, it also poses a significant risk to the confidentiality of those stored files. Specifically, the cloud servers managed by cloud providers are not fully trusted by users while the data files stored in the cloud may be sensitive and confidential, such as business plans. To preserve data privacy, a basic solution is to encrypt data files, and then upload the encrypted data into the cloud [2]. Unfortunately, designing an efficient and secure data sharing scheme for groups in the cloud is not an easy task due to the following challenging issues.

- *X. Liu, B. Wang, and J. Yan are with the National Key Laboratory of Integrated Services Networks, Xidian University, No. 2, Taibai Road, Xían city 710071, Shaanxi province, China. E-mail: {liuxf, bywang, yanjb}@mail.xidian.edu.cn, yanjb@nipc.org.cn.*
- *Y. Zhang is with the National Computer Network Intrusion Protection Center, Graduate University of Chinese Academy of Sciences, No. 19, Yuquan Road, Beijing 100049, China. E-mail: Zhangyq@gucas.ac.cn, zhangyq@ucas.ac.cn.*

Manuscript received 29 Feb. 2012; revised 1 Oct. 2012; accepted 22 Nov. 2012; published online 4 Dec. 2012.
Recommended for acceptance by V.B. Misic, R. Buyya, D. Milojicic, and Y. Cui.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2012-02-0167.
Digital Object Identifier no. 10.1109/TPDS.2012.331.

First, identity privacy is one of the most significant obstacles for the wide deployment of cloud computing. Without the guarantee of identity privacy, users may be unwilling to join in cloud computing systems because their real identities could be easily disclosed to cloud providers and attackers. On the other hand, unconditional identity privacy may incur the abuse of privacy. For example, a misbehaved staff can deceive others in the company by sharing false files without being traceable. Therefore, traceability, which enables the group manager (e.g., a company manager) to reveal the real identity of a user, is also highly desirable.

Second, it is highly recommended that any member in a group should be able to fully enjoy the data storing and sharing services provided by the cloud, which is defined as the *multiple-owner* manner. Compared with the *single-owner* manner [3], where only the group manager can store and modify data in the cloud, the multiple-owner manner is more flexible in practical applications. More concretely, each user in the group is able to not only read data, but also modify his/her part of data in the entire data file shared by the company.

Last but not least, groups are normally dynamic in practice, e.g., new staff participation and current employee revocation in a company. The changes of membership make secure data sharing extremely difficult. On one hand, the anonymous system challenges new granted users to learn the content of data files stored before their participation, because it is impossible for new granted users to contact with anonymous data owners, and obtain the corresponding decryption keys. On the other hand, an efficient membership revocation mechanism without updating the secret keys of the remaining users is also desired to minimize the complexity of key management.

Several security schemes for data sharing on untrusted servers have been proposed [4], [5], [6]. In these approaches, data owners store the encrypted data files in untrusted storage and distribute the corresponding decryption keys only to authorized users. Thus, unauthorized users as well as storage servers cannot learn the content of the data files because they have no knowledge of the decryption keys.

However, the complexities of user participation and revocation in these schemes are linearly increasing with the number of data owners and the number of revoked users, respectively. By setting a group with a single attribute, Lu et al. [7] proposed a secure provenance scheme based on the ciphertext-policy attribute-based encryption technique [8], which allows any member in a group to share data with others. However, the issue of user revocation is not addressed in their scheme. Yu et al. [3] presented a scalable and fine-grained data access control scheme in cloud computing based on the key policy attribute-based encryption (KP-ABE) technique [9]. Unfortunately, the single-owner manner hinders the adoption of their scheme into the case, where any user is granted to store and share data.

Our contributions. To solve the challenges presented above, we propose Mona, a secure multi-owner data sharing scheme for dynamic groups in the cloud. The main contributions of this paper include:

1. We propose a secure multi-owner data sharing scheme. It implies that any user in the group can securely share data with others by the untrusted cloud.
2. Our proposed scheme is able to support dynamic groups efficiently. Specifically, new granted users can directly decrypt data files uploaded before their participation without contacting with data owners. User revocation can be easily achieved through a novel revocation list without updating the secret keys of the remaining users. The size and computation overhead of encryption are constant and independent with the number of revoked users.
3. We provide secure and privacy-preserving access control to users, which guarantees any member in a group to anonymously utilize the cloud resource. Moreover, the real identities of data owners can be revealed by the group manager when disputes occur.
4. We provide rigorous security analysis, and perform extensive simulations to demonstrate the efficiency of our scheme in terms of storage and computation overhead.

The remainder of this paper is organized as follows: Section 2 overviews the related work. In Section 3, some preliminaries and cryptographic primitives are reviewed. In Section 4, we describe the system model and our design goals. In Section 5, the proposed scheme is presented in detail, followed by the security analysis and the performance analysis in Sections 6 and 7. Finally, we conclude the paper in Section 8.

2 Related Work

In [4], Kallahalla et al. proposed a cryptographic storage system that enables secure file sharing on untrusted servers, named Plutus. By dividing files into filegroups and encrypting each filegroup with a unique file-block key, the data owner can share the filegroups with others through delivering the corresponding lockbox key, where the lockbox key is used to encrypt the file-block keys. However, it brings about a heavy key distribution overhead for large-scale file sharing. Additionally, the file-block key needs to be updated and distributed again for a user revocation.

In [5], files stored on the untrusted server include two parts: file metadata and file data. The file metadata implies the access control information including a series of encrypted key blocks, each of which is encrypted under the public key of authorized users. Thus, the size of the file metadata is proportional to the number of authorized users. The user revocation in the scheme is an intractable issue especially for large-scale sharing, since the file metadata needs to be updated. In their extension version, the NNL construction [10] is used for efficient key revocation. However, when a new user joins the group, the private key of each user in an NNL system needs to be recomputed, which may limit the application for dynamic groups. Another concern is that the computation overhead of encryption linearly increases with the sharing scale.

Ateniese et al. [6] leveraged proxy reencryptions to secure distributed storage. Specifically, the data owner encrypts blocks of content with unique and symmetric content keys, which are further encrypted under a master public key. For access control, the server uses proxy cryptography to directly reencrypt the appropriate content key(s) from the master public key to a granted user's public key. Unfortunately, a collusion attack between the untrusted server and any revoked malicious user can be launched, which enables them to learn the decryption keys of all the encrypted blocks.

In [3], Yu et al. presented a scalable and fine-grained data access control scheme in cloud computing based on the KP-ABE technique. The data owner uses a random key to encrypt a file, where the random key is further encrypted with a set of attributes using KP-ABE. Then, the group manager assigns an access structure and the corresponding secret key to authorized users, such that a user can only decrypt a ciphertext if and only if the data file attributes satisfy the access structure. To achieve user revocation, the manager delegates tasks of data file reencryption and user secret key update to cloud servers. However, the single-owner manner may hinder the implementation of applications with the scenario, where any member in a group should be allowed to store and share data files with others.

Lu et al. [7] proposed a secure provenance scheme, which is built upon group signatures and ciphertext-policy attribute-based encryption techniques. Particularly, the system in their scheme is set with a single attribute. Each user obtains two keys after the registration: a group signature key and an attribute key. Thus, any user is able to encrypt a data file using attribute-based encryption and others in the group can decrypt the encrypted data using their attribute keys. Meanwhile, the user signs encrypted data with her group signature key for privacy preserving and traceability. However, user revocation is not supported in their scheme.

From the above analysis, we can observe that how to securely share data files in a multiple-owner manner for dynamic groups while preserving identity privacy from an untrusted cloud remains to be a challenging issue. In this paper, we propose a novel Mona protocol for secure data sharing in cloud computing. Compared with the existing works, Mona offers unique features as follows:

1. Any user in the group can store and share data files with others by the cloud.

2. The encryption complexity and size of ciphertexts are independent with the number of revoked users in the system.
3. User revocation can be achieved without updating the private keys of the remaining users.
4. A new user can directly decrypt the files stored in the cloud before his participation.

3 Preliminaries

3.1 Bilinear Maps

Let G_1 and G_2 be an additive cyclic group and a multiplicative cyclic group of the same prime order q, respectively [11]. Let $e : G_1 \times G_1 \rightarrow G_2$ denote a bilinear map constructed with the following properties:

1. Bilinear: For all $a, b \in Z_q^*$ and $P, Q \in G_1$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. Nondegenerate: There exists a point P such that $e(P, P) \neq 1$.
3. Computable: There is an efficient algorithm to compute $e(P, Q)$ for any $P, Q \in G_1$.

3.2 Complexity Assumptions

Definition 1 (q-strong Diffie-Hellman (q-SDH) Assumption [12]). *Given $(P_1, P_2, \gamma P_2, \gamma^2 P_2, \ldots, \gamma^q P_2)$, it is infeasible to compute $\frac{1}{\gamma+x} P_1$, where $x \in Z_q^*$.*

Definition 2 (Decision linear (DL) Assumption [12]). *Given $P_1, P_2, P_3, aP_1, bP_2, cP_3$, it is infeasible to decide whether $a + b = c \bmod q$.*

Definition 3 (Weak Bilinear Diffie-Hellman Exponent (WBDHE) Assumption [13]). *For unknown $a \in Z_q^*$, given $Y, aY, a^2Y, .., a^lY, P \in G_1$, it is infeasible to compute $e(Y, P)^{\frac{1}{a}}$.*

Definition 4 ((t,n)-general Diffie-Hellman Exponent (GDHE) Assumption [14]). *Let $f(X) = \Pi_{i=1}^{r}(X + x_i)$ and $g(X) = \Pi_{i=1}^{n-r}(X + x_i^{'})$ be the two random univariate polynomials. For unknown $k, \gamma \in Z_q^*$, given*

$$G_0, \gamma G_0, ..., \gamma^{t-1} G_0, \gamma f(\gamma) G_0, P_0, ..., \gamma^{t-1} P_0, kg(\gamma) H_0 \in G_1 \text{ and } e(G_0, H_0)^{f^2(\gamma) g(\gamma)} \in G_2,$$

it is infeasible to compute $e(G_0, H_0)^{kf(\gamma)g(\gamma)} \in G_2$.

3.3 Group Signature

The concept of group signatures was first introduced in [15] by Chaum and van Heyst. In general, a group signature scheme allows any member of the group to sign messages while keeping the identity secret from verifiers. Besides, the designated group manager can reveal the identity of the signature's originator when a dispute occurs, which is denoted as traceability. In this paper, a variant of the short group signature scheme [12] will be used to achieve anonymous access control, as it supports efficient membership revocation.

3.4 Dynamic Broadcast Encryption

Broadcast encryption [16] enables a broadcaster to transmit encrypted data to a set of users so that only a privileged subset of users can decrypt the data. Besides the above characteristics, dynamic broadcast encryption also allows the group manager to dynamically include new members while preserving previously computed information, i.e., user decryption keys need not be recomputed, the morphology and size of ciphertexts are unchanged and the group encryption key requires no modification. The first formal definition and construction of dynamic broadcast encryption are introduced based on the bilinear pairing technique in [14], which will be used as the basis for file sharing in dynamic groups.

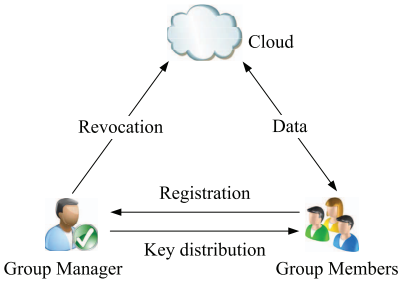


Fig. 1. System model.

4 System Model and Design Goals

4.1 System Model

We consider a cloud computing architecture by combining with an example that a company uses a cloud to enable its staffs in the same group or department to share files. The system model consists of three different entities: the cloud, a group manager (i.e., the company manager), and a large number of group members (i.e., the staffs) as illustrated in Fig. 1.

Cloud is operated by CSPs and provides priced abundant storage services. However, the cloud is not fully trusted by users since the CSPs are very likely to be outside of the cloud users' trusted domain. Similar to [3], [7], we assume that the cloud server is *honest but curious*. That is, the cloud server will not maliciously delete or modify user data due to the protection of data auditing schemes [17], [18], but will try to learn the content of the stored data and the identities of cloud users.

Group manager takes charge of system parameters generation, user registration, user revocation, and revealing the real identity of a dispute data owner. In the given example, the group manager is acted by the administrator of the company. Therefore, we assume that the group manager is fully trusted by the other parties.

Group members are a set of registered users that will store their private data into the cloud server and share them with others in the group. In our example, the staffs play the role of group members. Note that, the group membership is dynamically changed, due to the staff resignation and new employee participation in the company.

4.2 Design Goals

In this section, we describe the main design goals of the proposed scheme including access control, data confidentiality, anonymity and traceability, and efficiency as follows:

Access control: The requirement of access control is twofold. First, group members are able to use the cloud resource for data operations. Second, unauthorized users cannot access the cloud resource at any time, and revoked users will be incapable of using the cloud again once they are revoked.

Data confidentiality: Data confidentiality requires that unauthorized users including the cloud are incapable of

learning the content of the stored data. An important and challenging issue for data confidentiality is to maintain its availability for dynamic groups. Specifically, new users should decrypt the data stored in the cloud before their participation, and revoked users are unable to decrypt the data moved into the cloud after the revocation.

Anonymity and traceability: Anonymity guarantees that group members can access the cloud without revealing the real identity. Although anonymity represents an effective protection for user identity, it also poses a potential inside attack risk to the system. For example, an inside attacker may store and share a mendacious information to derive substantial benefit. Thus, to tackle the inside attack, the group manager should have the ability to reveal the real identities of data owners.

Efficiency: The efficiency is defined as follows: Any group member can store and share data files with others in the group by the cloud . User revocation can be achieved without involving the remaining users. That is, the remaining users do not need to update their private keys or reencryption operations. New granted users can learn all the content data files stored before his participation without contacting with the data owner.

5 The Proposed Scheme: Mona

5.1 Overview

To achieve secure data sharing for dynamic groups in the cloud, we expect to combine the group signature and dynamic broadcast encryption techniques. Specially, the group signature scheme enables users to anonymously use the cloud resources, and the dynamic broadcast encryption technique allows data owners to securely share their data files with others including new joining users.

Unfortunately, each user has to compute revocation parameters to protect the confidentiality from the revoked users in the dynamic broadcast encryption scheme, which results in that both the computation overhead of the encryption and the size of the ciphertext increase with the number of revoked users. Thus, the heavy overhead and large ciphertext size may hinder the adoption of the broadcast encryption scheme to capacity-limited users.

To tackle this challenging issue, we let the group manager compute the revocation parameters and make the result public available by migrating them into the cloud. Such a design can significantly reduce the computation overhead of users to encrypt files and the ciphertext size. Specially, the computation overhead of users for encryption operations and the ciphertext size are constant and independent of the revocation users.

5.2 Scheme Description

This section describes the details of Mona including system initialization, user registration, user revocation, file generation, file deletion, file access and traceability.

5.2.1 System Initialization

The group manager takes charge of system initialization as follows:

- Generating a bilinear map group system $S = (q, G_1, G_2, e(\cdot,\cdot))$.
- Selecting two random elements $H, H_0 \in G_1$ along with two random numbers $\xi_1, \xi_2 \in Z_q^*$, and computing $U = \xi_1^{-1} H$ and $V = \xi_2^{-1} H \in G_1$ such that $\xi_1 \cdot U = \xi_2 \cdot V = H$. In addition, the group manager computes $H_1 = \xi_1 H_0$ and $H_2 = \xi_2 H_0 \in G_1$.
- Randomly choosing two elements $P, G \in G_1$ and a number $\gamma \in Z_q^*$, and computing $W = \gamma \cdot P$, $Y = \gamma \cdot G$ and $Z = e(G, P)$, respectively.
- Publishing the system parameters including $(S, P, H, H_0, H_1, H_2, U, V, W, Y, Z, f, f_1, Enc())$, where f is a one-way hash function: $\{0,1\}^* \rightarrow Z_q^*$; f_1 is hash function: $\{0,1\}^* \rightarrow G_1$; and $Enc_k()$ is a secure symmetric encryption algorithm with secret key k.

In the end, the parameter $(\gamma, \xi_1, \xi_2, G)$ will be kept secret as the master key of the group manager.

TABLE 1
Revocation List

ID_{group}	A_1	x_1	t_1	P_1			
	A_2	x_2	t_2	P_2			
	.	.	.	.			
	A_r	x_r	t_r	P_r	Z_r	t_{RL}	$sig(RL)$

5.2.2 User Registration

For the registration of user i with identity ID_i, the group manager randomly selects a number $x_i \in Z_q^*$ and computes A_i, B_i as the following equation:

$$\begin{cases} A_i = \dfrac{1}{\gamma + x_i} \cdot P \in G_1 \\ B_i = \dfrac{x_i}{\gamma + x_i} \cdot G \in G_1. \end{cases} \tag{1}$$

Then, the group manager adds (A_i, x_i, ID_i) into the group user list, which will be used in the traceability phase. After the registration, user i obtains a private key (x_i, A_i, B_i), which will be used for group signature generation and file decryption.

5.2.3 User Revocation

User revocation is performed by the group manager via a public available revocation list (RL), based on which group members can encrypt their data files and ensure the confidentiality against the revoked users. As illustrated in Table 1, the revocation list is characterized by a series of time stamps $(t_1 < t_2 <, ..., t_r)$. Let ID_{group} denote the group identity. The tuple (A_i, x_i, t_i) represents that user i with the partial private key (A_i, x_i) is revoked at time t_i. $P_1, P_2, ..., P_r$ and Z_r are calculated by the group manager with the private secret γ as follows:

$$\begin{cases} P_1 = \dfrac{1}{\gamma + x_1} \cdot P \in G_1 \\ P_2 = \dfrac{1}{(\gamma + x_1)(\gamma + x_2)} \cdot P \in G_1 \\ \cdots \\ P_r = \dfrac{1}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)} \cdot P \in G_1 \\ Z_r = Z^{\frac{1}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)}} \in G_2. \end{cases} \tag{2}$$

Motivated by the verifiable reply mechanism in [19], to guarantee that users obtain the latest version of the

revocation list, we let the group manger update the revocation list each day even no user has being revoked in the day. In other words, the others can verify the freshness of the revocation list from the contained current date t_{RL}. In addition, the revocation list is bounded by a signature $sig(RL)$ to declare its validity. The signature is generated by the group manager with the BLS signature algorithm [20], i.e., $sig(RL) = \gamma f_1(RL)$. Finally, the group manager migrates the revocation list into the cloud for public usage.

5.2.4 File Generation

To store and share a data file in the cloud, a group member performs the following operations:

1. Getting the revocation list from the cloud. In this step, the member sends the group identity ID_{group} as a request to the cloud. Then, the cloud responds the revocation list RL to the member.
2. Verifying the validity of the received revocation list. First, checking whether the marked date is fresh. Second, verifying the contained signature $sig_(RL)$ by the equation $e(W, f_1(RL)) = e(P, sig(RL))$. If the revocation list is invalid, the data owner stops this scheme.
3. Encrypting the data file M. This encryption process can be divided into two cases according to the revocation list.
   a. **Case 1.** *There is no revoked user in the revocation list:*
      i. Selecting a unique data file identity ID_{data};
      ii. Choosing a random number $k \in Z_q^*$;
      iii. Computing the parameters C_1, C_2, K, C as the following equation:

$$\begin{cases} C_1 = k \cdot Y \in G_1 \\ C_2 = k \cdot P \in G_1 \\ K = Z^k \in G_2 \\ C = Enc_K(M). \end{cases} \quad (3)$$

   b. **Case 2.** *There are r revoked users in the revocation list.*
      i. Selecting a unique data file identity ID_{data};
      ii. Choosing a random number $k \in Z_q^*$;
      iii. Computing the parameters C_1, C_2, K, C as the following equation:

$$\begin{cases} C_1 = k \cdot Y \in G_1 \\ C_2 = k \cdot P_r \in G_1 \\ K = Z_r^k \in G_2 \\ C = Enc_K(M). \end{cases} \quad (4)$$

      In (4), Z_r and P_r are directly obtained from the revocation list.
4. Selecting a random number τ and computing $f(\tau)$. The hash value will be used for data file deletion operation. In addition, the data owner adds (ID_{data}, τ) into his local storage.
5. Constructing the uploaded data file as shown in Table 2, where t_{data} denotes the current time on the member, and σ is a group signature on $(ID_{data}, C_1, C_2, C, f(\tau), t_{data})$ computed by the data owner through Algorithm 1 with the private key (A, x).
6. Uploading the data shown in Table 2 into the cloud server and adding the ID_{data} into the local shared data list maintained by the manager. On receiving the data, the cloud first invokes Algorithm 2 to check its validity. If the algorithm returns true, the group signature is valid; otherwise, the cloud abandons the data. In addition, if several users have been revoked by the group manager, the cloud also performs revocation verification by using Algorithm 3. Finally, the data file will be stored in the cloud after successful group signature and revocation verifications.

TABLE 2
Message Format for Uploading Data

Group ID	Data ID	ciphertext	hash	Time	Signature
ID_{group}	ID_{data}	C_1, C_2, C	$f(\tau)$	t_{data}	σ

5.2.5 File Deletion

File stored in the cloud can be deleted by either the group manager or the data owner (i.e., the member who uploaded the file into the server). To delete a file ID_{data}, the group manager computes a signature $\gamma f_1(ID_{data})$ and sends the signature along with ID_{data} to the cloud. The cloud will delete the file if the equation $e(\gamma f_1(ID_{data}), P) = e(W, f_1(ID_{data}))$ holds.

Algorithm (1). Signature Generation
Input: Private key (A, x), system parameter (P, U, V, H, W) and data M.
Output: Generate a valid group signature on M.
begin
Select random numbers $\alpha, \beta, r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \in Z_q^*$
Set $\delta_1 = x\alpha$ and $\delta_2 = x\beta$
Computes the following values

$$\begin{cases} T_1 = \alpha \cdot U \\ T_2 = \beta \cdot V \\ T_3 = A_i + (\alpha + \beta) \cdot H \\ R_1 = r_\alpha \cdot U \\ R_2 = r_\beta \cdot V \\ R_3 = e(T_3, P)^{r_x} e(H, W)^{-r_\alpha - r_\beta} e(H, P)^{-r_{\delta_1} - r_{\delta_2}} \\ R_4 = r_x \cdot T_1 - r_{\delta_1} \cdot U \\ R_5 = r_x \cdot T_2 - r_{\delta_2} \cdot V \end{cases}.$$

Set $c = f(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5)$
Construct the following numbers

$$\begin{cases} s_\alpha = r_\alpha + c\alpha \\ s_\beta = r_\beta + c\beta \\ s_x = r_x + cx \\ s_{\delta_1} = r_{\delta_1} + c\delta_1 \\ s_{\delta_2} = r_{\delta_2} + c\delta_2 \end{cases}.$$

Return $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$
end

Algorithm (2). Signature Verification
Input: System parameter (P, U, V, H, W), M and a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$

Output: True or False.
begin
Compute the following values

$$\begin{cases} \tilde{R_1} = s_\alpha \cdot U - c \cdot T_1 \\ \tilde{R_2} = s_\beta \cdot V - c \cdot T_2 \\ \tilde{R_3} = (\frac{e(T_3, W)}{e(P, P)})^c e(T_3, P)^{s_x} e(H, W)^{-s_\alpha - s_\beta} \\ \qquad\qquad e(H, P)^{-s_{\delta_1} - s_{\delta_2}} \\ \tilde{R_4} = s_x \cdot T_1 - s_{\delta_1} \cdot U \\ \tilde{R_5} = s_x \cdot T_2 - s_{\delta_2} \cdot V \end{cases}.$$

if $c = f(M, T_1, T_2, T_3, \widetilde{R_1}, \widetilde{R_2}, \widetilde{R_3}, \widetilde{R_4}, \widetilde{R_5})$
Return True
else
Return False
end

Algorithm (3). Revocation Verification
Input: System parameter (H_0, H_1, H_2), a group signature σ, and a set of revocation keys $A_1, ..., A_r$
Output: Valid or Invalid.
begin
set $temp = e(T_1, H_1)e(T_2, H_2)$
for $i = 1$ to n
if $e(T_3 - A_i, H_0) = temp$
Return Valid
end if
end for
Return Invalid
end

In addition, Mona also allows data owners to delete their files stored in the cloud. Specially, the data owner does the following actions:

- Obtaining the tuple (ID_{data}, τ) from his local storage.
- Invoking Algorithm 1 to compute a group signature on (ID_{data}, τ).
- Sending (ID_{data}, τ) and the signature as a deletion request to the cloud.

Upon receiving the deletion request, the cloud calls Algorithms 2 and 3 to verify the group signature. After a successful group signature verification, the cloud will delete the data file if $f(\tau)$ equals to the hash value contained in the file.

5.2.6 File Access

To learn the content of a shared file, a member does the following actions:

1. Getting the data file and the revocation list from the cloud server. In this operation, the user first adopts its private key (A, x) to compute a signature σ_u on the message $(ID_{group}, ID_{data}, t)$ by using Algorithm 1, where t denote the current time, and the ID_{data} can be obtained from the local shared file list maintained by the manager. Then, the user sends a data request containing $(ID_{group}, ID_{data}, t, \sigma_u)$ to the cloud server. Upon receiving the request, the cloud server employs Algorithm 2 to check the validity of the signature and performs a revocation verification with Algorithm 3 if necessary according to the revocation list. After a successful verification, the cloud server responds the corresponding data file and the revocation list to the user.
2. Checking the validity of the revocation list. This operation is similar to the step 2 of file generation phase.
3. Verifying the validity of the file and decrypting it. The format of the downloaded file coincides with that given in Table 2. This operation can be divided into three cases according to the time stamp t_{data} and the revocation list. Suppose that there are r revoked users in the revocation list.

   a. *Case 1* ($t_{data} < t_1$). This case indicates that there is no revoked user before the data file is uploaded

      i. Invoking Algorithm 2 to check the group signature σ. If the algorithm returns false, the user stops this protocol.
      ii. Using his partial private key (A, B) to compute $\hat{K} = e(C_1, A)e(C_2, B)$.
      iii. Decrypting the ciphertext C with the computed key $\hat{K}$.

      Correctness:

$$\begin{aligned} \hat{K} &= e(C_1, A)e(C_2, B) \\ &= e\left(k \cdot Y, \frac{1}{\gamma + x} \cdot P\right) e\left(k \cdot P, \frac{x}{\gamma + x} \cdot G\right) \\ &= e(G, P)^{\frac{k\gamma}{\gamma + x}} e(P, G)^{\frac{kx}{\gamma + x}} \\ &= Z^k = K. \end{aligned}$$

   b. *Case 2* ($t_i < t_{data} < t_{i+1}$). This case indicates that i revoked users have been revoked before the data file is uploaded

      i. Verifying the group signature σ by using Algorithm 2.
      ii. Inputting $A_1, A_2, ...A_i$ to call Algorithm 3. If the algorithm returns invalid, the user terminates this operation.
      iii. Computing the value

$$A_{i,r} = \frac{1}{(\gamma + x)\prod_{\lambda=1}^{i}(\gamma + x_\lambda)} P$$

      by using Algorithm 4 with the input (A, x), $(P_1, x_1), ..., (P_i, x_i)$. The correctness of $A_{i,r}$ is due to the following relation:

$$\begin{aligned} &\frac{1}{x - x_i}\left(P_i - \frac{1}{(\gamma + x)\prod_{\lambda=1}^{i-1}(\gamma + x_\lambda)} P\right) \\ &= \frac{1}{x - x_i}\left(\frac{(\gamma + x) - (\gamma + x_i)}{(\gamma + x)(\gamma + x_i)(\prod_{\lambda=1}^{i-1}(\gamma + x_\lambda))}\right) P \\ &= \frac{1}{(\gamma + x)\prod_{\lambda=1}^{i}(\gamma + x_\lambda)} P. \end{aligned}$$

      iv. Calculating the decryption key $\hat{K} = e(C_1, A_{i,r})e(C_2, B)$.
      v. Decrypt the ciphertext C with the key $\hat{K}$.

Correctness

$$
\begin{aligned}
&e(C_1, A_{i,r})e(C_2, B)\\
&= e\left(kY, \frac{1}{(\gamma+x)\prod_{\lambda=1}^{i}(\gamma+x_\lambda)}P\right)\\
&\quad e\left(kP_i, \frac{x}{\gamma+x}\cdot G\right)\\
&= e(P,G)^{\frac{k\gamma}{(\gamma+x)\prod_{\lambda=1}^{i}(\gamma+x_\lambda)}} e(P,G)^{\frac{kx}{(\gamma+x)\prod_{\lambda=1}^{i}(\gamma+x_\lambda)}}\\
&= e(P,G)^{\frac{k\gamma+kx}{(\gamma+x)\prod_{\lambda=1}^{i}(\gamma+x_\lambda)}} = Z_i^k = K.
\end{aligned}
$$

c. *Case 3* ($t_r < t_{data}$). This case indicates that r revoked users have been revoked before the data file is uploaded

i. Verifying the group signature σ by using Algorithm 2.
ii. Inputting $A_1, A_2, ...A_r$ to call Algorithm 3. If the algorithm returns invalid, the user terminates this operation.
iii. Computing the value

$$A_{r,r} = \frac{1}{(\gamma+x)\prod_{\lambda=1}^{r}(\gamma+x_\lambda)}P$$

by using Algorithm 4 with the input (A, x), $(P_1, x_1), ..., (P_r, x_r)$.
iv. Calculating the decryption key $\hat{K} = e(C_1, A_{r,r})e(C_2, B)$.
v. Decrypting the ciphertext C with the key $\hat{K}$.

5.2.7 Traceability

When a data dispute occurs, the tracing operation is performed by the group manager to identify the real identity of the data owner. Given a signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, the group manager employs his private key (ξ_1, ξ_2) to compute $A_i = T_3 - (\xi_1 \cdot T_1 + \xi_2 \cdot T_2)$. Given the parameter A_i, the group manager can look up the user list to find the corresponding identity.

Algorithm 4. Parameters Computing
Input: The revoked user parameters $(P_1, x_1), ..., (P_r, x_r)$, and the private key (A, x).
Output: $A_{r,r}$ or NULL
begin
  set $temp = A$
  for $\lambda = 1$ to r
    if $x = x_\lambda$
      return NULL
    else
      set $temp = \frac{1}{x - x_\lambda}(P_\lambda - temp)$
  return $temp$
end

6 Security Analysis

In this section, we prove the security of Mona in terms of access control, data confidentiality, anonymity and traceability that are defined in Section 4.2.

Theorem 1. *Based on the group signature technique, the proposed scheme can achieve efficient access control.*

Proof. To access the cloud, a user needs to compute a group signature for his/her authentication. The employed group signature scheme can be regarded as a variant of the short group signature [12], which inherits the inherent unforgeability property, anonymous authentication, and tracking capability. The demonstration of Theorem 1 can be derived from the following three lemmas: □

Lemma 1.1. *Unrevoked users are able to access the cloud.*

Proof. The proof of Lemma 1.1 is equivalent to the correctness of Algorithm 2 (group signature verification). $\tilde{R_1} = R_1$ holds since $\tilde{R_1} = s_\alpha \cdot U - c \cdot T_1 = (r_\alpha + c\alpha)U - c \cdot \alpha \cdot U = R_1$. Analogously, we can directly obtain $\tilde{R_2} = R_2$, $\tilde{R_4} = R_4$, $\tilde{R_5} = R_5$. $\tilde{R_3} = R_3$ holds due to the following relations:

$$
\begin{aligned}
\tilde{R_3} &= \left(\frac{e(T_3, W)}{e(P,P)}\right)^c e(T_3, P)^{s_x} e(H, W)^{-s_\alpha - s_\beta} e(H, P)^{-s_{\delta_1} - s_{\delta_2}}\\
&= \left(\frac{e(T_3, W)}{e(P,P)}\right)^c e(T_3, P)^{r_x + cx_i} e(H, W)^{-r_\alpha - c\alpha - r_\beta - c\beta}\\
&\quad e(H, P)^{-r_{\delta_1} - cx_i\alpha - r_{\delta_2} - cx_i\beta}\\
&= \left(\frac{e(T_3, W)}{e(P,P)}\right)^c e(T_3, x_iP)^c e(-(\alpha+\beta)H, W + x_iP)^c\\
&\quad e(T_3, P)^{r_x} e(H, W)^{-r_\alpha - r_\beta} e(H, P)^{-r_{\delta_1} - r_{\delta_2}}\\
&= \left(\frac{e(T_3, W)}{e(P,P)}\right)^c e(T_3, x_iP)^c e(-(\alpha+\beta)H, W + x_iP)^c\\
R_3 &= \left(\frac{e(T_3, W)}{e(P,P)}\right)^c e(T_3 - (\alpha+\beta)H, W + x_iP)^c e(T_3, W)^{-c} R_3\\
&= \left(\frac{e(A_i, W + x_iP)}{e(P,P)}\right)^c R_3 = R_3.
\end{aligned}
$$

□

Lemma 1.2. *Revoked users cannot utilize the cloud after their revocation.*

Proof. Lemma 1.2 is equivalent to the correctness of Algorithm 3 (revocation verification). The correctness of revocation verification is based on the following relation:

$$
\begin{aligned}
e(T_3 - A_i, H_0) &= e(A_i + (\alpha+\beta)\cdot H - A_i, H_0)\\
&= e(\alpha H, H_0) e(\beta H, H_0)\\
&= e(\alpha U, \xi_1 H_0) e(\beta V, \xi_2 H_0)\\
&= e(T_1, H_1) e(T_2, H_2).
\end{aligned}
$$

□

Lemma 1.3. *An attacker is unable to access the cloud server based on the assumption of the intractability of q-SDH problem in G_1.*

Proof. The brief security analysis can be shown as follows: Suppose that an attacker $\mathcal{A}$ succeeds to forge a valid group signature with a nonnegligible probability in polynomial time. In addition, we assume f is a random oracle. According to the Forking Lemma [21], by using the

oracle replay technique, the attacker $\mathcal{A}$ obtains two valid signatures $(M, \sigma_0, c, \sigma_1)$ and $(M, \sigma_0, c^{'}, \sigma_1^{'})$ as follows:

$$\begin{cases} \sigma_0 = (T_1, T_2, T_3, c, R_1, R_2, R_3, R_4, R_5) \\ c = f(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \\ c^{'} = f^{'}(M, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5) \\ \sigma_1 = (s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}) \\ \sigma_1^{'} = (s_\alpha^{'}, s_\beta^{'}, s_x^{'}, s_{\delta_1}^{'}, s_{\delta_2}^{'}) \end{cases} \tag{5}$$

$$\begin{cases} s_\alpha = r_\alpha + c\alpha, s_\alpha^{'} = r_\alpha + c^{'}\alpha \\ s_\beta = r_\beta + c\beta, s_\beta^{'} = r_\beta + c^{'}\beta \\ s_x = r_x + cx, s_x^{'} = r_x + c^{'}x \\ s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_1}^{'} = r_{\delta_1} + c^{'}\delta_1 \\ s_{\delta_2} = r_{\delta_2} + c\delta_2, s_{\delta_2}^{'} = r_{\delta_2} + c^{'}\delta_2. \end{cases} \tag{6}$$

Then, $\mathcal{A}$ can compute an SDH tuple $(\hat{x} = \Delta s_x / \Delta c, \hat{A} = T_3 - ((\Delta s_\alpha + \Delta s_\beta)/\Delta c) \cdot H)$ such that $\hat{A} = \frac{1}{\gamma + \hat{x}}$ and

$$e(\hat{A}, W + \hat{x}P) = e(P, P),$$

where $\Delta s_x = s_x - s_x^{'}$, $\Delta c = c - c^{'}$, $\Delta s_\alpha = s_\alpha - s_\alpha^{'}$, and $\Delta s_\beta = s_\beta - s_\beta^{'}$. Obviously, this contradicts with q-SDH assumption. □

Theorem 2. *The proposed scheme supports privacy preserving and traceability.*

Proof. The demonstration of this theorem is twofold. On one hand, the group manager has the ability to identify the real signer. Given a valid group signature $\sigma = (T_1, T_2, T_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ and the private tuple (ξ_1, ξ_2), the group manger can compute the private key of the signer through the equation $A_i = T_3 - (\xi_1 \cdot T_1 + \xi_2 \cdot T_2)$. The correctness of the equation holds based on the following relation:

$$\begin{aligned} &T_3 - (\xi_1 \cdot T_1 + \xi_2 \cdot T_2) = A_i + (\alpha + \beta) \cdot \\ &H - (\xi_1 \alpha \cdot U + \xi_2 \beta \cdot V) = A_i. \end{aligned}$$

On the other hand, other entities cannot reveal the signer's identity from a group signature. Otherwise, DL assumption will be in contradiction. Further proofs about the correctness, unforgeability, anonymity and traceability of group signatures can be found in [12]. □

Theorem 3. *The proposed scheme protects data confidentiality under the hardness of the WBDHE problem and GDHE problem.*

Proof. Theorem 3 can be deduced from the following two lemmas:

Lemma 3.1. *The cloud server is unable to learn the content of the stored files.*

Proof. To prove this lemma, we take a data file (C_1, C_2, C) as an example to demonstrate the data confidentiality, where $C_1 = k \cdot Y, C_2 = k \cdot P, K = Z^k, C = Enc_K(M)$ and no user has been revoked before the data file is uploaded. Suppose that the cloud server can compute $K = Z^k$, i.e., "given $C_1 = k \cdot Y, C_2 = k \cdot P, P$, for unknown γ, computing $e(C_1, P)^{\frac{1}{\gamma}} = e(G, P)^k = K$." This contradicts with the WBDHE assumption. On the other hand, given the revocation list, the cloud server learns the partial private key of a revoked user i, i.e., (A_i, x_i) for a revoked user. Without the knowledge of the other part private key B_i, it is also unable to compute the decryption key through the equation $e(C_1, A_i)e(C_2, B_i) = Z^k$. Thus, the correctness of Lemma 3.1 can be ensured. □

Lemma 3.2. *Even under the collusion with revoked users, the cloud server is also incapable of learning the content of the files stored after their revocation.*

Proof. We first define two polynomial functions $f(X) = \Pi_{i=1}^{r}(X + x_i)$ and $g(X) = \Pi_{i=1}^{n-r}(X + x_i^{'})$. Let G_0 and P_0 denote two elements in group G_1. Then, we set $G = f(\gamma)G_0$ and $P = f(\gamma)g(\gamma)P_0$. To maintain the confidentiality against the revoked users, the data owner computes the header information C_1, C_2 and the encryption key K as follows:

$$\begin{cases} C_1 = kY = k\gamma f(\gamma) \cdot G_0 \\ C_2 = kP_r = \dfrac{k}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)} P = kg(\gamma)P_0 \\ K = Z_r^k = Z^{\frac{k}{(\gamma + x_1)(\gamma + x_2) \cdots (\gamma + x_r)}} = Z^{\frac{k}{f(\gamma)}} \\ = e(G, P)^{f(\gamma)} = e(G_0, H_0)^{kf(\gamma)g(\gamma)}. \end{cases} \tag{7}$$

We can observe that it is impossible for revoked users to compute the encryption key K, since "given $k\gamma f(\gamma) \cdot G_0$ and $kg(\gamma)P_0$, computing $e(G_0, H_0)^{kf(\gamma)g(\gamma)}$" is an instance of (t,n)-GDHE problem, which has been demonstrated to be intractable in polynomial time [14]. □

By the analysis above, we conclude that the proposed scheme achieves the security goals including access control, data confidentiality as well as anonymity and traceability.

7 Performance Evaluation

In this section, we first analyze the storage cost of Mona, and then perform experiments to test its computation cost.

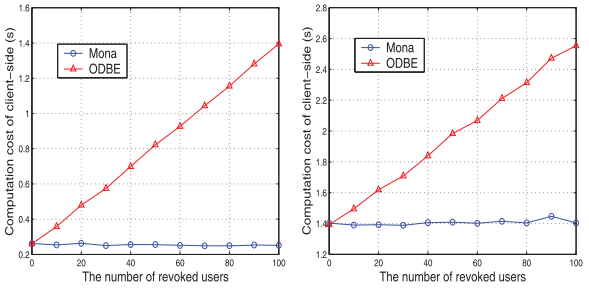
7.1 Storage

Without loss of generality, we set $q = 160$ and the elements in G_1 and G_2 to be 161 and 1,024 bit, respectively. In addition, we assume the size of the data identity is 16 bits, which yield a group capacity of 2^{16} data files. Similarly, the size of user and group identity are also set as 16 bits.

Group manager. In Mona, the master private key of the group manager is $(G, \gamma, \xi_1, \xi_2) \in G_1 \times {Z_q}^3$. Additionally, the user list and the shared data list should be stored at the group manager. Considering an actual system with 200 users and assuming that each user share 50 files in average, the total storage of the group manager is $(80.125 + 42.125 * 200 + 2 * 10,000) * 10^{-3} \approx 28.5$ Kbytes, which is very acceptable.

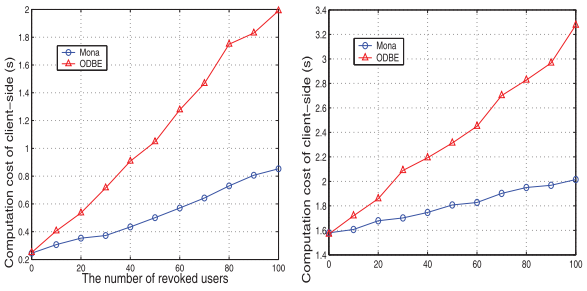
Group members. Essentially, each user in our scheme only needs to store its private key $(A_i, B_i, x_i) \in {G_1}^2 \times Z_q$, which is about 60 bytes. It is worth noting that there is a tradeoff between the storage and the computation overhead. For example, the four pairing operations including $(e(H, W), e(H, P), e(P, P), e(A_i, P)) \in {G_2}^4$ can be precomputed once and stored for the group signature generation and verification. Therefore, the total storage of each users is about 572 bytes.

The extra storage overhead in the cloud. In Mona, the format of files stored in the cloud is shown in Table 2. Since C_3 is the ciphertext of the file under the symmetrical encryption, the extra storage overhead to store the file is about 248 bytes, which includes $(ID_{group}, ID_{data}, C_1, C_2, C_3, f(\tau), t_{data}, \sigma)$.



(a) Generating a 10 MB file (b) Generating a 100 MB file

Fig. 2. Comparison on computation cost for file generation between Mona and ODBE [14].



(a) Accessing a 10 MB file (b) Accessing a 100 MB file

Fig. 3. Comparison on computation cost for file access between Mona and ODBE [14].

7.2 Simulation

To study the performance, we have simulated Mona by using C programming language with GMP Library [22], Miracl Library [23], and PBC Library [24]. The simulation consists of three components: client side, manager side as well as cloud side. Both client-side and manager-side processes are conducted on a laptop with Core 2 T7250 2.0 GHz, DDR2 800 2G, Ubuntu 10.04 X86. The cloud-side process is implemented on a machine that equipped with Core 2 i3-2350 2.3 GHz, DDR3 1066 2G,Ubuntu 12.04 X64. In the simulation, we choose an elliptic curve with 160-bit group order, which provides a competitive security level with 1,024-bit RSA.

7.2.1 Client Computation Cost

In Fig. 2, we list the comparison on computation cost of clients for data generation operations between Mona and the way that directly using the original dynamic broadcast encryption (ODBE) [14]. It is easily observed that the computation cost in Mona is irrelevant to the number of revoked users. On the contrary, the computation cost increases with the number of revoked users in ODBE. The reason is that the parameters (P_r, Z_r) can be obtained from the revocation list without sacrificing the security in Mona, while several time-consuming operations including point multiplications in G_1 and exponentiations in G_2 have to be performed by clients to compute the parameters in ODBE. From Figs. 2a and 2b, we can find out that sharing a 10-Mbyte file and a 100-Mbyte one, cost a client about 0.2 and 1.4 seconds in our scheme, respectively, which implies that the symmetrical encryption operation domains the computation cost when the file is large.

The computation cost of clients for file access operation with the size of 10 and 100 Mbytes are illustrated in Fig. 3. The computation cost in Mona increases with the number of revoked users, as clients require to perform Algorithms 3 and 4 to compute the parameter $A_{r,r}$ and check whether the data owner is a revoked user. Besides the above operations, $P_1, P_2, ..., P_r$ need to be computed by clients in ODBE. Therefore, Mona is still superior than ODBE in terms of computation cost. Similar to the data generation operation, the total computation cost is mainly determined by the symmetrical decryption operation if the accessed file is large, which can be verified from Figs. 3a and 3b. In addition, the file deletion for clients is about 0.075 seconds, because it only costs a group signature on a message (ID_{data}, τ), where τ is a 160-bit number in Z_q^*.

7.2.2 Cloud Computation Cost

To evaluate the performance of the cloud in Mona, we test its computation cost to respond various client operation requests including file generation, file access, and file deletion. Assuming the sizes of requested files are 100 and 10 MB, the test results are given in Table 3. It can be seen that the computation cost of the cloud is deemed acceptable, even when the number of revoked users is large. This is because the cloud only involves group signature and revocation verifications to ensure the validity of the requestor for all operations. In addition, it is worth noting that the computation cost is independent with the size of the requested file for access and deletion operations, since the size of signed message is constant, e.g., $(ID_{group}, ID_{data}, t)$ in file access and (ID_{data}, τ) in file deletion requests as described in Section 5.

8 Conclusion

In this paper, we design a secure data sharing scheme, Mona, for dynamic groups in an untrusted cloud. In Mona, a user is able to share data with others in the group without revealing identity privacy to the cloud. Additionally, Mona supports efficient user revocation and new user joining. More specially, efficient user revocation can be achieved through a public revocation list without updating the private keys of the remaining users, and new users can directly decrypt files stored in the cloud before their participation. Moreover, the storage overhead and the encryption computation cost are constant. Extensive analyses show that our proposed scheme satisfies the desired security requirements and guarantees efficiency as well.

TABLE 3
Computation Cost of the Cloud (s)

Request	The number of revoked users		
	0	50	100
File generation (100 MB)	0.065	0.154	0.271
File generation (10 MB)	0.045	0.125	0.226
File access (100 MB)	0.045	0.150	0.237
File access (10 MB)	0.045	0.151	0.240
File deletion (100 MB)	0.041	0.153	0.240
File deletion (10 MB)	0.042	0.156	0.238

ACKNOWLEDGMENTS

The authors thank the editors and anonymous reviewers for their valuable comments to significantly improve the quality of this paper. This work was supported in part by the National Science Foundation of China under Grant Nos. 60970140, 61272481, and 61272522.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Comm. ACM,* vol. 53, no. 4, pp. 50-58, Apr. 2010.

[2] S. Kamara and K. Lauter, "Cryptographic Cloud Storage," *Proc. Int'l Conf. Financial Cryptography and Data Security (FC),* pp. 136-149, Jan. 2010.

[3] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving Secure, Scalable, and Fine-Grained Data Access Control in Cloud Computing," *Proc. IEEE INFOCOM,* pp. 534-542, 2010.

[4] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable Secure File Sharing on Untrusted Storage," *Proc. USENIX Conf. File and Storage Technologies,* pp. 29-42, 2003.

[5] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing Remote Untrusted Storage," *Proc. Network and Distributed Systems Security Symp. (NDSS),* pp. 131-145, 2003.

[6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," *Proc. Network and Distributed Systems Security Symp. (NDSS),* pp. 29-43, 2005.

[7] R. Lu, X. Lin, X. Liang, and X. Shen, "Secure Provenance: The Essential of Bread and Butter of Data Forensics in Cloud Computing," *Proc. ACM Symp. Information, Computer and Comm. Security,* pp. 282-292, 2010.

[8] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization," *Proc. Int'l Conf. Practice and Theory in Public Key Cryptography Conf. Public Key Cryptography,* http://eprint.iacr.org/2008/290.pdf, 2008.

[9] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data," *Proc. ACM Conf. Computer and Comm. Security (CCS),* pp. 89-98, 2006.

[10] D. Naor, M. Naor, and J.B. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Proc. Ann. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO),* pp. 41-62, 2001.

[11] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO),* pp. 213-229, 2001.

[12] D. Boneh, X. Boyen, and H. Shacham, "Short Group Signature," *Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO),* pp. 41-55, 2004.

[13] D. Boneh, X. Boyen, and E. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," *Proc. Ann. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT),* pp. 440-456, 2005.

[14] C. Delerablee, P. Paillier, and D. Pointcheval, "Fully Collusion Secure Dynamic Broadcast Encryption with Constant-Size Ciphertexts or Decryption Keys," *Proc. First Int'l Conf. Pairing-Based Cryptography,* pp. 39-59, 2007.

[15] D. Chaum and E. van Heyst, "Group Signatures," *Proc. Int'l Conf. Theory and Applications of Cryptographic Techniques (EUROCRYPT),* pp. 257-265, 1991.

[16] A. Fiat and M. Naor, "Broadcast Encryption," *Proc. Int'l Cryptology Conf. Advances in Cryptology (CRYPTO),* pp. 480-491, 1993.

[17] B. Wang, B. Li, and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," *Proc. 10th Int'l Conf. Applied Cryptography and Network Security,* pp. 507-525, 2012.

[18] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing," *Proc. IEEE INFOCOM,* pp. 525-533, 2010.

[19] B. Sheng and Q. Li, "Verifiable Privacy-Preserving Range Query in Two-Tiered Sensor Networks," *Proc. IEEE INFOCOM,* pp. 46-50, 2008.

[20] D. Boneh, B. Lynn, and H. Shacham, "Short Signature from the Weil Pairing," *Proc. Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology,* pp. 514-532, 2001.

[21] D. Pointcheval and J. Stern, "Security Arguments for Digital Signatures and Blind Signatures," *J. Cryptology,* vol. 13, no. 3, pp. 361-396, 2000.

[22] The GNU Multiple Precision Arithmetic Library (GMP), http://gmplib.org/, 2013.

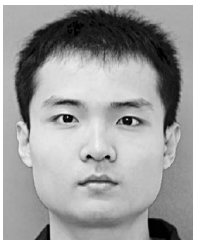
[23] Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL), http://certivox.com/, 2013.

[24] The Pairing-Based Cryptography Library (PBC), http://crypto.stanford.edu/pbc/howto.html, 2013.

Xuefeng Liu received the BSc degree in information security from Xidian University, China, 2007. He joined Xidian University in 2007 for the MSc and PhD degrees. His research interests include wireless network security, cloud computing, mobile computing and applied cryptography.

Yuqing Zhang received the BSc and MSc degrees in computer science from Xidian University, China, in 1987 and 1990, respectively. He received the PhD degree in cryptography from Xidian University in 2000. He is a professor and supervisor of PhD students at the Graduate University of Chinese Academy of Sciences. His research interests include cryptography, wireless security and trust management. He is a member of the IEEE.

Boyang Wang received the BS degree in information security from Xidian University in 2007. He is currently working toward the PhD degree from Xidian University, Xi'an, China. His research interests focus on security and privacy issues in cloud computing, social network and network coding.



Jingbo Yan received the BSc degree in information security from Xidian University, China, 2009. He joined Xidian University in 2009 for the MSc and PhD degrees. His research interests include privacy, applied cryptography and computer security.