Anonymous and Traceable Group Data Sharing in Cloud Computing

Jian Shen, *Member, IEEE*, Tianqi Zhou, Xiaofeng Chen, *Senior Member, IEEE*, Jin Li, and Willy Susilo, *Senior Member, IEEE*

***Abstract*— Group data sharing in cloud environments has become a hot topic in recent decades. With the popularity of cloud computing, how to achieve secure and efficient data sharing in cloud environments is an urgent problem to be solved. In addition, how to achieve both anonymity and traceability is also a challenge in the cloud for data sharing. This paper focuses on enabling data sharing and storage for the same group in the cloud with high security and efficiency in an anonymous manner. By leveraging the key agreement and the group signature, a novel traceable group data sharing scheme is proposed to support anonymous multiple users in public clouds. On the one hand, group members can communicate anonymously with respect to the group signature, and the real identities of members can be traced if necessary. On the other hand, a common conference key is derived based on the key agreement to enable group members to share and store their data securely. Note that a symmetric balanced incomplete block design is utilized for key generation, which substantially reduces the burden on members to derive a common conference key. Both theoretical and experimental analyses demonstrate that the proposed scheme is secure and efficient for group data sharing in cloud computing.**

***Index Terms*— Group data sharing, anonymous, traceability, key agreement, symmetric balanced incomplete block design (SBIBD).**

Manuscript received May 27, 2017; revised September 27, 2017; accepted November 4, 2017. Date of publication November 16, 2017; date of current version January 3, 2018. The CICAEET was supported by Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Nanjing University of Information Science & Technology, Nanjing, China, and the PAPD was supported by the Network Monitoring Center of Jiangsu Province. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Sherman S.-M. Chow. *(Corresponding author: Willy Susilo.)*

J. Shen and T. Zhou are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: s_shenjian@126.com; tq_zhou@126.com).

X. Chen is with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China (e-mail: xfchen@xidian.edu.cn).

J. Li is with the School of Computer Science, Guangzhou University, Guangzhou 510006, China (e-mail: jinli71@gmail.com).

W. Susilo is with the School of Computing and Information Technology, Institute of Cybersecurity and Cryptology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org., provided by the author. The material includes appendix of the paper. Contact s_shenjian@126.com for further questions about this work.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2774439

I. Introduction

COMPARED with the traditional information sharing and communication technology, cloud computing has attracted the interests of most researchers because of its low energy consumption and resource sharing characteristics. Cloud computing can not only provide users with apparently limitless computing resources but also provide users with apparently limitless storage resources [1]–[3]. Cloud storage is one of the most important services in cloud computing, which enables the interconnection of all types of electronic products. Moreover, various forms of data information can freely flow with respect to the cloud storage service, for instance, social networks, video editing and home networks. However, little attention has been given to group data sharing in the cloud, which refers to the situation in which multiple users want to achieve information sharing in a group manner for cooperative purposes [4], [5]. Group data sharing has many practical applications, such as electronic health networks [6], wireless body area networks [7], and electronic literature in libraries. There are two ways to share data in cloud storage. The first is a one-to-many pattern, which refers to the scenario where one client authorizes access to his/her data for many clients [8]. The second is a many-to-many pattern, which refers to a situation in which many clients in the same group authorize access to their data for many clients at the same time.

Consider the following real-life scenario: in a research group at a scientific research institution, each member wants to share their results and discoveries with their team members. In this case, members on the same team are able to access all of the team's results (e.g., innovative ideas, research results, and experimental data). However, the maintenance and challenges caused by the local storage increase the difficulty and workload of information sharing in the group. Outsourcing data or time-consuming computational workloads to the cloud solves the problems of maintenance and challenges caused by local storage and reduces the redundancy of data information, which reduces the burden on enterprises, academic institutions or even individuals. However, due to the unreliability of the cloud, the outsourced data are prone to be leaked and tampered with. In many cases, users have only relatively low control in the cloud service and cannot guarantee the security of the stored data. In addition, in some cases, the user would prefer to anonymously achieve data sharing in the cloud.

Our goal is to achieve anonymous data sharing under a cloud computing environment in a group manner with high security and efficiency. To achieve this goal, the following challenging problems should be taken into consideration.

Firstly, an arbitrary and variable number of group members should be supported. In practical applications, the number of members in each group is arbitrary, and the dynamic joining and exiting of group members is frequent [9]. A desired scheme not only supports the participation of any number of users but also supports efficient key and data updating. Secondly, the confidentiality of the outsourced data should be preserved. Since the uploaded data may be sensitive and confidential business plans or scientific research achievements, data leakages may cause significant losses or serious consequences [10]. Without the guarantee of confidentiality, users would not like to be involved in the cloud to share data. Thirdly, the way that data are shared should follow the many-to-many pattern, which makes the information sharing more convenient and efficient. Rather than the single-owner manner in which data storage and deletion can only be completed by the group manager, we need a multiple-owner manner [11], where users have greater authority over their stored data. Specifically, any user in the group can freely store and read their data stored in the cloud, and the deletion of data is performed by the user. Finally, in the many-to-many group data sharing pattern, it is essential to provide authentication services to resist misbehaving users. For instance, a misbehaving user may deliberately upload faulty data or misleading data to disturb and influence the cloud storage system. In addition, to resist the different key attack [12], a fault-tolerant property should be supported in the scheme.

A. Main Contributions

To address the above challenges, we present a novel traceable group data sharing scheme for cloud computing with traceability and anonymity. The main contributions of this paper include the following.

1) Arbitrary Number of Users and Dynamic Changes Are Supported: To address an arbitrary number of users in real applications, we introduce the concept of the volunteer, which is used to satisfy the specific structure of the SBIBD such that the number of users can be arbitrary rather than restricted by the value of a prime k. Moreover, in real cloud storage applications, users may join or leave freely. The proposed scheme can efficiently support dynamic changes of users with respect to the access control and the many-to-many data sharing pattern.

2) The Confidentiality of the Outsourced Data Is Preserved: In our scheme, the outsourced data are encrypted with a common conference key prior to being uploaded. Attacks or the cloud having no access to the common conference key cannot reveal any information of the data stored in the cloud. The security of the encryption key is based on elliptic curve cryptography (ECC) and the bilinear Diffie-Hellman (BDH) assumption. Consequently, users can safely exchange data with others under the semi-trusted cloud.

3) Traceability Under an Anonymous Environment Is Achieved by Our Scheme: With respect to the key agreement, every user in the cloud is able to freely share data with other users. Moreover, users can exchange information in the cloud anonymously with respect to the group signature. Note that the group manager can reveal the real identity of the data owner based on the group signature bound with the data when a dispute occurs.

4) Authentication Services and Fault-Tolerant Property Are Provided: During the key agreement, each member exchange messages along with the group signature to declare that their identity is valid. Furthermore, the uploaded data file will be bound with the group signature such that the cloud can verify the validity of the file. In addition, the fault-tolerant property is supported in our scheme, which guarantees that malicious users can be identified and removed such that a unique common conference key can be derived.

The above main contributions address challenges for group data sharing in cloud computing elegantly. Therefore, the proposed scheme is suitable for data sharing in a group manner under the cloud environment. Meanwhile, it can prompt the further development and employment of key agreement for data sharing using the SBIBD technique.

B. Related Work

Ateniese *et al.* [13] proposed a proxy re-encryption scheme to manage distributed file systems that attempt to achieve secure data storage in the semi-trusted party. Based on bilinear maps, the scheme offers improved security guarantees. Although the scheme provides a stronger concept of security compared with [14], it is still vulnerable under collusion attacks and revoked malicious users.

In order to overcome the above vulnerabilities, an effective access control for cloud computing was proposed by Yu *et al.* [11], which attempts to protect the outsourced data from attackers and revoked malicious users. With respect to the key policy attribute-based encryption (KA-ABE) technique, it provides effective access control with fine grainedness, scalability and data confidentiality simultaneously. Specifically, each data file is encrypted with a random key chosen by the user. Subsequently, the random key will be encrypted by the KA-ABE. An access structure and secret key maintained by the group manager are distributed to authorized users, which can be used to decrypt the outsourced data. Note that if and only if the attribute of the data satisfies the access structure can the outsourced data be decrypted. However, the scheme is designed only for a general one-to-many communication system, which makes it inapplicable for the many-to-many pattern.

On the other hand, a number of studies have been proposed to protect users' privacy [15]. In [16], a traceable privacy-preserving communication scheme was proposed for vehicle-to-grid networks in smart grids. However, this scheme is only suitable for two entities (i.e., vehicles and the central aggregator or the local aggregator); thus, it cannot be applied in cloud environments for the purpose of group data sharing. An example of group data sharing in cloud computing was proposed by Liu *et al.* [17]. In [17], a secure scheme was proposed to support anonymous data sharing in cloud computing. Both anonymity and traceability are well supported by employing the group signature technique. In addition, efficient user changes are achieved by taking advantage of the dynamic broadcast encryption. However, this scheme suffers from the collusion attack performed by the cloud server and the revoked malicious user. In addition, compared with the broadcast encryption, we believe that the decentralized model is more suitable for data sharing in the cloud. Specifically, in [18], the key management system falls into two categories. The first is key distribution, in which the generation and distribution of the key is completely accomplished by a centralized controller. The second is key agreement, where all the members in the group fairly contribute, negotiate and determine a common conference key together. In the cloud environment, key distribution may be vulnerable since the centralized controller is the bottleneck of the system. Moreover, the large amount of computation and distribution for a common conference key may cause a large burden for the centralized controller.

Many researchers have devoted themselves to the design of data sharing schemes in the cloud. But the problems existing in the above research still need to be resolved. In this paper, we focus on constructing an efficient and secure data sharing scheme that can support anonymous and traceable group data sharing in cloud computing. Note that the collusion attack is considered and addressed. Moreover, many-to-many group data sharing is supported in the proposed scheme.

C. Organization

The remainder of this paper is organized as follows. Section 2 presents some preliminaries in cryptographic and combinatorial mathematics. Section 3 describes the system model and our design goals. Section 4 presents the proposed scheme in detail. Section 5 and Section 6 perform the security and performance analyses, respectively. Section 7 concludes this paper and our work.

II. Preliminaries and Security Assumptions

A. Cryptographic Bilinear Maps

Let G_1 be an additive cyclic group of order q and G_2 be a cyclic multiplicative group of order q, where q is a large prime number. G_1 is the group of points of an elliptic curve over F_p, and G_2 is a subgroup of $F^*_{p^2}$. Weil pairing is an example of the bilinear map, which has the following properties [19].

1) Bilinear: For any $\mathcal{P}, \mathcal{Q} \in G_1$ and $a, b \in Z$, we have $\hat{e}(a\mathcal{P}, b\mathcal{Q}) = \hat{e}(\mathcal{P}, \mathcal{Q})^{ab}$.

2) Non-degenerate: If $\mathcal{P}$ is a generator of G_1, then $\hat{e}(\mathcal{P}, \mathcal{P}) \in F^*_{p^2}$ is a generator of G_2. In other words, $\hat{e}(\mathcal{P}, \mathcal{P}) \neq 1$.

3) Computable: Given $\mathcal{P}, \mathcal{Q} \in G_1$, an efficient algorithm exists to compute $\hat{e}(\mathcal{P}, \mathcal{Q})$.

B. Group Signature

The group signature scheme is a technique that allows a group member to anonymously sign messages in the name of the group. The concept of group signatures [20] was first introduced by David Chaum and Eugene van Heyst in 1991. A group manager is an important part of the group signature scheme, who is responsible for managing the entry and exit of the group members and for revealing the true identity of the group members when necessary. A well-designed group signature scheme should meet the requirements of unforgeable, anonymity and traceability. A variant of the short group signature scheme [21] will be utilized in this paper to achieve anonymous data sharing with efficient access control and traceability.

C. Block Design and $(v, k+1, 1)$-Design

In combinatorial mathematics, a block design is a set together with a family of subsets whose members are chosen to satisfy some set of properties that are deemed to be useful for a particular application [22]. Definition 1 below defines the balanced incomplete block design (BIBD) in detail.

Definition 1: Let $V = \{0, 1, 2 \ldots v-1\}$ be a set of v elements and $B = \{B_0, B_1, B_2 \ldots B_{b-1}\}$ be a set of b blocks, where B_i is a subset of V and $|B_i| = k$. For a finite incidence structure $\sigma = \{V, B\}$, if σ satisfies the following conditions, then it is a BIBD, which is called a (b, v, r, k, λ)-design.

1. Each element of V appears in exactly r of the b blocks.

2. Every two elements of V appear simultaneously in exactly λ of the b blocks.

3. Parameters k and v of V satisfy the condition of $k < v$ such that no block contains all the elements of the set V.

4. Parameters b and v of V satisfy the condition of $b \geq v$. The case of equality is called a symmetric design.

Here, v is the number of elements of V, b denotes the number of blocks, and k indicates the number of elements in each block, while r and λ are the parameters of the design. For a (b, v, r, k, λ)-design, if the condition of $k = r$ and $b = v$ holds, then it is a symmetric balanced incomplete block design (SBIBD). It is also called a (v, k, λ)-design. In this paper, we require a $(v, k+1, 1)$-design to construct our multicast decentralized model, where k is a prime number and $\lambda = 1$. The reason for us to choose the $(v, k+1, 1)$-design will be shown in detail in Section 4. Moreover, in the BIBD and the SBIBD, these five parameters are not all independent: b and r are determined by v, k and λ. Two basic equations connecting these parameters in the BIBD and the SBIBD are $bk = v\lambda$ and $\lambda(v-1) = r(k-1)$.

Note that information exchange in our key agreement protocol is based on the $(v, k+1, 1)$-design. Consequently, each participant can determine the intended message receivers or message senders based on the multicast model constructed by the $(v, k+1, 1)$-design.

D. Security Assumptions

1) Bilinear Diffie-Hellman (BDH) Problem: In $(G_1, G_2, \hat{e})$, the bilinear Diffie-Hellman problem is defined as follows. Given $\mathcal{P} \in G_1$ and $(\mathcal{P}, a\mathcal{P}, b\mathcal{P}, c\mathcal{P})$ for some $a, b, c \in Z^*_q$, compute $W = \hat{e}(\mathcal{P}, \mathcal{P})^{abc} \in G_2$ [19].

An algorithm $\mathcal{A}$ is said to have advantage ε in solving the BDH problem in $(G_1, G_2, \hat{e})$ if

$$\Pr[\mathcal{A}(\mathcal{P}, a\mathcal{P}, b\mathcal{P}, c\mathcal{P}) = \hat{e}(\mathcal{P}, \mathcal{P})^{abc}] \geq \varepsilon,$$

where the probability is based on the random choice of $a, b, c \in Z_q^*$, the random choice of $\mathcal{P} \in G_1$ and the random bits of $\mathcal{A}$.

Definition 2: The BDH assumption states that no polynomial-time algorithm $\mathcal{A}$ has an advantage of at least ε in solving the BDH problem in $(G_1, G_2, \hat{e})$, which means that this advantage is negligible.

Definition 3: A BDH parameter generator is a randomized algorithm that takes a security parameter $l \in Z^+$ as input and outputs the BDH parameters $\langle q, G_1, G_2, \hat{e}\rangle$ within polynomial time. Let *gen* represent the randomized algorithm; then, it can be described as $gen(1^l) = \langle q, G_1, G_2, \hat{e}\rangle$

2) q-Strong Diffie-Hellman (q-SDH) Problem: In $(G_1, G_2, \hat{e})$, let $\mathcal{P}_1$ and $\mathcal{P}_2$ be generators of G_1 and G_2, respectively. The q-strong Diffie-Hellman problem is defined as follows. Given $(\mathcal{P}_1, \mathcal{P}_2, \gamma\mathcal{P}_2, \gamma^2\mathcal{P}_2, \ldots, \gamma^q\mathcal{P}_2)$ as the input, output $(\frac{1}{\gamma+x} \cdot \mathcal{P}_1, x)$, where $x \in Z_p^*$ [21].

An algorithm $\mathcal{A}$ is said to have advantage ε in solving the q-SDH problem in $(G_1, G_2, \hat{e})$ if

$$\Pr[\mathcal{A}(\mathcal{P}_1, \mathcal{P}_2, \gamma\mathcal{P}_2, \gamma^2\mathcal{P}_2, \ldots, \gamma^q\mathcal{P}_2) = (\frac{1}{\gamma + x} \cdot \mathcal{P}_1, x)] \geq \varepsilon,$$

where the probability is with respect to the random choice of γ in Z_p^* and the random bits of $\mathcal{A}$.

Definition 4: The q-SDH assumption states that no polynomial-time algorithm $\mathcal{A}$ has an advantage of at least ε in solving the q-SDH problem in $(G_1, G_2, \hat{e})$, which means that this advantage is negligible.

3) Decision Linear (DL) Problem: In $(G_1, G_2, \hat{e})$, the DL problem is defined as follows. Given $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, a\mathcal{P}_1, b\mathcal{P}_2, c\mathcal{P}_3, \mathcal{P} \in G_1$, decide whether $a + b = c$.

An algorithm $\mathcal{A}$ is said to have advantage ε in solving the DL problem in G_1 if

$$\left|\begin{array}{l}\Pr[\mathcal{A}(\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, a\mathcal{P}_1, b\mathcal{P}_2, (a+b)\mathcal{P}_3) = yes] \\ -\Pr[\mathcal{A}(\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, a\mathcal{P}_1, b\mathcal{P}_2, \mathcal{P}) = yes]\end{array}\right| \geq \varepsilon,$$

where the probability is over the random choice of $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3, a\mathcal{P}_1, b\mathcal{P}_2, c\mathcal{P}_3, \mathcal{P} \in G_1$, the random choice of $a, b, c \in Z_q^*$ and the coin tosses of $\mathcal{A}$.

Definition 5: The DL assumption states that no polynomial-time algorithm $\mathcal{A}$ has an advantage of at least ε in solving the DL problem in $(G_1, G_2, \hat{e})$, which means that this advantage is negligible.

4) Elliptic Curve Discrete Logarithm Problem (ECDLP): The elliptic curve discrete logarithm problem is defined as follows. Given $\mathcal{P}_1, \mathcal{P}_2 \in G_1$, compute the integer k such that $\mathcal{P}_2 = k\mathcal{P}_1$ [23].

III. Problem Statement

A. System Model

The architecture of our cloud computing scheme is considered by combining with a concrete example: users with similar interests and specialists in the related areas hope to store and share their works in the cloud (e.g., results and discoveries). The system model contains three entities: cloud, group manager (e.g., an active specialist) and group members (e.g., bidder).

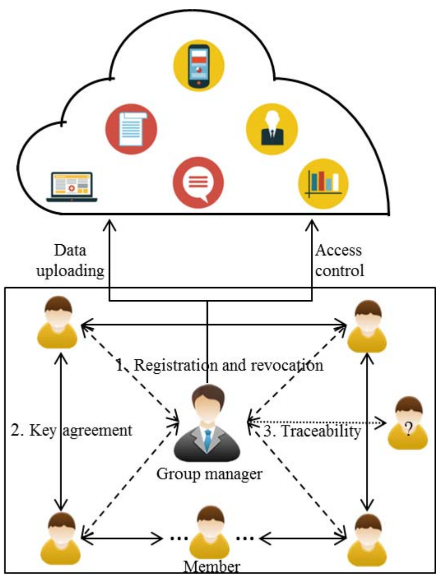


Fig. 1. The system model.

1) Cloud: provides users with seemingly unlimited storage services. In addition to providing efficient and convenient storage services for users, the cloud can also provide data sharing services. However, the cloud has the characteristic of honest but curious [11], [24]. In other words, the cloud will not deliberately delete or modify the uploaded data of users, but it will be curious to understand the contents of the stored data and the user's identity. The cloud is a semi-trusted party in our scheme.

2) Group Manager: is responsible for generating system parameters, managing group members (i.e., uploading members' encrypted data, authorizing group members, revealing the real identity of a member) and for the fault tolerance detection. The group manager in our scheme is a fully trusted third party to both the cloud and group members.

3) Members: are composed of a series of users based on the SBIBD communication model. In our scheme, members are people with the same interests (e.g., bidder, doctors, and businessmen) and they want to share data in the cloud.

The most worrying problem when users store data in the cloud server is the confidentiality of the outsourced data. In our system, users of the same group conduct a key agreement based on the SBIBD structure. Subsequently, a common conference key can be used to encrypt the data that will be uploaded to the cloud to ensure the confidentiality of the outsourced data. Attackers or the semi-trusted cloud server cannot learn any content of the outsourced data without the common conference key. In addition, anonymity is also a concern for users. Our scheme uses a technique called group signatures, which allows users in the same group to anonymously share data in the cloud.

The system model of the presented scheme is illustrated in Fig. 1. Firstly, users with the same interest register at the group manager so as to share data in the cloud. In addition, user revocation is also performed by the group manager. Secondly, all members of the group based on the SBIBD structure jointly negotiate a common session key, which can be used to encrypt or decrypt the outsourced data. Finally, when a dispute occurs, the group manager is able to reveal the real identity of the group member. Note that in our system model, data uploading and access control are performed by the group manager.

B. Threat Model

Under the system model mentioned above, we consider the following types of attack which may threat the security of the proposed scheme.

1) An attacker outside the group tries to reveal the common conference key to decrypt the outsourced data, who can be the external attacker or the curious cloud server.

2) A revoked member may collude with the cloud server to retain the outsourced data without the permission of the group manager. Since the common conference key is updated periodically, the revoked member may hold the current common conference key of the group even though he/she is added in the revocation list (RL) and the cloud server stores the outsourced data of the group.

3) A malicious member may generate different sub keys and transmit different messages (containing different sub keys) to different members during the key generation process. As a result, the final key calculated by different members is distinct.

4) An outside attacker or a revoked member may forge a valid signature of a legal group member. Besides, the attacker may try to reveal the real identity of the member.

In our system, the cloud server stores the group data will not change or delete data, which can be guaranteed by the data integrity verification [25]. However, the cloud server is curious about the stored data and he may collude with a revoked user.

C. Design Goals

In order to overcome the problems above, the main design goals of the proposed scheme include the following.

1) Dynamic Change: The intractable problem when sharing data in the cloud using the group manner is to ensure the security of the data when group members dynamically join and quit the group. A scheme that can support users' dynamic changes should guarantee that new users can access the previous data, whereas revoked users will not be able to obtain data in the cloud.

2) Data Confidentiality: In the cloud storage, data confidentiality requires that the outsourced data are invisible to the cloud server and to illegal users [26]. Taking advantage of the key agreement, a common conference key can be derived among all the group members such that they can encrypt their data prior to uploading it to the cloud. Moreover, with respect to the SBIBD, the communication and computation complexities for generating the common conference key are relatively small compared with [16] and [17].

3) Anonymity: Personal data are expected to be shared in the cloud without making the real identity public. Otherwise, few users are willing to share their information. Therefore, anonymity should be supported in the proposed scheme.

4) Traceability: Although data are shared anonymously in the cloud, a well-designed scheme should be able to locate the owner of the controversial data in disputes.

5) Fault-Tolerant Property: Fault tolerance guarantees that in the presence of malicious members, the remaining legitimate group members can still derive an effective common conference key.

IV. The Proposed Scheme

In this section, we present our scheme in detail. Benefiting from the SBIBD structure, the presented scheme can be applied for group data sharing with low communication and computation complexities. The detailed analysis and comparison are introduced in Section 6.

Our scheme can be divided into five parts: initialization, key generation, fault detection, file generation and key update, file access and traceability.

A. Initialization

The initialization is performed by the group manager, and this part includes parameter initialization, user registration and SBIBD construction.

1) Parameter Initialization:

- A security parameter l is selected as the input of the BDH parameter generator. Then, a bilinear map group system $\langle q, G_1, G_2, \hat{e}\rangle$ is returned.
- Two elements $H, H_0 \in G_1$ and two integers $\xi_1, \xi_2 \in Z_q^*$ are randomly chosen. Then, the group manager computes $H_1 = \xi_1 \cdot H_0$, $H_2 = \xi_2 \cdot H_0$ and $U = \xi_1^{-1} \cdot H,\ V = \xi_2^{-1} \cdot H$.
- A generator $\mathcal{G} \in G_1$ and an integer $\gamma \in Z_q^*$ are randomly selected, and $P = \gamma \cdot \mathcal{G}, W = \gamma \cdot P$ is computed. In addition, the group manager selects two hash functions h_1 and h_2, which map its arbitrary length to a nonzero integer and a nonzero point of G_1, respectively (i.e., $h_1 : \{0,1\}^* \rightarrow Z_q^*$, $h_2 : \{0,1\}^* \rightarrow G_1^*$).
- An additional integer $f \in Z_q^*$ is selected, and the system parameters $(\mathcal{G}, P, H, H_0, H_1, H_2, U, V, W, h_1, h_2, f, Enc_{\mathcal{K}}())$ are published; however, the parameter (γ, ξ_1, ξ_2) is kept private as the master key. Note that $Enc_{\mathcal{K}}()$ is a secure symmetric encryption algorithm with secret key $\mathcal{K}$.

2) User Registration: Each group member registers with the group manager with his/her identity ID_i. After receiving the identity information ID_i of the user, the group manager randomly selects a number $x_i \in Z_q^*$ and computes $A_i = \frac{1}{\gamma + x_i} \cdot P$. In addition, the group manager maps the identity information of member i to a nonzero point $Q_i = h_2(ID_i)$ in G_1 and computes $S_i = \gamma \cdot Q_i$, which can be used as the secret key for member i. Finally, the secret key for member i is (x_i, A_i, S_i), and (x_i, A_i, ID_i) is added to the group user list of the group manager.

3) SBIBD Construction: The communication model for the key generation is based on the SBIBD structure, which reduces the communication complexity and computational complexity for generating a common conference key. Meanwhile, the key agreement based on this structure also supports efficient key updating. Therefore, the structure of the SBIBD should first be constructed. After the registration, the group manager is responsible for building the structure of SBIBD according to the number of group members.

Algorithm 1 Generation of a $(v, k+1, 1)$-Design

```
Input: A prime number $k$.
Output: An SBIBD structure $B$.
  for $i = 0;\ i \le k;\ i++$ do
    for $j = 0;\ j \le k;\ j++$ do
      if $j == 0$ then
        $B_{i,j} = 0$;
      else
        $B_{i,j} = i \times k + j$;
      end if
    end for
  end for
  for $i = k+1;\ i \le k^2 + k;\ i++$ do
    for $j = 0;\ j \le k;\ j++$ do
      if $j == 0$ then
        $B_{i,j} = \lfloor (i-k-1)/k \rfloor + 1$;
      else
        $B_{i,j} = jk + 1 + MOD_k((i-k-1)+$
              $(j-1)\lfloor (i-k-1)/k \rfloor)$;
      end if
    end for
  end for
```

Algorithm 2 The Reconstruction of B

```
Input: An SBIBD structure $B$.
Output: An SBIBD structure $E$.
  $E_0 = B_0$; (step 1)
  for $t = 1;\ t \le k;\ t++$ do
    $E_t = B_{tk+1}$; (step 1)
    $B_{tk+1}[flag] = 1$;
    $E_{E_{t,t}} = B_{\lfloor (E_{t,t}-1)/k \rfloor}$; (step 2)
    $B_{tk+1}[flag] = 1$;
  end for
  for $i = k+1;\ i \le k^2 + k;\ i++$ do
    if $B_i[flag] \ne 1$ then
      $E_{B_{i,\lfloor (i-1)/k \rfloor}} = B_i$; (step 3)
    end if
  end for
```

The group manager first checks whether $k_n = (-1 + \sqrt{4n-1})/2$ is a prime number, where n is the number of registered members. If k_n is a prime number, then the number of members satisfies the structure of the SBIBD. Otherwise, the smallest prime number k_p, which is larger than $\lceil k_n \rceil$, should be found. Subsequently, an $(n, k_n+1, 1)$-design or an $(n_p, k_p+1, 1)$-design can be constructed according to *Algorithm 1* and *Algorithm 2*, where $n = k_n^2 + k_n + 1$, $n_p = k_p^2 + k_p + 1$.

Algorithm 1 creates the structure B of a $(v, k+1, 1)$-design. However, the key agreement protocol requires that in the structure of the SBIBD, each block B_i contains element i. Therefore, *Algorithm 2* is designed to transform B to E to satisfy this requirement. Here, the notation MOD_k represents the modular operation that takes the class residue as an integer in the range $0, 1, 2, \ldots, k-1$

The reconstructed E can be used as the communication model for key agreement. Based on this model, the key agreement protocol can be processed by v members, and a common conference key can be derived, which is shown in Section 4.2 in detail. Moreover, the structure of E can be determined by mathematical descriptions such that general formulas to compute the common conference key for each member can be derived. According to *Algorithm 1* and *Algorithm 2*, the structure of E is described by mathematical descriptions as follows.

Case 1: $E_0 = B_0 = \{0, 1, \ldots, k\}$

Case 2: $0 \le m \le k, 1 \le t \le k$

$$E_{t,m} = B_{tk+1,m} = \begin{cases} t, (m=0) \\ mk + 1 + MOD_k((t-1)(m-1)), (m > 0) \end{cases}$$

Case 3: $0 \le m \le k, t = E_{i,i}, (1 \le i \le k)$

$$E_{t,m} = B_{\lfloor (t-1)/k \rfloor, m} = \begin{cases} 0, (m=0) \\ \lfloor t - 1/k \rfloor k + m, (m > 0) \end{cases}$$

Case 4: $0 \le m \le k, t = B_{i,\lfloor (i-1)/k \rfloor}, (t \ne E_{i,i})$

$$E_{t,m} = B_{k(\lfloor (t-1)/k \rfloor + 1) + r, m} = \begin{cases} \lfloor \frac{t-1}{k} \rfloor, & (m=0) \\ mk+1+MOD_k((r-1)+(m-1)(\lfloor \frac{t-1}{k} \rfloor - 1)), & (m > 0) \end{cases}$$

Here, $r = -(k-2), -(k-3), \ldots, -1, 0$. Note that *Case 4* is correct due to the following reasons. In *Algorithm 1*, $B_{i,j}(k + 1 \le i \le k^2 + k)$ is calculated as Eq. 1.

$$B_{i,j} = j \times k + 1 + MOD_k((i-k-1) + (j-1) \times \frac{\lfloor i-k-1 \rfloor}{k}) \quad (1)$$

Let $t = B_{i,j}$; then, we have Eq. 2 as follows.

$$i = k \times (\left\lfloor \frac{t-1}{k} \right\rfloor + 1) + r \quad (2)$$

Note that r has $k-1$ different values, which describe the structure of the $(k-1)$ blocks of E.

The structure of E of a $(v, k+1, 1)$-design can be described in detail based on the mathematical descriptions in *Case 1*, *Case 2*, *Case 3* and *Case 4*. Members contained in E_t can be determined by the mathematical descriptions in the four different cases. Moreover, the structure of E is illustrated in detail in the Appendix.

B. Key Generation

In key generation, two rounds are required to generate a common conference key for team members, and the way that messages are exchanged is based on the structure E of the $(v, k+1, 1)$-design. Note that if k_n is not exactly a prime number, a larger SBIBD structure $(n_p, k_p+1, 1)$-design is constructed according to the prime k_p. However, if and only if the number of group members is exactly n_p can a common conference key be derived based on the $(n_p, k_p+1, 1)$-design.

Since the number of members cannot meet the structure of SBIBD, many messages will be missing in key generation. Therefore, some volunteers will be introduced to support data sharing with an arbitrary number of group members in the cloud, and a common conference key will be generated among them.

Specifically, the group manager selects a number of trusted volunteers to help group members complete the key agreement. These volunteers can be users in the system who enjoy a good reputation, and the number of volunteers is $n_p - n$. All selected users need to submit his/her ID_i to register with the group manager, and they will obtain his/her secret key as (A_i, x_i). After the volunteer selection and registration are completed, all group members and volunteers need two rounds of key agreement to generate a common conference key. Assuming that n users want to share data in a group and an $(n_p, k_p + 1, 1)$-design is constructed, the detailed key agreement process is described as follows.

Round 1: In round 1, each group member selects a random number r_i as a secret key and calculates $\mathcal{M}_i = \hat{e}(\mathcal{G}, r_i \mathcal{S}_i)$, which contributes to generating the common conference key among all members in the same group. Meanwhile, each member i runs *GenSign()* to generate a group signature on $\mathcal{M}_i$ with his/her secret key (A_i, x_i). For a volunteer, he/she does not need to calculate $\mathcal{M}$.

Subsequently, member i or volunteer i receives messages $D_j = \{M_j, \sigma_j\}$ from member j in the case that $j \in E_i$ $(j \neq i, j < n)$. Here, $j < n$ means that in round 1, the received messages only come from real members rather than from volunteers. According to the four mathematical descriptions of the structure of E, the key generation in round 1 is divided into four cases.

Case 1: Member 0 needs to receive messages from member $j(1 \leq j \leq k)$.

Case 2: For member $i(i \leq k)$, they need to receive messages from member $j(j = mk + 1 + MOD_k(i-1)(m-1), j \neq i, j < n)$.

Case 3: For member $i(i = E_{m,m}, i < n)$, they need to receive messages from member 0 and member $j(j = \lfloor (i-1)/k \rfloor k + m, j \neq i, j < n)$.

Case 4: For the remaining $k^2 - k$ members, they need to receive messages from $member_{\lfloor (i-1)/k \rfloor}$ and member $j, (j = mk + 1 + MOD_k((r-1) + (m-1)\lfloor (i-1)/k \rfloor - 1), j \neq i, j < n)$, where $r = -(k-2), -(k-3), \ldots, -1, 0$.

After every member and volunteer receives at most k messages that contribute to generating a common conference key from the intended members, they invoke *VerSign()* to verify the validity of the received messages. Moreover, if some members have been revoked, each member runs *VerRevo()* to authenticate the validity of their intended message senders.

After a successful verification of the received messages, each member and volunteer calculates $C_{j,i}, (i < n)$ as Eq. (3), which will be used in round 2 to generate a common conference key for member i. In fact, every $C_{j,i}, (i < n)$ contributes at most k messages for member i to generate a common conference key. In Eq. (3), $i < n$ means that volunteers do not need to receive messages in round 2 to derive a common conference key, while $x < n$ implies that volunteers do not contribute to the final common conference key.

Note that the calculation has the importance of volunteers in key generation such that an arbitrary number of users can be supported. The $C_{j,i}s, (i < n)$ calculated by volunteers are the missing messages when the number of users cannot exactly satisfy the structure of the SBIBD.

$$C_{j,i} = \prod_{x \in E_j - \{i\}} \mathcal{M}_x, \quad (i < n, x < n) \tag{3}$$

Round 2: Member i receives message $E_j = \{C_{j,i}, \sigma_j\}$ from member j or volunteer j in the case that $i \in E_j$, where $C_{j,i}$ is used to generate a common conference key and σ_j is the group signature generated by member j or volunteer j with *GenSign()*. Subsequently, each member i can verify the validity of the received messages through *VerSign()*. If the verification is successful, then the common conference

Algorithm 3 GenSign()

Input: Secret key (A_i, x_i), system parameters (P, U, V, H, W) and message M.

Output: A valid group signature σ on message M.

```
Selecting random integer
    $\alpha, \beta, t_\alpha, t_\beta, t_x, t_{\delta_1}, t_{\delta_2} \in Z_q^*$;
and setting numbers
    $\delta_1 = x_i\alpha, \delta_2 = x_i\beta$;
Computing the following values
    $\mathcal{T}_1 = \alpha \cdot U$
    $\mathcal{T}_2 = \beta \cdot V$
    $\mathcal{T}_3 = A_i + (\alpha + \beta) \cdot H$
    $\mathcal{R}_1 = t_\alpha \cdot U$
    $\mathcal{R}_2 = t_\beta \cdot V$
    $\mathcal{R}_3 = \hat{e}(T_3, P)^{t_x} \cdot \hat{e}(H, W)^{-t_\alpha - t_\beta} \cdot \hat{e}(H, P)^{-t_{\delta_1} - t_{\delta_2}}$
    $\mathcal{R}_4 = t_x \cdot T_1 - t_{\delta_1} \cdot U$
    $\mathcal{R}_5 = t_x \cdot T_2 - t_{\delta_2} \cdot V$
Then, generating a hash value
    $c = h_1(M, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_5)$;
Calculating the following values
    $s_\alpha = t_\alpha + c\alpha$
    $s_\beta = t_\beta + c\beta$
    $s_x = t_x + cx$
    $s_{\delta_1} = t_{\delta_1} + c\delta_1$
    $s_{\delta_2} = t_{\delta_2} + c\delta_2$
Generating the group signature
    $\sigma = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, c, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$;
Return $\sigma$;
```

Algorithm 4 VerSign()

```
Input: Message $M$ together with the signature $\sigma$ and some system parameters $(P, U, V, H, W)$.
Output: True or False.
  Calculating the following values
      $\tilde{\mathcal{R}}_1 = s_\alpha \cdot U - c \cdot \mathcal{T}_1$
      $\tilde{\mathcal{R}}_2 = s_\beta \cdot V - c \cdot \mathcal{T}_2$
      $\tilde{\mathcal{R}}_3 = \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3, P)^{s_x} \cdot \hat{e}(H, W)^{-s_\alpha - s_\beta} \cdot \hat{e}(H, P)^{-s_{\delta_1} - s_{\delta_2}}$
      $\tilde{\mathcal{R}}_4 = s_x \cdot \mathcal{T}_1 - s_{\delta_1} \cdot U$
      $\tilde{\mathcal{R}}_5 = s_x \cdot \mathcal{T}_2 - s_{\delta_2} \cdot V$
  Generating a hash value
      $c' = h_1(M, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \tilde{\mathcal{R}}_1, \tilde{\mathcal{R}}_2, \tilde{\mathcal{R}}_3, \tilde{\mathcal{R}}_4, \tilde{\mathcal{R}}_5)$;
  if $c = c'$ then
    return ‘True’;
  else
    return ‘False’;
  end if
```

Algorithm 5 VerRevo()

```
Input: A group signature $\sigma$, some system parameters $(H_0, H_1, H_2)$ and revocation keys $(A_1, A_2, \ldots, A_r)$ in the RL.
Output: Valid or invalid.
  Set $temp = \hat{e}(\mathcal{T}_1, H_1) \cdot \hat{e}(\mathcal{T}_2, H_2)$;
  for $i = 1$ to $r$ do
    if $temp == \hat{e}(\mathcal{T}_3 - A_i, H_0)$ then
      return ‘invalid’;
    end if
  end for
  return ‘valid’;
```

key is computed as Eq. (4) for member i.

$$\mathcal{K}=\mathcal{M}_i(\prod_{i\in E_j} C_{j,i})=\hat{e}(\mathcal{G}, r_i\mathcal{S}_i)\cdot(\prod_{i\in E_j} C_{j,i})=\hat{e}(\mathcal{G}, \sum_{i=0}^{n-1} r_i\mathcal{S}_i) \tag{4}$$

Finally, each member in the group obtains a common conference key, which can be used to ensure the security of the data sharing in the cloud.

In summary, a volunteer is a virtual member in a conference who helps the real members complete some calculations and transfer information. In addition, a volunteer does not need to obtain a conference key. Specifically, in round 1, volunteers only receive messages but do not send messages, and then they perform the corresponding calculations for real members. In round 2, volunteers only send their own calculations of messages but do not receive messages sent by real members. To understand our protocol well, the detailed key generation with volunteers is shown in the Appendix.

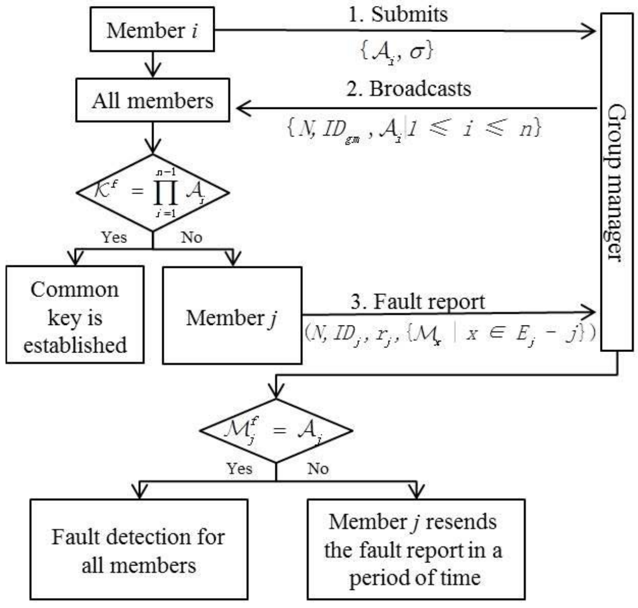

Member i
1. Submits
$\{\mathcal{A}_i, \sigma\}$
All members
2. Broadcasts
$\{N, ID_{gm}, \mathcal{A}_i | 1 \leqslant i \leqslant n\}$
Group manager
$\mathcal{K}^f = \prod_{i=1}^{n-1} \mathcal{A}_i$
Yes
No
Common key is established
Member j
3. Fault report
$(N, ID_j, r_j, \{\mathcal{M}_x \mid x \in E_j - j\})$
$\mathcal{M}_j^f = \mathcal{A}_j$
Yes
No
Fault detection for all members
Member j resends the fault report in a period of time

Fig. 2. Fault detection process.

C. Fault Detection

In practice, we cannot guarantee that all members in the group are honest. The existence of malicious members can seriously destroy the conference by generating different sub keys with the same group. Consequently, group members may generate different conference keys. Therefore, the group manager is required to ensure that each member only generates a unique sub key such that the conference will not be delayed or destroyed by malicious members. The detailed flow chart of the fault-tolerant property is shown in Fig. 2.

Specifically, during initialization, each member i needs to submit $\mathcal{A}_i = \mathcal{M}_i^f$ together with the group signature σ_i to the group manager. Here, f is a nonzero integer $f \in Z_q^*$. After the key agreement, the group manager broadcasts $\{N, ID_{gm}, \mathcal{A}_i | 0 \le i \le \upsilon - 1\}$ among all members, where $N = h_2(ID_i || ID_{gm} || t | 0 \le i \le n-1)$, ID_{gm} is the identity of the group manager and $\mathcal{A}_i$ denotes the verified unique sub key of all members. Then, every member verifies the authenticity of the common conference key $\mathcal{K}$ by checking whether the equation $\mathcal{K}^f = \prod_{i=0}^{n-1} \mathcal{A}_i$ holds. If the equation is true for all members, then a common conference key $\mathcal{K}$ is established among all members. Otherwise, fault detection is performed to resist the different key attack.

During fault detection, $member_j$, who finds that the above equation $\mathcal{K}^f = \prod_{i=0}^{n-1} \mathcal{A}_i$ does not hold, needs to send a fault report $(N, ID_j, r_j, \mathcal{M}_x, x \in E_j - j)$ to the group manager. The fault report contains the secret key of $member_j$ and the messages $\mathcal{M}$ that he received from the intended members. Subsequently, the group manager checks whether $\mathcal{M}_j^f = \hat{e}(\mathcal{G}, r_j\mathcal{S}_j)^f = \mathcal{A}_j$ holds. If the above does not hold, then the message that $member_j$ sends to other members is different from the message that $member_j$ submits to the group manager.

Thus, $member_j$ has to resend the fault report in a period of time $\triangle t$. Note that $member_j$ should be removed from the conference if the failure occurrence of $member_j$ exceeds a threshold τ or if $member_j$ did not resend the report within $\triangle t$. In this case, $member_j$ is either a malicious member or undergoing a denial of service attack. Otherwise, the fault detection phase should be processed among all the remaining members.

When the fault detection phase is conducted by all the remaining members, every member except $member_j$ should send $(N, ID_i, r_i, \mathcal{M}_x, x \in E_i - i)$ to the group manager. Subsequently, the group manager checks whether $\mathcal{M}_i^f = \hat{e}(\mathcal{G}, r_i\mathcal{S}_i)^f = \mathcal{A}_j$ holds. If the above does not hold, then $member_i$ has to resend the fault report in a period of time $\triangle t$. Similar to $member_j$, $member_i$ should be removed from the conference if the failure occurrence of $member_i$ exceeds a threshold τ or if $member_i$ did not resend the report within $\triangle t$. Otherwise, the group manager checks whether $\mathcal{M}_i = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)$ calculated by $member_i$ is equal to $\mathcal{M}_i^*$ received from $member_y(i \in E_y(y \neq i))$. If it is not, then $member_i$ is a malicious member. If yes for all the remaining members, then $member_j$ is a malicious member. The group manager removes the malicious member and denial of service member, and the protocol restarts. After the fault detection phase, an authenticated common conference key is derived among all the honest members. Following the proof of Theorem 2, the presented protocol can resist the different key attack and support the fault tolerance property.

According to Theorem 2, an honest member will not be removed from the conference, while a malicious member will be detected and removed from the conference. In addition, after some malicious members are removed from the conference, the common conference key cannot be derived because some messages are missing for generating the conference key. Then, the positions of the malicious members should be replaced by volunteers to ensure that the key agreement performs well. Therefore, the proposed scheme can not only resist the different key attack from malicious members but also provide the property of fault tolerance.

D. File Generation and Key Update

1) File Generation: When the member wants to upload the data file, the following operations are performed.

- The member encrypts the data file $\mathcal{F}$ by the symmetric encryption algorithm $Enc_{\mathcal{K}}()$ with the common conference key $\mathcal{K}$. The ciphertext $Cipher_G$ is calculated by $Cipher_G = Enc_{\mathcal{K}}(\mathcal{F})$. Then, the group member submits the encrypted data to the group manager, which consists of
$$(ID_{gm}, ID_{data}, Cipher_G, t_{data}, \sigma),$$
where σ is the group signature derived by the group member through *GenSign()* and t_{data} indicates the current time.
- The group manager verifies the validity of the member through *VerSign()* and *VerRevo()*. After a successful verification, the group manager selects two large primes p and q and calculates $n = pq$. Then based on the RSA public-key encryption [27], the group manager selects a relatively large integer e and computes the corresponding d, which satisfies $ed \equiv 1 \bmod (p-1)(q-1)$. The public key for the group manager is (e, n) while the private key is (d, n). With the public key (e, n), the group manager re-encrypts the encrypted message $Cipher_G$ as
$$Cipher_C = (Cipher_G)^e.$$
Here, $Cipher_C$ will be uploaded to the cloud by the group manager. In order to authorize a legal member, the group manager calculates
$$P_d = (\mathcal{G}^{d \cdot k}, d \cdot Z^k),$$
where k is a random number selected by the group manager and $Z = \hat{e}(\mathcal{G}, \mathcal{G})$. In addition, the group manager generates the group signature for the tuple $(Cipher_C, P_d)$ with his/her secret key (A_G, x_G) issued by the cloud previously, which can support the authentication of uploaded messages. Finally, the tuple $(Cipher_C, P_d)$ together with the signature σ is uploaded to the cloud.

2) User Revocation: In our scheme, the revocation of members is accomplished by the group manager, who is in charge of managing a RL. The RL consisting of tuple (A_r, x_r, t_r) represents that member r with private key (A_r, x_r) is revoked at time t_r. In addition, the RL is bound with ID_{gm} and $sig(RL)$ to denote the identity of the RL. Based on the RL and *VerRevo()*, a revoked member cannot be verified to access the cloud.

3) Key Update: Key update consists of two parts: the updating for the common conference key among group members and the updating of the group manager's private key (d, n).

Note that benefiting from the re-encryption and the access control, the outsourced data is still prevented from the collusion performed by the cloud server and the revoked member. In our scheme, the common conference key can be updated in a period of time. In addition, the RL can be emptied after the update of the common conference key.

On the other hand, the private key (d, n) of the group manager should be updated once the number of group members is changed.

- The group manager generates new public/private key pairs (e^*, n) and (d^*, n) and selects a new random k^*.
- The group manager calculates $P_d^* = (\mathcal{G}^{d^* \cdot k^*}, d^* \cdot Z^{k^*})$ and generates the group signature on the message through *GenSign()*. Then, P_d^* and the corresponding signature are uploaded to the cloud.
- The cloud replaces P_d by P_d^* after a successful verification of the signature. Then, it computes
$$Cipher_C^* = (Cipher_C)^{e^*/e}.$$

After the above operations, the key update of the group manager is accomplished. Note that the revoked member would pose no threat to the group data due to the re-encryption and the access control of the group manager.

E. File Access and Traceability

1) File Access: To obtain data stored in the cloud, the following operations are performed.

- The member sends a data request containing $(ID_{gm}, ID_{data}, t, \sigma)$ to the group manager, where ID_{data} is the identity of the shared group data, t denotes the current time and σ is the group signature on the message (ID_{gm}, ID_{data}, t).
- The group manager sends an authorization information $r_{GM\to M} = \mathcal{G}^{x_i/d_{cur}}$ to the cloud after a successful verification of *VerSign()* and *VerRevo()*. Here $r_{GM\to M}$ represents the authorization information from the group manager to the group member, x_i is the secret key of the group member and d_{cur} is the current private key of the group manager.
- After receiving the authorization information from the group manager, the cloud computes

$$\begin{aligned} per &= (\hat{e}(\mathcal{G}^{x_i/d_{cur}}, \mathcal{G}^{d_{cur}\cdot k_{cur}}), d_{cur}\cdot Z^{k_{cur}}) \\ &= (\hat{e}(\mathcal{G},\mathcal{G})^{x_i\cdot k_{cur}}, d_{cur}\cdot Z^{k_{cur}}) \end{aligned}$$

and responds with the requested data and per to the member.
- After receiving the requested data and per from the cloud, the member with his/her secret key x_i can obtain the re-encryption secret key of the group manager, which is calculated as

$$d_{cur} = (d_{cur}\cdot Z^{k_{cur}})/(\hat{e}(\mathcal{G},\mathcal{G})^{x_i\cdot k_{cur}})^{1/x_i}.$$

Finally, the authorized member can obtain his/her required group data by the re-encryption secret key of the group manager and the common conference key of the group.

2) Traceability: In our scheme, the group manager can track the real identity of the data owner when a dispute occurs. Specifically, when an argument is generated for a data file ID_{data}, the group manager will obtain a signature σ_{data} on the file. After verifying the correctness of the signature and a successful revocation verification, the group manager performs the following operations.

- Computing $A_i = T_3 - (\xi_1\cdot T_1 + \xi_2\cdot T_2)$ by his/her master key (ξ_1, ξ_2).
- Looking up his/her group user list to reveal the real identity of the data owner.

V. SECURITY ANALYSIS

In this section, we prove the security of our scheme in terms of data confidentiality, fault-tolerant property, anonymity, traceability and access control.

Theorem 1: The security of the outsourced data is based on the ECDLP [23] and the BDH assumption [19].

Proof: In our scheme with n users, the user and volunteer in the scheme are a probabilistic polynomial-time Turing machine, as is an adversary. A passive adversary (i.e., the could) is the person who attempts to learn information about the outsourced data by eavesdropping on the communication channel to acquire the common conference key. Note that an adversary has access to the system parameters $\{\mathcal{G}, P, h_2(ID_i)|0\le i\le n-1\}$, whereas the session key r_i for member i is protected from the adversary due to the ECDLP and the BDH assumption. According to [19], if $X \approx_{poly} Y$, then the presented scheme is secure against the passive attack. $X \approx_{poly} Y$ represents that two tuples of random variables $X = \{\mathcal{G}, P, h_2(ID_i), \hat{e}(\mathcal{G}, \sum_{i=1}^{n} r_i\mathcal{S}_i)|0\le i\le n-1\}$ and $Y = \{\mathcal{G}, P, h_2(ID_i), y|0\le i\le n-1, y\in Z_q^*\}$ are polynomially indistinguishable, where $y\in Z_q^*$ is a randomly chosen number. Specifically, if $X\approx_{poly} Y$, for all polynomial-time distinguishers, the probability of distinguishing X and Y is smaller than $\frac{1}{2}+\frac{1}{Q(l)}$ for all polynomials $Q(l)$. Here, $l\in Z^+$ is a security parameter in our key agreement scheme, which can determine the size of p defined in Definition 3. All algorithms run in probabilistic polynomial time with l as input. □

Lemma 1: If the condition of $X_i \approx_{poly} Y_i$ holds for all user$_i$, then $X\approx_{poly} Y$.

Proof: Let $X_i = \{\mathcal{G}, P_{PUB}, H_2(ID_i), \hat{e}(\mathcal{G}, \sum_{i=1}^{n} r_i\mathcal{S}_i)|0\le i\le n-1\}$ and $Y_i = \{\mathcal{G}, P_{PUB}, H_2(ID_i), y_i\}$.

$$\begin{aligned} X &= (\mathcal{G}, P_{PUB}, H_2(ID_i), \hat{e}(\mathcal{G}, \sum_{i=1}^{n} r_i\mathcal{S}_i)|0\le i\le n-1) \\ &= (\mathcal{G}, P_{PUB}, H_2(ID_0), H_2(ID_1), \ldots, H_2(ID_{n-1}), \\ &\quad \hat{e}(\mathcal{G}, r_0\mathcal{S}_0)\cdot \hat{e}(\mathcal{G}, r_1\mathcal{S}_1)\cdot\ldots\cdot \hat{e}(\mathcal{G}, r_{n-1}\mathcal{S}_{n-1})) \\ &= \prod_{i=0}^{n-1} X_i \\ Y &= (\mathcal{G}, P_{PUB}, H_2(ID_i), y|0\le i\le n-1, y\in Z_q^*) \\ &= (\mathcal{G}, P_{PUB}, H_2(ID_0), H_2(ID_1), \ldots, H_2(ID_{n-1}), y) \\ &= \prod_{i=0}^{n-1} Y_i \end{aligned}$$

Due to the discrete logarithm problem over elliptic curves being hard when p is more than 512-bits long and the BDH assumption, we have $X_i \approx_{poly} Y_i$. Thus, $\prod_{i=0}^{n-1} X_i \approx_{poly} \prod_{i=0}^{n-1} Y_i$ (i.e., $X\approx_{poly} Y$). □

Theorem 2: The proposed scheme can provide a fault-tolerant property and generate a common conference key among all members. The proof of Theorem 2 follows from lemma 2 and lemma 3.

Lemma 2: During fault detection, an honest member will not be removed by the group member.

Proof: For an honest member $member_h$, two situations should be taken into consideration. The first is that $member_h$ finds $\mathcal{K}^f \ne \prod_{i=0}^{n-1}\mathcal{A}_i$. Subsequently, the fault detection begins, and the fault report $(N, ID_h, r_h, \mathcal{M}_x, x\in E_h - h)$ of $member_h$ is sent to the group manager. Due to the honesty of $member_h$, there exists $member_i$ such that either $\mathcal{M}_i^g \ne \mathcal{A}_i$ or $\mathcal{M}_i \ne \mathcal{M}_i^*$. Therefore, $member_i$ is detected as a malicious member. The second situation is that $member_h$ is asked to submit $(N, ID_h, r_h, \mathcal{M}_x, x\in E_h - h)$ to the group manager.

Because of the honesty of $member_h$, the group manager finds $\mathcal{M}_h^f = \mathcal{A}_h$ and $\mathcal{M}_h = \mathcal{M}_h^*$. In conclusion, an honest member will never be removed by the group manager. □

Lemma 3: During fault detection, a malicious member who attempts to delay or destroy the conference will be removed by the group manager.

Proof: For a malicious member $member_m$, who attempts to delay or destroy the conference, three cases where $member_m$ attempts to sabotage the conference should be taken into consideration. The first case is that $member_m$ delays submitting a required message or continues sending invalid messages to the group manager. In this case, $member_m$ will be removed from the conference if the failure occurrences exceed a threshold τ or $member_m$ did not resend the report within $\triangle t$. The second case is that $member_m$ deliberately sends a fault report $(N, ID_m, r_m, \mathcal{M}_x, x \in E_m - m)$ to the group manager. In this case, the team leader finds $\mathcal{K}^f \neq \prod_{i=0}^{n-1} \mathcal{A}$, and all the remaining members have to send a fault report to the group manager. However, if $\mathcal{M}_i = \mathcal{M}_i^*$ holds for all the remaining members, then $member_m$ is detected as being malicious and is removed by the group manager. The third case is that $member_m$ performs the different key attack. $Member_m$ selects two different sub keys r_m and r_m^* and submits a false message to the team leader. Due to the different sub keys of $member_m$, the common conference key generated from different members is distinct. In this case, there is at least one of $\mathcal{M}_m^g$ not equal to $\mathcal{A}_m$ since $r_m \neq r_m^*$. $Member_i$ who detects $\mathcal{K}^f \neq \prod_{i=0}^{n-1} \mathcal{A}_i$ will report this fault to the group manager. Subsequently, $member_m$ is required to submit $(N, ID_m, r_m, \mathcal{M}_x, x \in E_m - m)$. Because $\mathcal{M}_m$ calculated by $member_m$ does not equal $\mathcal{M}_m^*$ received from other members, $member_m$ is detected as a malicious member. □

Theorem 3: Based on the DL assumption, the real identity of the signer is preserved, which implies anonymity [21] of our scheme.

Proof: Given the group signature σ, an entity can acquire parameters $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$. However, any entity cannot reveal the signer's identity $A_i = \mathcal{T}_3 - (\xi_1 \cdot \mathcal{T}_1 + \xi_2 \cdot \mathcal{T}_2)$; otherwise, the DL assumption will be in contradiction. □

Theorem 4: The real identity of the signer can be traced by the group manager, which implies traceability [21] of our scheme.

Proof: The group manager can acquire A_i by his/her master key (ξ_1, ξ_2) efficiently. Then, he can reveal the real identity ID_i of the signer by looking up the group user list maintained by himself. □

Theorem 5: Based on the q-SDH assumption, it is infeasible for an attacker to generate a valid group signature.

Proof: We present the proof of Theorem 5 under the random oracle model with the help of the Forking Lemma [28] as follows. Suppose that a polynomial-time algorithm $\mathcal{A}$ exists that can forge a valid group signature with a non-negligible probability. Given a random oracle h, algorithm $\mathcal{A}$ can generate two valid signatures $(M, \sigma_0, c, \sigma_1)$ and $(M, \sigma_0, c', \sigma'_1)$ according to the Forking Lemma, where

$$\begin{cases} \sigma_0 = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, c, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_5) \\ c = h(M, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_5) \\ c' = h'(M, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, c, \mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_4, \mathcal{R}_5) \\ \sigma_1 = (s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2}) \\ \sigma'_1 = (s'_\alpha, s'_\beta, s'_x, s'_{\delta_1}, s'_{\delta_2}) \end{cases}$$

and

$$\begin{cases} s_\alpha = t_\alpha + c\alpha, & s'_\alpha = t_\alpha + c'\alpha \\ s_\beta = t_\beta + c\beta, & s'_\beta = t_\beta + c'\beta \\ s_x = t_x + cx, & s'_x = t_x + c'x \\ s_{\delta_1} = t_{\delta_1} + c\delta_1, & s'_{\delta_1} = t_{\delta_1} + c'\delta_1 \\ s_{\delta_2} = t_{\delta_2} + c\delta_2, & s'_{\delta_2} = t_{\delta_2} + c'\delta_2 \end{cases}$$

Subsequently, algorithm $\mathcal{A}$ can obtain the secret key (x', A') by calculating $x' = \frac{\Delta s_x}{\Delta c}$ and $A' = \mathcal{T}_3 - \frac{\Delta s_\alpha + \Delta s_\beta}{\Delta c} \cdot H$, which can be used to forge a valid group signature. □

Theorem 6: The proposed scheme can achieve effective access control.

Proof: The proof of Theorem 6 follows from the following Lemma 4 and Lemma 5, which show that 1) a user who has successfully registered and has not been revoked is able to access the cloud, and 2) the revoked user is unable to access the cloud after the revocation. □

Lemma 4: Users that have not been revoked can access the cloud.

Proof: According to the algorithm *VerSign()*, for unrevoked users, the following equations are established: $\tilde{\mathcal{R}_1} = \mathcal{R}_1$, $\tilde{\mathcal{R}_2} = \mathcal{R}_2$, $\tilde{\mathcal{R}_3} = \mathcal{R}_3$, $\tilde{\mathcal{R}_4} = \mathcal{R}_4$, and $\tilde{\mathcal{R}_5} = \mathcal{R}_5$. Therefore, the hash value c is equal to $h_1(M, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \tilde{\mathcal{R}_1}, \tilde{\mathcal{R}_2}, \tilde{\mathcal{R}_3}, \tilde{\mathcal{R}_4}, \tilde{\mathcal{R}_5})$, and the algorithm returns 'true'. Specifically, we can prove $\tilde{\mathcal{R}_1} = \mathcal{R}_1$ due to the following reason:

$$\begin{aligned} \tilde{\mathcal{R}_1} &= s_\alpha \cdot U - c \cdot \mathcal{T}_1 \\ &= (t_\alpha + c\alpha) \cdot U - c \cdot \alpha \cdot U \\ &= t_\alpha \cdot U \\ &= \mathcal{R}_1 \end{aligned}$$

Similarly, we have $\tilde{\mathcal{R}_2} = \mathcal{R}_2$, $\tilde{\mathcal{R}_4} = \mathcal{R}_4$, $\tilde{\mathcal{R}_5} = \mathcal{R}_5$. Moreover, $\tilde{\mathcal{R}_3} = \mathcal{R}_3$ due to the following reason:

$$\begin{aligned} \tilde{\mathcal{R}_3} &= \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3, P)^{s_x} \cdot \hat{e}(H, W)^{-s_\alpha - s_\beta} \\ &\quad \cdot \hat{e}(H, P)^{-s_{\delta_1} - s_{\delta_2}} \\ &= \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3, P)^{r_x + cx} \cdot \hat{e}(H, W)^{-r_\alpha - c\alpha - r_\beta - c\beta} \\ &\quad \cdot \hat{e}(H, P)^{-r_{\delta_1} - cx\alpha - r_{\delta_2} - cx\beta} \\ &= \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3, xP)^c \cdot \hat{e}(-(\alpha + \beta)H, W + xP)^c \\ &\quad \cdot \hat{e}(\mathcal{T}_3, P)^{r_x} \cdot \hat{e}(H, W)^{-r_\alpha - r_\beta} \cdot \hat{e}(H, P)^{-r_{\delta_1} - r_{\delta_2}} \\ &= \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3, xP)^c \cdot \hat{e}(-(\alpha + \beta)H, W + xP)^c \cdot \mathcal{R}_3 \end{aligned}$$

TABLE I
PERFORMANCE AND FUNCTIONALITY COMPARISON

Scheme	Signature size	Operations for key generation	Communication cost	Pattern of information sharing	Security of the system
TPP	$3G_1^* + 2G_2^* + 1Z_q^* + \|m\|$	$2C_w + 4C_p + 1C_m + 1C_a$	$O(n^2)$	one-to-one	reliable
Mona	$3G_1^* + 6Z_q^* + \|m\|$	$3C_w + (r+5)C_p + 1C_m$	$O(n^2)$	one-to-many	vulnerable
Our	$3G_1^* + 6Z_q^* + \|m\|$	$1C_w + 1C_p + 2kC_a$	$O(n\sqrt{n})$	many-to-many	reliable

[1] G_1^*, G_2^*, Z_q^*: The length of nonzero points in G_1, G_2 and Z_q.
[2] $|m|$: The length of the message to be signed.
[3] C_w, C_p, C_m, C_a: Operations of the Weil pairing, point multiplication, modular exponentiation and point addition.
[4] r, k, n: The number of revoked users, the parameter in the $(v, k+1, 1)$-design and the number of all users.

$$\begin{aligned}
&= \left(\frac{\hat{e}(\mathcal{T}_3, W)}{\hat{e}(P, P)}\right)^c \cdot \hat{e}(\mathcal{T}_3 - (\alpha+\beta)H, W + xP)^c \\
&\quad \cdot \hat{e}(\mathcal{T}_3, W)^{-c} \cdot \mathcal{R}_3 \\
&= \left(\frac{\hat{e}(A_3, W + xP)}{\hat{e}(P, P)}\right)^c \cdot \mathcal{R}_3 \\
&= \left(\frac{\hat{e}(1/(x+\gamma)P, (\gamma + x)P)}{\hat{e}(P, P)}\right)^c \cdot \mathcal{R}_3 \\
&= \mathcal{R}_3
\end{aligned}$$

□

Lemma 5: Users that have been revoked by the group manager cannot access the cloud after their revocation.

Proof: According to the algorithm $VerRevo()$, if the group signature σ is generated by a revoked user, then there must be a A_i that makes equation $\hat{e}(\mathcal{T}_3 - A_i, H_0) = temp$. This equation is equal due to the following reason:

$$\begin{aligned}
\hat{e}(\mathcal{T}_3 - A_i, H_0) &= \hat{e}(A_i + (\alpha+\beta)\cdot H - A_i, H_0) \\
&= \hat{e}(\alpha \cdot H, H_0) \cdot \hat{e}(\beta \cdot H, H_0) \\
&= \hat{e}(\alpha \cdot U, \xi_1 \cdot H_0) \cdot \hat{e}(\beta \cdot V, \xi_2 \cdot H_0) \\
&= \hat{e}(\mathcal{T}_1, H_1) \cdot \hat{e}(\mathcal{T}_2, H_2)
\end{aligned}$$

□

VI. PERFORMANCE ANALYSIS AND EVALUATION

In this section, we first present a simplified comparison in TABLE I, and then we provide a thorough experimental evaluation of the proposed scheme.

A. Performance Analysis

We compare the proposed scheme with two recent protocols: Mona [17] and TPP [16]. According to [21], q is set as a random 160-bit prime. Each element in G_1 and G_2 is 161 and 1024 bits, respectively. Thus, regardless of the length of the message to be signed (i.e., $|m|$), the signature length of our scheme and Mona is 1443 bits or approximately 180 bytes, while TPP requires 2691 bits or roughly equal to 336 bytes. During the key generation, most of the computational cost comes from the Weil pairing, point multiplication and modular exponentiation, which are denoted as C_w, C_p, and C_m, respectively. Our scheme is designed based on the SBIBD structure and the two rounds key agreement. Note that compared to the above operations (i.e., C_w, C_p, C_m), the computational cost of point addition (i.e., C_a) can be ignored. Specifically, operations for key generation in Mona is $3C_w + (r+5)C_p + 1C_m$, where the number of point multiplications are advanced with the increase in the number of revoked users (i.e., r). Whereas operations for key generation in our scheme is $1C_w + 1C_p + 2kC_a$, where only the number of point additions are advanced with the increase in the prime number k. Note that based on the structure of the SBIBD, k is approximately $\sqrt{n}$. In addition, even though the number of operations for key generation in TPP is constant, TPP follows the one-to-one pattern. Consequently, TPP takes $n(2C_w + 4C_p + 1C_m + 1C_a)$ operations for key generation for n users. TABLE I clearly shows that our protocol has less computational cost than the other two protocols. Regarding to communication cost, in a $(v, k+1, 1)$-design with v members in our scheme, each member is connected to k members. In order to derive a common conference key, every member needs to receive k messages in each round based on the SBIBD structure. Note that the communication complexity of our mechanism is $O(vk)$ and only two rounds are required for messages exchange. Furthermore, with respect to Definition 1 of the SBIBD, the following equations are satisfied.

$$\begin{aligned}
k &= r \\
\lambda(v-1) &= r(k-1)
\end{aligned}$$

Here, $\lambda = 1$ in a $(v, k+1, 1)$-design with v members, and then $k \approx \sqrt{v}$. Thus, based on the SBIBD structure, the communication complexity of the proposed scheme is only $O(v\sqrt{v})$. Note that traditional group key agreement schemes require either $O(v^2)$ communication complexity or multiple rounds for generating a common conference key when there are v members in the group. Meanwhile, our protocol can also provide some features, such as many-to-many data sharing and reliability in the semi-trusted cloud. In TPP, the message is transmitted between two entities, whereas in our scheme and Mona, data and information exchange follow the many-to-many pattern. Considering the data sharing situation, in Mona, the cloud holding the vital RL can easily perform a collusion attack with any users (i.e., valid users and revoked users). The final symmetric key is derived from the RL, which makes the scheme vulnerable. However, in our scheme, the RL is used to achieve access control, which is not a threat to the security of the symmetric key or the system.

B. Performance Evaluation

To evaluate the performance, we simulate our scheme using the C programming language with the GMP Library (GMP-6.1.2) and PBC Library (pbc-0.5.14). All simulations

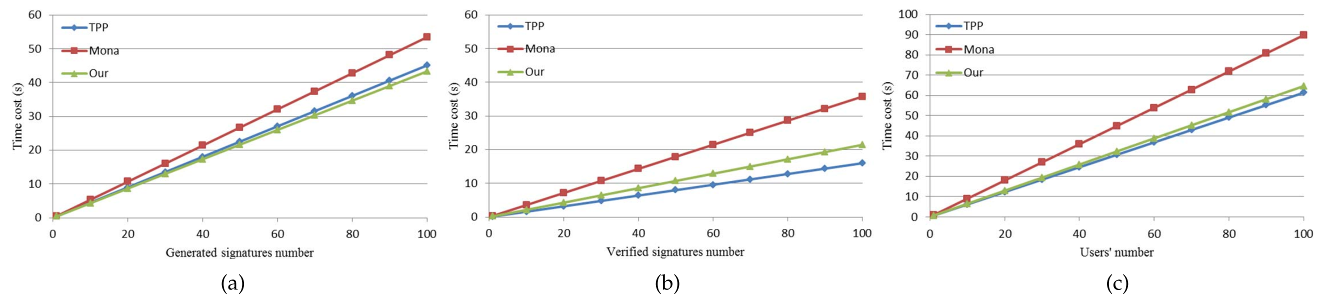


Fig. 3. Efficiency comparison for access control. (a) Time cost for signature generation in access control. (b) Time cost for signature verification in access control. (c) Total time cost for access control.

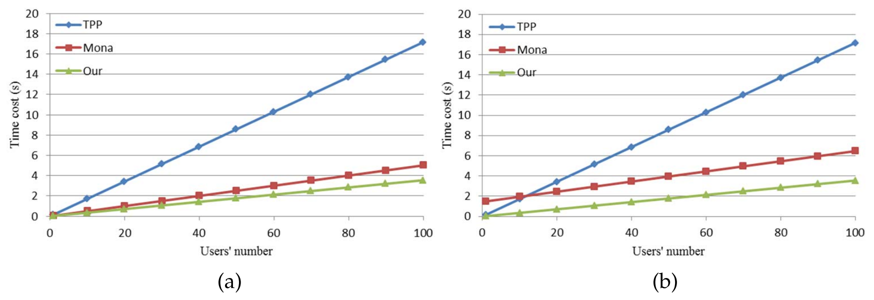


Fig. 4. Efficiency comparison for key generation. (a) Key generation without revoked users. (b) Key generation with 30 revoked users.

are implemented on a desktop computer with the following features: 1) CPU: Intel Pentium G3260 @ 3030 GHz; 2) Physical Memory: 4 GB; and 3) OS: Ubuntu 14.04 over VMware workstation full 12.5.2.

We provide the time cost evaluations for access control in TPP, Mona and our scheme in Fig. 3. We first present the time cost of signature generation and signature verification for all three schemes in Fig. 3(a) and Fig. 3(b), respectively. The X-axis represents the number n of generated/verified signatures. The Y-axis represents time cost (s) to generate/verify n signatures. In Fig. 3(a), the time cost of our scheme is more efficient than that of TPP and Mona, whereas in Fig. 3(b), the time cost of our scheme is slightly higher than that of TPP. We argue that the generated signature in our scheme can support authentication services in a group well, but TPP can only support authentication between two entities. In addition, it is easily observed that our scheme is more efficient than Mona in both signature generation and verification processes. As the conclusion, Fig. 3(c) depicts the total time cost for access control, where the X-axis represents the number n of users and the Y-axis represents time cost in s (i.e., seconds) to support access control for n users. Fig. 3(c) shows that our scheme is almost the same as TPP but superior to that of Mona.

Fig. 4 presents the comparison of the computational cost for key generation. The X-axis represents the number n of users, and the Y-axis represents the time cost (s) to generate the symmetric key for n users. Fig. 4(a) depicts the comparison for the three schemes with no revoked users, and Fig. 4(b) presents the comparison with 30 revoked users. It is obvious that our scheme is superior to TPP and Mona in both scenarios. This result is because TPP can only support key agreement between two entities, which asks for a large amount of computational and communication costs for key generation among multiple users. However, Mona adopts a centralized model (i.e., key distribution), where the revoked keys contribute to the final session key. This mechanism requires more computational and communication costs than the decentralized model (i.e., key agreement). In addition, the time cost in Mona will increase with the involvement of revoked users. Therefore, the proposed scheme is practical for sharing data in the cloud environment.

VII. Conclusion

In this paper, we present a secure and fault-tolerant key agreement for group data sharing in a cloud storage scheme. Based on the SBIBD and group signature technique, the proposed approach can generate a common conference key efficiently, which can be used to protect the security of the outsourced data and support secure group data sharing in the cloud at the same time. Note that algorithms to construct the SBIBD and mathematical descriptions of the SBIBD are presented in this paper. Moreover, authentication services and efficient access control are achieved with respect to the group signature technique. In addition, our scheme can support the traceability of user identity in an anonymous environment. In terms of dynamic changes of the group member, taking advantage of the key agreement and efficient access control, the computational complexity and communication complexity for updating the common conference key and the encrypted data are relatively low.

References

[1] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[2] X. Chen, J. Li, X. Huang, J. Ma, and W. Lou, "New publicly verifiable databases with efficient updates," *IEEE Trans. Depend. Sec. Comput.*, vol. 12, no. 5, pp. 546–556, Sep. 2015.

[3] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, “New algorithms for secure outsourcing of modular exponentiations,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

[4] J. Li, Y. Zhang, X. Chen, and Y. Xiang, “Secure attribute-based data sharing for resource-limited users in cloud computing,” *Comput. Secur.*, vol. 72, pp. 1–12, Jan. 2018, doi: 10.1016/j.cose.2017.08.007.

[5] J. Shen, T. Zhou, D. He, Y. Zhang, X. Sun, and Y. Xiang, “Block design-based key agreement for group data sharing in cloud computing,” *IEEE Trans. Depend. Sec. Comput.*, to be published, doi: 10.1109/TDSC.2017.2725953.

[6] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, “FRR: Fair remote retrieval of outsourced private medical records in electronic health networks,” *J. Biomed. Inform.*, vol. 50, pp. 226–233, Aug. 2014.

[7] J. Shen, S. Chang, J. Shen, Q. Liu, and X. Sun, “A lightweight multi-layer authentication protocol for wireless body area networks,” *Future Generat. Comput. Syst.*, vol. 78, pp. 956–963, Jan. 2016, doi: 10.1016/j.future.2016.11.033.

[8] Q. Liu, G. Wang, and J. Wu, “Time-based proxy re-encryption scheme for secure data sharing in a cloud environment,” *Inf. Sci.*, vol. 258, pp. 355–370, Feb. 2014.

[9] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, “Verifiable computation over large database with incremental updates,” *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[10] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, “An efficient public auditing protocol with novel dynamic structure for cloud data,” *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[11] S. Yu, C. Wang, K. Ren, and W. Lou, “Achieving secure, scalable, and fine-grained data access control in cloud computing,” in *Proc. Conf. Inf. Commun.*, 2010, pp. 1–9.

[12] X. Yi, “Identity-based fault-tolerant conference key agreement,” *IEEE Trans. Depend. Sec. Comput.*, vol. 1, no. 3, pp. 170–178, Jul. 2004.

[13] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, “Improved proxy re-encryption schemes with applications to secure distributed storage,” *ACM Trans. Inf. Syst. Secur.*, vol. 9, no. 1, pp. 1–30, 2006.

[14] M. Blaze, G. Bleumer, and M. Strauss, “Divertible protocols and atomic proxy cryptography,” *Eurocrypt*, vol. 1403, pp. 127–144, May 1998.

[15] P. Li, J. Li, Z. Huang, C.-Z. Gao, W.-B. Chen, and K. Chen, “Privacy-preserving outsourced classification in cloud computing,” in *Cluster Computing*, 2017, doi: 10.1007/s10586-017-0849-9.

[16] H. Wang, B. Qin, Q. Wu, and L. Xu, “TPP: Traceable privacy-preserving communication and precise reward for vehicle-to-grid networks in smart grids,” *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 11, pp. 2340–2351, Nov. 2015.

[17] X. Liu, Y. Zhang, B. Wang, and J. Yan, “Mona: Secure multi-owner data sharing for dynamic groups in the cloud,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1182–1191, Jun. 2013.

[18] J. Shen, S. Moh, and I. Chung, “Identity-based key agreement protocol employing a symmetric balanced incomplete block design,” *J. Commun. Netw.*, vol. 14, no. 6, pp. 682–691, Dec. 2012.

[19] D. Boneh and M. K. Franklin, “Identity-based encryption from the weil pairing,” in *Proc. Int. Cryptol. Conf. Adv. Cryptol.*, 2001, pp. 213–229.

[20] D. Chaum and E. Van Heyst, “Group signatures,” in *Proc. Workshop Theory Appl. Cryptogr. Technol.*, 1991, pp. 257–265.

[21] D. Boneh, X. Boyen, and H. Shacham, “Short group signatures,” in *Proc. Annu. Int. Cryptol. Conf.*, 2004, pp. 41–55.

[22] R. A. Brualdi, *Introductory Combinatorics*. Upper Saddle River, NJ, USA: Pearson Prentice-Hall, 2010.

[23] W. Stallings, “Cryptography and network security: Principles and practice,” *Int. Ann. Criminol.*, vol. 46, no. 4, pp. 121–136, 2008.

[24] R. Lu, X. Lin, X. Liang, and X. Shen, “Secure provenance: The essential of bread and butter of data forensics in cloud computing,” in *Proc. ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, Beijing, China, Apr. 2010, pp. 282–292.

[25] T. Jiang, X. Chen, and J. Ma, “Public integrity auditing for shared dynamic cloud data with group user revocation,” *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2363–2373, Aug. 2016.

[26] J. Shen, D. Liu, J. Shen, Q. Liu, and X. Sun, “A secure cloud-assisted urban data sharing framework for ubiquitous-cities,” *Pervasive Mobile Comput.*, vol. 41, pp. 219–230, Oct. 2017, doi: 10.1016/j.pmcj.2017.03.013.

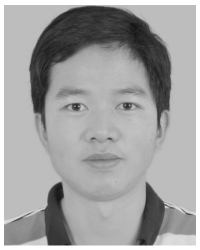
[27] R. L. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.

[28] D. Pointcheval and J. Stern, “Security arguments for digital signatures and blind signatures,” *J. Cryptol.*, vol. 13, no. 3, pp. 361–396, 2000.

Jian Shen received the M.E. and Ph.D. degrees in computer science from Chosun University, South Korea, in 2009 and 2012, respectively. Since 2012, he has been a Professor with the Nanjing University of Information Science and Technology, Nanjing, China. His research interests include public cryptography, cloud computing and security, data auditing and sharing, and information security systems.

Tianqi Zhou received the B.E. degree from the Nanjing University of Information Science and Technology, Nanjing, China, in 2016, where she is currently pursuing the master’s degree. Her research interests include computer and network security, security systems, and cryptography.

Xiaofeng Chen received the B.S. and M.S. degrees in mathematics from Northwest University, China, in 1998 and 2000, respectively, and the Ph.D. degree in cryptography from Xidian University in 2003. He is currently a Professor with Xidian University. He has published over 100 research papers in refereed international conferences and journals. His research interests include applied cryptography and cloud computing security. He has served as program/general chair or program committee member in over 30 international conferences. He is on the Editorial Board of *Security and Communication Networks*, *Computing and Informatics*, and the *International Journal of Embedded Systems*. His work has been cited over 3000 times in Google Scholar.



Jin Li received the B.S. degree in mathematics from Southwest University in 2002 and the Ph.D. degree in information security from Sun Yat-sen University in 2007. He is currently a Professor with Guangzhou University. He has been selected as one of Science and Technology New Star in Guangdong Province. He has published over 50 research papers in refereed international conferences and journals. His research interests include applied cryptography and security in cloud computing. He has served as the program chair or program committee member in many international conferences.

Willy Susilo (SM’01) received the Ph.D. degree in computer science from the University of Wollongong, Australia. He is currently a Professor and the Head of the School of Computing and Information Technology and the Director of the Institute of Cybersecurity and Cryptology, University of Wollongong. He has published over 400 research papers in the area of cybersecurity and cryptology. His main research interests include cybersecurity, cryptography, and information security. He was a recipient of the Australian Research Council (ARC) Future Fellow by the ARC and the Researcher of the Year Award by the University of Wollongong in 2016. He is the Editor-in-Chief of the *Information* journal. He has served as a program committee member in dozens of international conferences. He is currently serving as an Associate Editor in several international journals, including the *Computer Standards and Interface* (Elsevier) and the *International Journal of Information Security* (Springer). His work has been cited over 9000 times in Google Scholar.