This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

Block Design-based Key Agreement for Group Data Sharing in Cloud Computing

Jian Shen, *Member, IEEE,* Tianqi Zhou, Debiao He, Yuexin Zhang, Xingming Sun, *Senior Member, IEEE,* and Yang Xiang, *Senior Member, IEEE*

Abstract—Data sharing in cloud computing enables multiple participants to freely share the group data, which improves the efficiency of work in cooperative environments and has widespread potential applications. However, how to ensure the security of data sharing within a group and how to efficiently share the outsourced data in a group manner are formidable challenges. Note that key agreement protocols have played a very important role in secure and efficient group data sharing in cloud computing. In this paper, by taking advantage of the symmetric balanced incomplete block design (SBIBD), we present a novel block design-based key agreement protocol that supports multiple participants, which can flexibly extend the number of participants in a cloud environment according to the structure of the block design. Based on the proposed group data sharing model, we present general formulas for generating the common conference key $\mathcal{K}$ for multiple participants. Note that by benefiting from the $(v, k+1, 1)$-block design, the computational complexity of the proposed protocol linearly increases with the number of participants and the communication complexity is greatly reduced. In addition, the fault tolerance property of our protocol enables the group data sharing in cloud computing to withstand different key attacks, which is similar to Yi's protocol.

Index Terms—Key agreement protocol, symmetric balanced incomplete block design (SBIBD), data sharing, cloud computing.

1 Introduction

Cloud computing and cloud storagehave become hot topics in recent decades. Both are changing the way we live and greatly improving production efficiency in some areas. At present, due to limited storage resources and the requirement for convenient access, we prefer to store all types of data in cloud servers, which is also a good option for companies and organizations to avoid the overhead of deploying and maintaining equipment when data are stored locally. The cloud server provides an open and convenient storage platform for individuals and organizations, but it also introduces security problems. For instance, a cloud system may be subjected to attacks from both malicious users and cloud providers. In these scenarios, it is important to ensure the security of the stored data in the cloud. In [1], [2], [3], several schemes were proposed to preserve the privacy of the outsourced data. The above schemes only considered security problems of a single data owner. However, in some applications, multiple data owners would like to securely share their data in a group manner. Therefore, a protocol that supports secure group data sharing under cloud computing is needed.

A key agreement protocol is used to generate a common conference key for multiple participants to ensure the security of their later communications, and this protocol can be applied in cloud computing to support secure and efficient data sharing. Since it was introduced by Diffie-Hellman in their seminal paper [4], the key agreement protocol has become one of the fundamental cryptographic primitives. The basic version of the Diffie-Hellman protocol provides an efficient solution to the problem of creating a common secret key between two participants. In cryptography, a key agreement protocol is a protocol in which two or more parties can agree on a key in such a way that both influence the outcome. By employing the key agreement protocol, the conferees can securely send and receive messages from each other using the common conference key that they agree upon in advance. Specifically, a secure key agreement protocol ensures that the adversary cannot obtain the generated key by implementing malicious attacks, such as eavesdropping. Thus, the key agreement protocol can be widely used in interactive communication environments with high security requirements (e.g., remote board meetings, teleconferences, collaborative workspaces, radio frequency identification [5], cloud computing and so on).

The Diffie-Hellman key agreement [4] provides a way to generate keys. However, it does not provide an authentication service, which makes it vulnerable to man-in-the-middle attacks. This situation can be addressed by adding some forms of authentication mechanisms to the protocol, as proposed by Law *et al*. in [6]. In addition, the Diffie-Hellman key agreement can only support two participants. Subsequently, to solve the different key attacks

- *J. Shen is with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China and is 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.*
- *T. Zhou and X. Sun are with the School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing 210044, China. E-mail: tq_zhou@126.com, sunnudt@163.com.*
- *D. He is with the State Key Laboratory of Software Engineering, Computer School, Wuhan University, Wuhan 430072, China, and also with the State Key Laboratory of Cryptology, Beijing 100878, China. E-mail: hedebiao@163.com.*
- *Y. Zhang is with the School of Information Technology, Deakin University, Burwood, Victoria, Australia. E-mail: yuexinz@deakin.edu.au.*
- *Y. Xiang (corresponding author) is with the Digital Research & Innovation Capability Platform, Swinburne University of Technology, John Street, Hawthorn, Victoria 3122, Australia. E-mail: yxiang@swin.edu.au.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

from malicious conferees, who attempt to deliberately delay or destroy the conference, Yi proposed an identity-based fault-tolerant conference key agreement in [7]. Currently, many researches have been devoted to improving the security and communication efficiency of the key agreement protocol, which is covered in the literature [8], [9], [10], [11]. Note that in Chung and Bae's paper [12] and Lee *et al*.'s paper [13], block design is utilized in the design of an efficient load balance algorithm to maintain load balancing in a distributed system. Inspired by [12] and [13], we introduce the symmetric balanced incomplete block design (SBIBD) in designing the key agreement protocol to reduce the complexity of communication and computation. As far as we know, the work to design the key agreement protocol with respect to the SBIBD is novel and original.

1.1 Main Contributions

In this paper, we present an efficient and secure block design-based key agreement protocol by extending the structure of the SBIBD to support multiple participants, which enables multiple data owners to freely share the outsourced data with high security and efficiency. Note that the SBIBD is constructed as the group data sharing model to support group data sharing in cloud computing. Moreover, the protocol can provide authentication services and a fault tolerance property. The main contributions of this paper are summarized as follows.

1. Model of group data sharing according to the structure of the SBIBD is constructed. In this paper, a group data sharing model is established based on the definition of the SBIBD, which can be used to determine the way of communication among the participants. Regarding mathematical descriptions of the structure of the SBIBD, general formulas for computing the common conference key for multiple participants are derived.

2. Fault detection and fault tolerance can be provided in the protocol. The presented protocol can perform fault detection to ensure that a common conference key is established among all participants without failure. Moreover, in the fault detection phase, a volunteer will be used to replace a malicious participant to support the fault tolerance property. The volunteer enables the protocol to resist different key attacks [7], which makes the group data sharing in cloud computing more secure.

3. Secure group data sharing in cloud computing can be supported by the protocol. According to the data sharing model applying the SBIBD, multiple participants can form a group to efficiently share the outsourced data. Subsequently, each group member performs the key agreement to derive a common conference key to ensure the security of the outsourced group data. Note that the common conference key is only produced by group members. Attackers or the semi-trusted cloud server has no access to the generated key. Thus, they cannot access the original outsourced data (i.e., they only obtain some unintelligible data). Therefore, the proposed key agreement protocol can support secure and efficient group data sharing in cloud computing.

Notably, the above contributions substantially widen the field of applications of the key agreement protocol by applying an SBIBD with high security and flexibility. Moreover, the communication complexity is reduced without introducing extra computational complexity. Specifically, the communication complexity of our protocol is $O(n\sqrt{n})$, and the computational complexity is $O(nm^2)$. Here, n is the number of participants, and m is the extension degree of the finite field F_{p^m}, which is the space for rational points in a supersingular elliptic curve.

1.2 Organization

The remainder of this paper is organized as follows. Section 2 introduces related works. Section 3 briefly presents preliminaries and the system model. Section 4 describes the algorithm for constructing the SBIBD and depicts the group data sharing model. Section 5 shows the block design-based key agreement protocol with the general formulas for calculating the common conference key for multiple participants. Section 6 and Section 7 present the security and performance analyses, respectively. Finally, conclusions are drawn in Section 8. To understand our protocol well, the detailed process of the key agreement with multiple participants and a concrete example with 31 participants are provided in the Appendix.

2 Related Works

It is well known that data sharing in cloud computing can provide scalable and unlimited storage and computational resources to individuals and enterprises. However, cloud computing also leads to many security and privacy concerns, such as data integrity, confidentiality, reliability, fault tolerance and so on. Note that the key agreement protocol is one of the fundamental cryptographic primitives, which can provide secure communication among multiple participants in cloud environments.

In [14] and [15], based on symmetric-key cryptography, several schemes were proposed to enable efficient encryption of the outsourced data. However, encryption keys should be transmitted in a secure channel, which is not possible in practice, particularly in the open cloud environment. Since it was introduced in [16], resistance to compromised keys has been taken into consideration, which is an important issue in the context of cloud computing. Note that cloud storage auditing with verifiable outsourcing of key updates paradigm was proposed by Yu *et al*. in [17] to achieve resistance to compromised keys. In this paradigm, the third party auditor (TPA) takes responsibility for the cloud storage auditing and key updates. In particular, the TPA is responsible for the selection and distribution of the key. The key downloaded from the TPA can be used by the client to encrypt files that he will upload to the cloud. In contrast, the generation and distribution of the key is based on a centralized model in [17], which not only imparts a burden to the TPA but also introduces some security problems. In [18], a key agreement algorithm was exploited by De Capitani di Vimercati *et al.* to achieve data access when data are controlled by multiple owners. Therefore, the key agreement protocol can be applied in group data sharing to solve related security problems in cloud computing.

Following the first pioneering work for key agreement [4], many works have attempted to provide authentication services in the key agreement protocol. In [19],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

a public key infrastructure (PKI) is used to circumvent man-in-the-middle attacks. However, these protocols are not suitable for resource-constrained environments since they require executions of time-consuming modular exponentiation operations. Key agreement protocols that use elliptic curve cryptography (ECC) have been proposed in [20], [21]. These protocols are more efficient than the protocols that resort to the PKI because point additions or multiplications in elliptic curves are more efficient compared with the modular exponentiation. Moreover, based on the difficulty of solving the elliptic curve discrete logarithm problem (ECDLP), protocols that use ECC are more secure.

To avoid the requirement of the public key certificate, in 1984, identity-based cryptography (IBC) was proposed by Shamir [22]. However, it was not until 2001 that the first practical IBC scheme [10] was proposed by Boneh and Franklin. Due to the strict security proof and high efficiency, this scheme has received widespread recognition in academic fields. In the same year, a popular proof model for group key establishment was proposed by Bresson *et al*. [23]. In this protocol, to manage the complexity of definitions and proofs for the authenticated group Diffie-Hellman key exchange, a formal model was presented, where two security goals of the group Diffie-Hellman key exchange were addressed. However, some security properties are missing in [23], which are essential for preventing malicious protocol participants.

Note that all the above protocols have been proven and analyzed for security, but some of them can only be applied to the key agreement between two entities and need a large amount of resources to perform calculations. Recently, an identity-based authenticated key agreement protocol was proposed by Shen *et al*. in [9], which improves the efficiency of the conference key agreement and provides entity authentication services. However, there are some obstacles in Shen *et al*.'s protocol [9] in real applications. One is that the protocol only discusses a specific situation when the number of conferees is exactly 7. The other is that the protocol does not discuss the general situation and does not provide the key agreement process for multiple participants, which makes the protocol lack flexibility and practicability.

Motivated by the above observation, the key agreement protocol is applicable to support data sharing in cloud computing for the following reasons.

1. The generation of a common conference key is performed in a public channel, which is suitable for cloud computing environments.

2. The key agreement protocol can support and provide secure data sharing for multiple data owners within a group, where the data sharing follows a many-to-many pattern. Compared with the one-to-many pattern, the many-to-many pattern in group data sharing provides higher efficiency in the environment of cooperative storage.

3. The key agreement protocol is based on a decentralized model, where a trusted third party is not required. This means that every data owner in a group fairly contributes and determines the common conference key such that the outsourced data are controlled by all the data owners within a group.

Therefore, we design a block design-based key agreement protocol for data sharing in cloud computing. First, we propose an algorithm to construct the $(v, k+1, 1)$-design. Then, with respect to the mathematical description of the structure of the $(v, k+1, 1)$-design, general formulas for generating the common conference key $\mathcal{K}$ for multiple participants are derived. Namely, the proposed protocol supports multiple participants. We believe that our contributions can widen the application scope of the key agreement protocol in cloud computing employing an SBIBD.

3 Preliminaries and System Model

3.1 Cryptographic Bilinear Maps

Modified Weil pairing [10] is an example of a cryptographic bilinear map. One way to construct this map is described as follows. Let p be a prime such that $p = 6q - 1$ for some prime q and E be a supersingular elliptic curve defined by the Weierstrass equation $y^2 = x^3 + 1$ over F_p. The group of rational points $E(F_p) = \{(x, y) \in F_p \times F_p : (x, y) \in E\}$ forms a cyclic group of order $p + 1$. Furthermore, because $p + 1 = 6q$ for some prime q, the group of points of order q in $E(F_p)$ forms a cyclic subgroup, denoted as G_1. Further discussion of the Weil pairing is shown in the literature [8].

Definition 1. Let $\mathcal{G}$ be a generator of G_1, and let G_2 be the subgroup of F_{p^2} containing all elements of order q. A modified Weil pairing is a map $\hat{e} : G_1 \times G_1 \to G_2$, which has the following properties for points in $E(F_p)$:

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. Non-commutative: For any $\mathcal{P}, \mathcal{Q} \in G_1$, $\mathcal{P} \neq \mathcal{Q}$, $\hat{e}(\mathcal{P}, \mathcal{Q}) \neq \hat{e}(\mathcal{Q}, \mathcal{P})$.

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

5. For any $\mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}_1, \mathcal{Q}_2 \in G_1$, we have
$\hat{e}(\mathcal{P}_1 + \mathcal{P}_2, \mathcal{Q}_1) = \hat{e}(\mathcal{P}_1, \mathcal{Q}_1) \cdot \hat{e}(\mathcal{P}_2, \mathcal{Q}_1)$
$\hat{e}(\mathcal{P}_1, \mathcal{Q}_1 + \mathcal{Q}_2) = \hat{e}(\mathcal{P}_1, \mathcal{Q}_1) \cdot \hat{e}(\mathcal{P}_1, \mathcal{Q}_2)$

3.2 Security Assumption

Security is one of the most essential conditions that a good cryptographic algorithm or protocol should first meet. Studies on safety issues can boil down to the security model. The attacker's ability and the goal of security achieved can be well reflected by the correct and appropriate security model. In this paper, we use the security model defined in the literature [9]. Note that the security of our protocol relies on a variant of the computational Diffie-Hellman (CDH) assumption: the bilinear Diffie-Hellman (BDH) assumption, which is defined as follows. According to the proof in [9], the presented protocol can resist both passive attacks and active attacks. Many formal security analyses of the key agreement protocol can be found in the literature [11].

Definition 2. In $(G_1, G_2, \hat{e})$, the BDH problem is defined as follows. Given $\mathcal{G} \in G_1$ and $(\mathcal{G}, a\mathcal{G}, b\mathcal{G}, c\mathcal{G})$ for some $a, b, c \in Z_q^*$, compute $W = \hat{e}(\mathcal{G}, \mathcal{G})^{abc} \in G_2$ [10].

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

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

where $\varepsilon > 0$ and the probability is based on the random choice of $a, b, c \in Z_q^*$, the random choice of $\mathcal{G} \in G_1^*$ and the random bits of $\mathcal{A}$.

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.

3.3 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 useful for a particular application. Definition 3 defines the balanced incomplete block design (BIBD) in detail below [12], [13], [24].

Definition 3. Let $V = \{0, 1, 2...v-1\}$ be a set of v elements and $B = \{B_0, B_1, B_2...B_{b-1}\}$ 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 meet the condition of $k < v$. Thus, no block contains all the elements of the set V.

4. Parameters b and v of V meet 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, k implies the number of elements in each block, and 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, 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 group data sharing decentralized model, where k is a prime number and $\lambda = 1$. The reason for why the $(v, k+1, 1)$-design is chosen 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 $bk = vr$ 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 group data sharing model constructed by the $(v, k+1, 1)$-design.

In Section 5 will be noted that information exchange in our key agreement protocol is based on the $(v, k+1, 1)$-design and the detailed processes are described.

3.4 System Model and Adversary Model

3.4.1 System Model

The system model of our group data sharing scheme in cloud computing is illustrated in Fig. 1. A TPA, cloud and users are involved in the model, where the TPA is responsible for cloud storage auditing, fault detection and generating the system parameters. The cloud, who is a semi-trusted party, provides users with data storage services and

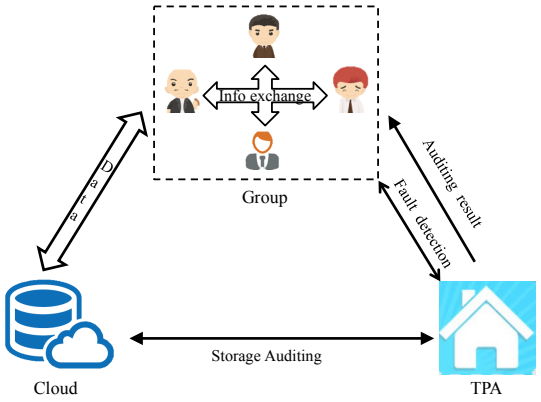


Fig. 1: System model of data sharing in cloud computing.

download services. Users can be individuals or staff in a company. To work together, they form a group, upload data to the cloud server and share the outsourced data with the group members. In practice, users can be mobile Android devices, mobile phones, laptops, nodes in underwater sensor networks and so forth.

Moreover, the group data sharing model is based on the SBIBD, where a trusted third party is not required. The construction of the SBIBD group data sharing model is described in detail in Section 4. With respect to this model, all the participants exchange messages from intended entities according to the structure of the SBIBD to determine a common conference key. In addition to participants, volunteers and adversaries are also included in the presented protocol, and all of them run as a probabilistic polynomial-time Turing machine. Two types of adversaries may be involved in the protocol: passive adversaries and active adversaries. A passive adversary is a person who attempts to learn information about the conference key by eavesdropping on the multicast channel, whereas an active adversary is a person who attempts to impersonate a participant or disrupt a conference. Note that the generation and update of the key are accomplished by the participants. Moreover, with the fault tolerance property of our protocol, the participants are able to ascertain the correctness of the common conference key. Since the storage auditing can follow the state of the art auditing protocols (e.g., [25]), we only focus on the design of group data sharing scheme in cloud computing in the paper.

3.4.2 Adversary Model

The adversary model determines the capabilities and possible actions of the attacker. Similar to [11], [26] and [27], the adversary model is defined as follows.

1. The adversary reveals a long-term secret key of a participant in a conference and then impersonates others to this participant.

2. The adversary reveals some previous session keys and then learns the information about the session key of a fresh participant. Consequently, the adversary can impersonate the fresh participant with the session key to others.

3. The adversary reveals the long-term keys of one or more participants in the current run. Then, the adversary attempts to learn the previous session key.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

4. A malicious participant chooses different sub keys, generates different signatures and broadcasts the messages to the corresponding participants, which makes the conference key derived by different participants distinct.

4 The Construction of the Group Data Sharing Model

To support a group data sharing scheme for multiple participants applying an SBIBD, we design an algorithm to construct the $(v, k+1, 1)$-design. Moreover, the constructed $(v, k+1, 1)$-design requires some transformations to establish the group data sharing model such that v participants can perform the key agreement protocol.

4.1 Construct the $(v, k+1, 1)$-design

In our group data sharing model, the parameters of the SBIBD have some specific meanings. In a $(v, k+1, 1)$-design, v denotes the number of participants and the number of blocks. Every block embraces $k+1$ participants, and every participant appears $k+1$ times in these v blocks. Furthermore, every two participants appear simultaneously in exactly one of the v blocks. Following papers [12] and [13], *Algorithm 1 is designed to construct the structure of a $(v, k+1, 1)$-design. First, a prime number k is selected. Then, the number of participants is determined by the value of k, which is computed as $v = k^2 + k + 1$†. Finally, according to Definition 3, $V = \{0, 1, 2, ..., v-1\}$ represents the set of v participants, whereas $B = \{B_0, B_1, B_2, ..., B_{v-1}\}$ implies v blocks constituted by these v participants. Note that the block is defined as $B_i = \{B_{i,0}, B_{i,1}, B_{i,2}, ..., B_{i,k}\}$, which means each block embraces $k+1$ participants, and $B_{i,j}$ denotes which participant is contained in the j^{th} column of the i^{th} block. Sometimes we will consider blocks organized as a matrix in which column j is composed by elements $B_{i,j}$ for $i = 0, 1, 2, ..., k$ and row i is composed by elements $B_{i,j}$ for $j = 0, 1, 2, ..., k$. The structure of the $(v, k+1, 1)$-design is constructed by* Algorithm 1, *which outputs numbers $B_{i,j}$ for $i = 0, 1, ..., k^2 + k$ and $j = 0, 1, ..., k$.*

In *Algorithm 1*, the notation MOD_k represents the modular operation that takes the class residue as an integer in the range $0, 1, 2, ..., k-1$. Based on *Algorithm 1*, we can create the structure of a $(v, k+1, 1)$-design that involves v participants. Moreover, *Algorithm 1* can directly determine which participant should be involved in each block. For example, taking the $(13, 4, 1)$-design into consideration, where 13 participants are involved in this structure, we can decide which participant should be contained in the 3^{rd} column of the 8^{th} block by computing

$$\begin{aligned} B_{7,2} &= jk + 1 + MOD_k(i - j + (j-1)\lfloor (i-1)/k \rfloor) \\ &= 2 \cdot 3 + 1 + MOD_3(7 - 2 + (2-1)\lfloor (7-1)/3 \rfloor) \\ &= 7 + MOD_3(5 + 1 \cdot 2) \\ &= 7 + 1 = 8. \end{aligned}$$

Therefore, from the above calculation, it is concluded that $participant_8$ is contained in the 3^{rd} column of the 8^{th} block. Here, $participant_i$ represents the i^{th} participant.

†. From here to the end of the paper, the value of v is $v = k^2 + k + 1$ and $(v, k+1, 1)$-design is used to represent $(k^2+k+1, k+1, 1)$-design for brevity.

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

```
for i = 0; i ≤ k; i++ do
  for j = 0; j ≤ k; j++ do
    if j == 0 then
      B_{i,j} = 0;
    else
      B_{i,j} = ik + j;
    end if
  end for
end for
for i = k+1; i ≤ k^2 + k; i++ do
  for j = 0; j ≤ k; j++ do
    if j == 0 then
      B_{i,j} = ⌊(i − 1)/k⌋;
    else
      B_{i,j} = jk + 1 + MOD_k(i − j + (j − 1)⌊(i − 1)/k⌋);
    end if
  end for
end for
```

Note that *Algorithm 1* is an optimization of the algorithm in [12] and the proof of the correctness follows the same lines than the proof in [12] and [13]. The structure created by *Algorithm 1* can be proven to satisfy the conditions of the $(v, k+1, 1)$-design, which means that each participant of V appears exactly $k+1$ times in B and that each pair of participants of V appears exactly once in B. These properties can be utilized to design the group data sharing model, which can diminish the communication cost of the proposed protocol. The detailed process of the protocol and the corresponding performance analysis based on the model can be found in Section 5 and Section 7, respectively.

Definition 4. In our $(v, k+1, 1)$-design of an SBIBD, a sector is a collection of blocks defined by $S_x = \{B_i : B_{i,0} = x\}$ for $x = 0, 1, 2, ..., k$. Sector $S_0 = \{B_0, B_1, B_2, ..., B_k\}$ is formed by $k+1$ blocks, and sector $S_j = \{B_{kj+1}, B_{kj+2}, B_{kj+3}, ..., B_{k(j+1)}\}$ is formed by k blocks for $j = 1, 2, 3, ..., k$.

For example, in a $(v, k+1, 1)$-design, $S_1 = \{B_{k+1}, B_{k+2}, ..., B_{2k}\}$.

Lemma 1. *In S_0 with $(k+1) \cdot (k+1)$ elements, element 0 appears k+1 times in the first column of S_0, and the remaining $k^2 + k$ elements $\{1, 2, ..., k^2 + k\}$ appear exactly once in S_0 in order.*

Proof. According to *Algorithm 1*, when $i = 0, 1, ..., k$, if $j = 0$, then $B_{i,j} = 0$. Therefore, element 0 appears $k+1$ times in the $k+1$ blocks (i.e., $B_{0,0}, B_{1,0}, ..., B_{k,0}$) of S_0. If $j \neq 0$, then $B_{i,j} = ik + j$, which implies that $B_{i,j+1} = B_{i,j} + 1 (for\ j = 0, 1, ..., k-1)$ and $B_{i+1,1} = B_{i,k} + 1 (for\ i = 0, 1, ..., k-1)$. Given $B_{i,j} = 0$, we have the progression $B_{0,0}, B_{0,1}, B_{0,2}, ..., B_{0,k}, B_{1,1}, B_{1,2}, ..., B_{1,k}, B_{2,1}, B_{2,2}, ..., B_{2,k}, ..., B_{k,1}, B_{k,2}, ..., B_{k,k}$ is the arithmetic progression $0, 1, 2, 3, 4, ..., k^2 + k$. It is concluded that the remaining $k^2 + k$ elements $\{1, 2, ..., k^2 + k\}$ appear exactly once in S_0 in order. □

In any sector S_x with k or $k+1$ blocks, the first element of each block has the same value as x. Based on Lemma 1,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

in sector S_0, the first element of these $k+1$ blocks is 0. Then, in the last k^2 blocks, $B_{i,j} = \lfloor (i-1)/k \rfloor$ if $j = 0$. Based on Definition 4, the index of each sector is $x = \lfloor i - 1/k \rfloor$, which is equal to $B_{i,j}$. Therefore, the first element of each block has the same value as $\lfloor (i-1)/k \rfloor$, namely, x.

Lemma 2. *In sector $S_x (x \neq 0)$, except for the first element of each block that are the same, the set of the other k^2 elements is equal to $V - B_0$.*

Proof. Based on Definition 4, in S_x, the first element of each block has the same value as x. Then, according to *Algorithm 1*, the remaining k^2 elements of S_x are calculated as $B_{i,j} = jk + 1 + MOD_k(i - j + (j-1)\lfloor (i-1)/k \rfloor)$.

Due to the property of the modular arithmetic and elementary properties of floor function, the values of these k^2 elements are between $k+1$ and k^2+k and these k^2 values are distinct. Note that $V = \{0, 1, ..., k^2 + k\}$ and $B_0 = \{0, 1, ..., k\}$. Therefore, the set of the other k^2 elements in S_x is equal to $V - B_0 = \{k+1, k+2, ..., , k^2+k\}$.

In addition, the detailed proof for the fact that the values of these k^2 elements are between $k+1$ and k^2+k and these k^2 values are distinct is given as follows.

For a given value between $k+1$ and k^2+k, say $k+n$ for $1 \leq n \leq k^2$ we look for an index $j, 0 \leq j \leq k$ such that $k+n$ is in sector S_x for a fixed $x = 0, 1, 2..., k$. On the other hand sector $S_x = \{B_{kx+1}, B_{kx+2}, B_{kx+3}, ..., B_{k(x+1)}\}$, then we also look for m and $j, 1 \leq m \leq k$ such that $B_{kx+m,j} = k+n$. We obtain $jk+1+MOD_k(kx+m-j+(j-1))\lfloor \frac{kx+m-1}{k} \rfloor) = k+n$ and then $MOD_k(m-j+(j-1)x) = k+n-jk-1$ must be a number in the range $0, 1, 2..., k-1$, that is, $0 \leq k+n-jk-1 < k$ obtaining $0 \leq 1 + \frac{n-1}{k} - j < 1$ and then $j - 1 \leq \frac{n-1}{k} < j$ equivalent to $j - 1 = \lfloor \frac{n-1}{k} \rfloor$. We can conclude that the value j is unique and $j = 1 + \lfloor \frac{n-1}{k} \rfloor$. From the value of j, $MOD_k(m-j+(j-1)x) = k+n-jk-1$ and taking into account that $1 \leq m \leq k-1$, we can obtain a unique possible value $m-1 = MOD_k((1-j)x+j+n-2)$, that is $m = 1 + MOD_k((1-j)x+j+n-2)$.

□

Lemma 3. *In sector $S_x (x \neq 0)$ with k blocks, the set of the k elements of the x^{th} column is equal to the index set of the k blocks in S_x.*

Proof. According to Definition 4, in S_x, the index set of k blocks is $\{xk+1, xk+2, ..., xk+k\}$. Then, elements of the x^{th} column of S_x are computed as $B_{i,j} = jk+1+MOD_k(i-j+(j-1)\lfloor (i-1)/k \rfloor)$. The x^{th} column of S_x means that $j = x$. Therefore, $B_{i,j} = xk+1+MOD_k(i-x+(x-1)\lfloor (i-1)/k \rfloor)$. Based on the modular arithmetic, when i takes a value of the set $[xk+1, xk+k]$, we have $B_{i,j} = xk+1, xk+2, ..., xk+k$, which is equal to the index set of the k blocks in S_x. □

4.2 Design of the Group Data Sharing Model

Through *Algorithm 1*, the structure B of the $(v, k+1, 1)$-design is constructed for v participants, which satisfies the properties of an SBIBD. However, to generate a common conference key for the v participants, the structure of the $(v, k+1, 1)$-design should have the property that each block B_i embraces $participant_i$. Here, B_i is the i^{th} block of the structure of the $(v, k+1, 1)$-design, and the order of the appearance of these v blocks is represented by i. Note that the structure B constructed by *Algorithm 1* does not have the required property. Thus, some transformations of the structure of B are needed. Based on Lemma 1, Lemma 3 and Definition 4, the v blocks of B can be reconstructed to derive a new structure E of the $(v, k+1, 1)$-design such that each block E_t embraces $participant_t$. Notably, the adjustment of the structural order among blocks in the block design does not affect its characteristics and this transformed structure E is therefore a standard one in the theory of SBIBDs. *Algorithm 2* can be employed to accomplish the reconstruction of B to E after the structure of B is created by *Algorithm 1*. During the reconstruction process, a flag bit for each block B_i is required to indicate whether B_i is transformed. The flag bit is denoted as $B_i[flag]$, which is 0 if B_i has not been transformed and is 1 otherwise. The detailed reconstruction process is given as follows. To make the process from B to E clear, a concrete example is shown in Fig. 2. In Fig. 2, a $(13, 4, 1)$-design is constructed by *Algorithm 1* first, which is depicted on the left of Fig. 2. Then, *Algorithm 2* with three steps is used to accomplish the reconstruction of B to E. Note that the transformed structure of E is shown on the right of Fig. 2.

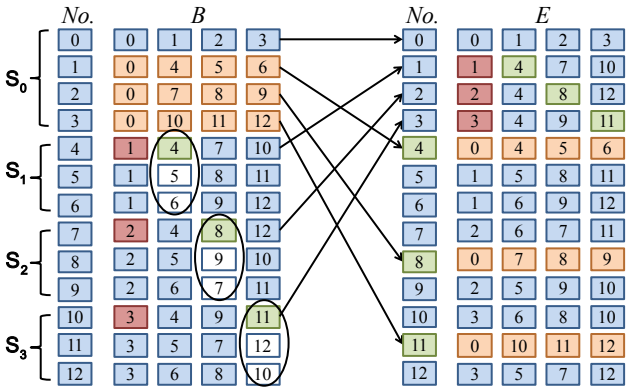


Fig. 2: The reconstruction process of B to E.

Algorithm 2 The reconstruction of B.

$E_0 = B_0$; (step 1)
for $t = 1; t \leq k; t++$ **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 $i = k+1; i \leq k^2+k; i++$ **do**
  if $B_i[flag] \neq 1$ **then**
    $E_{B_{i,\lfloor (i-1)/k \rfloor}} = B_i$; (step 3)
  end if
end for

Step 1: Step 1 describes transformations of the first $k+1$ blocks of $\{S_0, S_1, S_2, ..., S_k\}$ in B to the first $k+1$ blocks in E. Here, B_0 needs no transformations; thus, we have $E_0 = B_0$. Based on Definition 4, in any sector S_x of B, the first element of each block has the same value as x. To satisfy the property that each block E_t embraces $participant_t$, the first block of $\{S_1, S_2, ..., S_k\}$ of B will be

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

transformed to the E_1 to E_k blocks of E. Consequently, the results of transformations in step 1 are $E_0 = B_0$ and $E_t = B_{tk+1}$,$(1 \leq t \leq k)$. For example, in Fig. 2, the first elements of the first block in S_1, S_2, and S_3 in B are $4, 7$, and 10, respectively, which are marked with a red color. The results of the transformations of step 1 in Fig. 2 are $E_0 = B_0$, $E_1 = B_4$, $E_2 = B_7$ and $E_3 = B_{10}$. It is clearly observed from Fig. 2 that the first $k+1$ blocks of E have the property that $0 \in E_0$, $1 \in E_1$, $2 \in E_2$, and $3 \in E_3$.

Step 2: Transformations of step 2 are based on Lemma 1; in S_0 with $(k+1)(k+1)$ elements, element 0 appears k+1 times in the first column of S_0 and the remaining k^2+k elements $\{1, 2, ..., k^2+k\}$ appear exactly once in S_0 in order. To satisfy the property that each block E_t embraces $participant_t$, the k blocks of $B_1, B_2, ..., B_k$ in B will be transformed to the intended k blocks of E. Note that the index of the k blocks of E is determined by the x^{th} element of the first block of $S_x(x \neq 0)$ in B, which is equal to $E_{t,t}(1 \leq t \leq k)$ of E. The results of the transformations in step 2 are $E_{E_{t,t}} = B_{\lfloor (E_{t,t}-1)/k \rfloor}(1 \leq t \leq k)$. For example, in Fig. 2, the x^{th} element of the first block of $S_x(x \neq 0)$ in B is $4, 8, 11$, respectively, which is marked with a green color. The results of the transformations of step 2 in Fig. 2 are $E_4 = B_1$, $E_8 = B_2$ and $E_{11} = B_3$. It is clearly observed from Fig. 2 that the $E_{t,t}(1 \leq t \leq k)$ blocks of E have the property that $4 \in E_4$, $8 \in E_8$, and $11 \in E_{11}$.

Step 3: The transformations of step 3 are based on Lemma 3; in sector $S_x(x \neq 0)$ with k blocks, the set of the k elements of the x^{th} column is equal to the index set of the k blocks in S_x. In step 3, the remaining $k-1$ blocks of each sector $S_x(x \neq 0)$ in B are transformed to the intended $k \cdot (k-1)$ blocks of E. Note that the index of the $k \cdot (k-1)$ blocks of E is determined by the x^{th} column of the remaining $k-1$ blocks of sector $S_x(x \neq 0)$ in B. Hence, $B_{i,x}(k+1 \leq i \leq k^2+2)$ is used as the index of E. The results of the transformations in step 3 are $E_{B_{i,x}} = B_i(k+1 \leq i \leq k^2+k, B_{i,x} \neq E_{t,t}(1 \leq t \leq k))$, where B_i belongs to the x^{th} sector in B. According to Definition 4, the B_i block in B belongs to the $\lfloor (i-1)/k \rfloor$ sector in B; thus, in step 3 of *Algorithm 2*, x is denoted as $\lfloor (i-1)/k \rfloor$. The k $E_{t,t}(1 \leq t \leq k)$ blocks of $S_x(x \neq 0)$ in B have been transformed in step 2; therefore, the k blocks need no transformations in step 3. For example, in Fig. 2, the x^{th} column of the $k-1$ blocks of sector S_x is $\{5, 6\}, \{9, 7\}, \{12, 10\}$, respectively, which is marked with a white color. The results of the transformations of step 3 in Fig. 2 are $\{E_5 = B_5, E_6 = B_6\}$, $\{E_9 = B_8, E_7 = B_9\}$, $\{E_{12} = B_{11},$ and $E_{10} = B_{12}\}$. It is clearly observed from Fig. 2 that the $B_{i,x}(k+1 \leq i \leq k^2+k, B_{i,x} \neq E_{t,t}(1 \leq t \leq k))$ blocks of E have the property that $5 \in E_5$, $6 \in E_6$, $9 \in E_9$, $7 \in E_7$, $12 \in E_{12}$, and $10 \in E_{10}$.

By *Algorithm 2*, the structure of E is reconstructed, which not only conforms to the properties of a $(v, k+1, 1)$-design but also satisfies the property that each block E_t contains $participant_t$. Hence, the reconstructed E can be used to design the group data sharing model. Based on this model, the key agreement protocol can be processed by v participants and a common conference key can be derived. Moreover, the structure of E should be determined by mathematical descriptions to derive general formulas to compute the common conference key for each participant.

In summary, based on *Algorithm 1*, mathematical descriptions of the structure of B can be deduced first. Then, to derive the mathematical descriptions of the structure of E, the functional relationships of the transformations of B to E should be determined. Based on *Algorithm 2*, the transformations of B to E can be divided into four different cases. In the following four different cases, t denotes the index of the block of E, m implies the m^{th} column of one block of E, and $E_{t,m}$ indicates which participant is contained in the m^{th} column of the t^{th} block in E.

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

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

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

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

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

Case 4: $0 \leq m \leq k, t = B_{i,x}, (t \neq E_{i,i})$

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

Case 1 and *Case 2* correspond to step 1 of *Algorithm 2*, *Case 3* corresponds to step 2 of *Algorithm 2*, and *Case 4* corresponds to step 3 of *Algorithm 2*. In *Case 1*, the structure of E_0 is directly described by B_0, which is $\{0, 1, ..., k\}$. Since in step 1 and step 2 of *Algorithm 2*, the index of B is a function of the index of E, namely, $tk+1$ is a function of t, $\lfloor (E_{t,t}-1)/k \rfloor$ is a function of $E_{t,t}$. The transformations of B to E can be directly determined by the functional relationships between the index of B and the index of E. Thus, the mathematical descriptions in *Case 2* and *Case 3* can easily be obtained by *Algorithm 1*. However, in step 3 of *Algorithm 2*, the index of B is not a function of the index of E, namely, i is not a function of $B_{i,x}$. According to *Algorithm 1*, the index $B_{i,x}(k+1 \leq i \leq k^2+k)$ of E in step 3 of *Algorithm 2* is calculated as Eq. 1, where x and k are known and the index $B_{i,x}(k+1 \leq i \leq k^2+k)$ of E is a function of the index i of B.

$$B_{i,j} = xk + 1 + MOD_k(i - x + (x-1)\lfloor (i-1)/k \rfloor) \quad (1)$$

Let $t = B_{i,x}(k+1 \leq i \leq k^2+k)$, according to [28], the values i in Eq. 1 are $i = k\left\lfloor \frac{t-1}{k} \right\rfloor + r$, where $r = 2, 3, 4, ..., k-1, k$ and the index i of B is a function of the index $t = B_{i,x}$ of E. According to Definition 4, in B, $x = \left\lfloor \frac{t-1}{k} \right\rfloor$. Thus, $i = k\left\lfloor \frac{t-1}{k} \right\rfloor + r$ is equivalent to Eq. 2.

$$i = kx + r \quad (2)$$

Based on Eq. 2, the mathematical descriptions in *Case 4* are derived to describe the structure of the $k-1$ blocks of $S_x(x \neq 0)$ in E. Note that r has $k-1$ different values, which describes the structure of the $(k-1)$ blocks of S_x in E_t.

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*, which is illustrated in TABLE 1. In TABLE 1, the index of E is between 0 and k^2+k, and which participant is contained in the m^{th} column in E_t

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

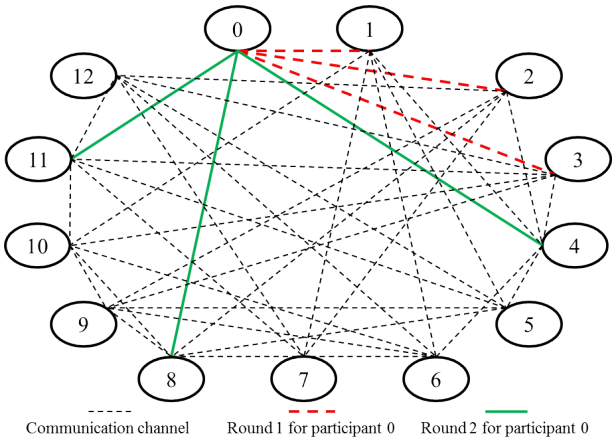


Fig. 3: $(13, 4, 1)$-design group data sharing model.

TABLE 1: The structure of E of a $(v, k+1, 1)$-design

$E_0 = \{0, 1, ..., k\}$
$E_1 = \{1, k+1, 2k+1+MOD_k(0\cdot(2-1)), ..., k^2+1\}$
$E_2 = \{2, k+1, 2k+1+MOD_k(1\cdot(2-1)), ..., k^2+1+ MOD_k(1\cdot(k-1))\}$
$E_3 = \{3, k+1, 2k+1+MOD_k(2\cdot(2-1)), ..., k^2+1+ MOD_k(2\cdot(k-1))\}$
...
$E_k = \{k, k+1, k^2+1+MOD_k((k-1)(2-1)), ..., k^2+1+MOD_k(k-1)^2\}$
$E_{E_{1,1}} = \{0, k+1, k+2, ..., k+k\}$
$E_t = \{1, k+1+MOD_k(r-1), ..., k^2+1+ MOD_k(kx-x-k+r)\}$
$E_{E_{2,2}} = \{0, 2k+1, 2k+2, ..., 2k+k\}$
$E_t = \{2, k+1+MOD_k(r-1), ..., k^2+1+ MOD_k(kx-x-k+r)\}$
...
$E_{E_{k,k}} = \{0, k^2+1, k^2+2, ..., k^2+k\}$
$E_t = \{k, k+1+MOD_k(r-1), ..., k^2+1+ MOD_k(k^2-2k+r)\}$

can be determined by the mathematical descriptions in the four different cases. A concrete example can be found in Appendix, where the structure of E of a $(31, 6, 1)$-design is constructed.

In our protocol, two rounds are required to generate a common conference key. In each round, every participant will receive messages from their intended participants. The group data sharing model can determine which participants are the intended message senders of $participant_i$. The group data sharing model is established as follows. If $j \in E_i$, $participant_j$ is the intended message sender of $participant_i$ in Round 1. If $i \in E_j$, $participant_j$ is the intended message sender of $participant_i$ in Round 2. Based on the group data sharing model, every participant can receive messages from their $2k$ intended message senders after two rounds of the key agreement. For example, in Fig. 3, the $(13, 4, 1)$-design group data sharing model is established by the structure of E of a $(13, 4, 1)$-design. Based on the group data sharing model in Fig. 3, 13 participants are involved, where each participant has 2×3 intended message senders. Taking $participant_0$ into consideration, the intended message senders of $participant_0$ in Round 1 are $1, 2, 3$, whereas the intended message senders of $participant_0$ in Round 2 are $4, 8, 11$. Moreover, for $participant_i$, the messages sending from his $2k$ intended message senders can generate the common conference key for himself.

After the construction of the group data sharing model, the block design-based key agreement protocol is designed for data sharing in cloud computing, which is described in detail in Section 5.

5 A Block Design-Based Key Agreement Protocol

5.1 Initial Phase

In the protocol, a TPA takes responsibility for generating some system parameters and distributing the private key for all participants. In the key generation phase of the protocol, the TPA publishes $\{p, q, G_1, G_2, \mathcal{G}, \hat{e}, P_{pub}, H_1, H_2\}$ but keeps his private key $s \in Z_q^*$ secret. Here, p and q are two prime numbers, and $\mathcal{G}, G_1, G_2$ and $\hat{e}$ are the parameters of the Weil pairing, which are defined in Definition 1. In addition, H_1 and H_2 are two hash functions, which map its arbitrary length to a nonzero point of G_1 and nonzero integer, respectively. In our block design-based key agreement protocol, $participant_i$'s public key and private key are mapped as $H_1(ID_i)$ and $\mathcal{S}_i = sH_1(ID_i)$, respectively. Here, $ID_i \in \{0,1\}^*$ is the identity for $participant_i$. Moreover, to provide authentication, based on the RSA cryptographic algorithm, the TPA selects a public key e_i and a private key d_i for each participant and distributes (e_i, n) to all the participants, where n is the product of two large prime numbers. Subsequently, participant i computes $Y_i = H_2(ID_i)$, $X_i = (Y_i)^{d_i}$ and keeps (d_i, X_i) secret.

5.2 Key Agreement Phase

In the key agreement phase, two rounds are required for generating a common conference key for multiple participants, and the way of message exchanges is with respect to the group data sharing model established by the structure E of the $(v, k+1, 1)$-design.

Round 1: In Round 1, a random number r_i is chosen as a secret key and $\mathcal{M}_i = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)$ is calculated by each participant, which contributes to generating a common conference key among all participants. Then, $Y_i = H_2(ID_i)$, $T_i = X_i \cdot \hat{e}(\mathcal{G}, w_i r_i \mathcal{S}_i)$ and a time stamp t_i are used to support authentication services, where $X_i = Y_i^{d_i}$, $w_i = H_2(\mathcal{M}_i, t_i)$. Subsequently, $participant_i$ receives message $D_j = \{Y_j, (\mathcal{M}_j)^{e_i}, T_j, t_j\}$ from $participant_j$ in the case that $j \in E_i$. In addition, according to the property that each block E_i contains $participant_i$, we have $i \in E_i$. However, $participant_i$ does not have to receive a message from himself. Therefore, $participant_i$ receives message $D_j = \{Y_j, (\mathcal{M}_j)^{e_i}, T_j, t_j\}$ from $participant_j$ in the case that $j \in E_i (j \neq i)$. According to the four mathematical descriptions of the structure of E of a $(v, k+1, 1)$-design, the key agreement phase in Round 1 is divided into four cases.

Case 1:
$Participant_0$ needs to receive messages from $participant_j (1 \leq j \leq k)$.

Case 2:
For $participant_i (i \leq k)$, they need to receive messages from

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

$participant_j(j = mk + 1 + MOD_k(i-1)(m-1), j \neq i)$.

Case 3:

For $participant_i(i = E_{m,m})$, they need to receive messages from $participant_0$ and $participant_j(j = \lfloor (i-1)/k \rfloor k + m, j \neq i)$.

Case 4:

For the remaining $k^2 - k$ participants, they need to receive messages from $participant_{\lfloor (i-1)/k \rfloor}$ and participant $j, (j = mk + 1 + MOD_k(mx - x - m + r), j \neq i)$, where $r = 2, 3, 4, ..., k-1, k$.

After every participant receives k messages contributed to generate a common conference key from their intended message senders, Eq. 3 is calculated by $participant_i$ to decrypt the messages.

$$\mathcal{M}_j = [(\mathcal{M}_j)^{e_i}]^{d_i}, j \in E_i - \{i\} \tag{3}$$

where d_i is the secret key of $participant_i$. To authenticate $participant_j$'s identity, $participant_i$ computes $T_j^{e_j}/\mathcal{M}_j^{w_j^*}$. If the condition of $T_j^{e_j}/\mathcal{M}_j^{w_j^*} = Y_j$ holds, $participant_i$ can authenticate $participant_j$, where $w_j^* = H_2(\mathcal{M}_j, t_j)$. In addition, Eq. 4 is used to derive $C_{i,j}$, which will be used in Round 2 to generate a common conference key for $participant_j$.

$$C_{i,j} = \prod_{x \in E_i - \{j\}} \mathcal{M}_x \tag{4}$$

Round 2: $Participant_i$ receives message $\mathcal{E}_{j,i} = \{Y_j, (C_{j,i})^{e_i}, (M_j)^{e_i}, T_j, t_j\}$ from $participant_j$ in the case that $i \in E_j$, where $C_{j,i}$ is used to generate a common conference key. In fact, every $C_{j,i}$ contributes k messages for $participant_i$ to generate a common conference key. Similar to Round 1, $participant_i$ verifies the equation $T_j^{e_j}/M_j^{w_j^*} = Y_j$ to support authentication services. If the equation holds, $participant_i$ can authenticate $participant_j$'s identity, but not vice versa. Subsequently, for $participant_i$, the common conference key is computed as Eq. 5.

$$\begin{aligned} \mathcal{K} &= \mathcal{M}_i(\prod_{j\ such\ that\ i \in E_j} C_{j,i}) \\ &= \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i) \cdot (\prod_{j\ such\ that\ i \in E_j} C_{j,i}) \\ &= \hat{e}(\mathcal{G}, \sum_{i=0}^{v-1} e_i r_i \mathcal{S}_i) \end{aligned} \tag{5}$$

Theorem 1. *According to the presented block design-based key agreement protocol, a common conference key is derived for multiple participants in the same group.*

Proof. The conference key agreement requires all conferees to obtain messages from the others. In the group data sharing model of the $(v, k+1, 1)$-design, v participants are involved in a group, where $v = k^2 + k + 1$. It is required that each participant receives messages from the remaining $k^2 + k$ participants. Based on the group data sharing model, every participant receives k messages from their intended participants in each round. In Round 1, $participant_i$ receives k secret messages $\mathcal{M}_j$ from $participant_j$ in the case that $j \in E_i (j \neq i)$. In Round 2, $participant_i$ receives k secret messages $C_{j,i}$ from $participant_j$ in the case that $i \in E_j (j \neq i)$. Furthermore, based on Eq. 4, each $C_{j,i}$ contains k secret messages $\mathcal{M}_x$ of k participants. Thus, every participant receives k^2 messages in Round 2. In summary, $participant_i$ receives $k^2 + k$ messages after two rounds of key agreement. According to Definition 3, in a $(v, k+1, 1)$-design, every pair of two elements appears simultaneously in exactly one of the b blocks; here, $v = b$. Therefore, the $k^2 + k$ messages of each participant are not repeated. For $participant_i$, he obtains messages from $(participant_0, ..., participant_{i-1}, participant_{i+1}, ..., participant_{v-1})$ without redundancy, which contribute to generating a common conference key. □

Theorem 2. *In our protocol, $participant_i$ can authenticate their counterparts if the condition of $T_j^{e_j}/\mathcal{M}_j^{w_j^*} = Y_j$ holds.*

Proof. According to Definition 1, we have

$$\begin{aligned} T_j^{e_j}/M_j^{w_j^*} &= \frac{(X_j \cdot \hat{e}(\mathcal{G}, w_j r_j S_j))^{e_j}}{\hat{e}(\mathcal{G}, e_j r_j S_j)^{w_j^*}} \\ &= \frac{X_j^{e_j} \cdot \hat{e}(\mathcal{G}, w_j e_j r_j S_j)}{\hat{e}(\mathcal{G}, w_j^* e_j r_j S_j)} \end{aligned}$$

Here, $w_j = H_2(\mathcal{M}_j, t_j)$ is computed by $participant_j$, while $w_j^* = H_2(\mathcal{M}_j, t_j)$ is computed by $participant_i$. In addition, according to Euler's Theorem, we have $X_j^{e_j} = (Y_j^{d_j})^{e_j} = Y_j$. The equality is held between $T_j^{e_j}/\mathcal{M}_j^{w_j^*}$ and Y_j if $w_j = w_j^*$. Note that the w_j calculated by $participant_j$ and the w_j^* calculated by $participant_i$ are equal only if the message is actually sent from $participant_j$. An adversary that has no access to r_j and $\mathcal{S}_j$ could not derive $\mathcal{M}_j = \hat{e}(\mathcal{G}, e_j r_j S_j)$. Therefore, if the equation $T_j^{e_j}/\mathcal{M}_j^{w_j^*} = Y_j$ holds, $participant_i$ can authenticate that the message is actually transmitted from $participant_j$ in Round 1 and Round 2. □

If all participants follow the protocol, they can form a data sharing group, derive a common conference key and ascertain its correctness. To facilitate understanding, the detailed process for computing the common conference key for multiple participants based on a $(v, k+1, 1)$-design is illustrated in Appendix A. In addition, a concrete example of the protocol can be found in Appendix B, where 31 participants are involved.

5.3 Fault Detection Phase

In practice, we cannot guarantee that all participants in the group are honest. The existence of malicious participants can seriously destroy the conference. In Yi's protocol [7], an attack from malicious participants is called a different key attack. In different key attacks, a malicious participant chooses different sub keys, generates different signatures and broadcasts different messages to different participants such that the signatures of malicious participants are valid and malicious participants can be authenticated by other participants. In addition, the different sub keys make different participants derive different conference keys, which may lead to serious damage of the conference and make the protocol invalid. Therefore, the fault detection phase is added to prevent different key attacks from malicious participants.

The role of the TPA in the fault detection phase is to ensure that each participant only generates a unique sub key and to prevent the conference from being delayed

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

or destroyed by malicious participants. In our protocol, $ID_{TPA} \in \{0,1\}^*$ represents the identity of the TPA. In the initial phase of the protocol, the TPA needs to select one more integer $g \in Z_q^*$ and each $participant_i$ needs to submit $\mathcal{A}_i = \mathcal{M}_i^g$ to the TPA. After all the participants generate a common conference key following the protocol, the TPA broadcasts $\{N, ID_{TPA}, \mathcal{A}_i | 0 \leq i \leq v-1\}$ among all participants, where $N = H_2(ID_1, ID_2, ..., ID_v, ID_1, ID_{TPA}, t)$ is an unique serial number for this conference and $\mathcal{A}_i$ denotes the verified unique sub key of all participants. Then, every participant verifies the authenticity of the common conference key $\mathcal{K}$ by checking whether the equation $\mathcal{K}^g = \prod_{i=0}^{v-1} \mathcal{A}_i$ holds. If the equation does not hold for some participants, some malicious participants are involved in the group and the fault detection phase begins. Otherwise, a common conference key is established among all participants.

In the fault detection phase, $participant_j$, who finds that the above equation $\mathcal{K}^g = \prod_{i=0}^{v-1} \mathcal{A}_i$ does not hold, needs to send a fault report $(N, ID_j, r_j, \mathcal{M}_x, x \in E_j - j)$ to the TPA. The fault report contains the secret key of $participant_j$ and the messages $\mathcal{M}$ he received from the intended participants. Then, the TPA checks whether $\mathcal{M}_j^g = \hat{e}(\mathcal{G}, e_j r_j \mathcal{S}_j)^g = \mathcal{A}_j$ holds. If not, the message that $participant_j$ sends to other participants is different from the message that $participant_j$ submits to the TPA. Thus, $participant_j$ has to resend the fault report in a period of time $\triangle t$. Note that $participant_j$ should be removed from the conference if the failure occurrence of $participant_j$ exceeds a threshold τ or $participant_j$ did not resend the report within $\triangle t$. Here, τ represents the tolerable number of errors. In this case, $participant_j$ is either a malicious participant or undergoes a denial of service attack. Otherwise, the fault detection should be processed among all the remaining participants.

When the fault detection phase is conducted by all the remaining participants, every participant except $participant_j$ should send $(N, ID_i, r_i, \mathcal{M}_x, x \in E_i - i)$ to the TPA. Then, the TPA checks whether $\mathcal{M}_i^g = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)^g = \mathcal{A}_j$ holds. If not, $participant_i$ has to resend the fault report in a period of time $\triangle t$. Similar to $participant_j$, $participant_i$ should be removed from the conference if the failure occurrence of $participant_i$ exceeds a threshold τ or $participant_i$ did not resend the report within $\triangle t$. Otherwise, the TPA checks whether $\mathcal{M}_i = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)$ calculated by $participant_i$ is equal to $\mathcal{M}_i^*$ received from $participant_y (i \in E_y (y \neq i))$. If not, $participant_i$ is a malicious participant. If yes for all the remaining participants, $participant_j$ is a malicious participant. The TPA removes the malicious participant and denial of service participant, and the protocol restarts. After the fault detection phase, an authenticated common conference key is derived among all the honest participants in a group. Following the proof of Theorem 3, the presented protocol can resist different key attacks and support the fault tolerance property.

Theorem 3. *In fault detection phase, an honest participant will not be removed by the TPA and a malicious participant who attempts to delay or destroy the conference will be removed by the TPA.*

Proof. For an honest participant $participant_h$, two situations should be taken into consideration. The first is that $participant_h$ finds $\mathcal{K}^g \neq \prod_{i=0}^{v-1} \mathcal{A}_i$. Subsequently, the fault detection phase begins, and the fault report $(N, ID_h, r_h, \mathcal{M}_x, x \in E_h - h)$ of $participant_h$ is sent to the TPA. Due to the honesty of $participant_h$, there exists $participant_i$ such that either $\mathcal{M}_i^g \neq \mathcal{A}_i$ or $\mathcal{M}_i \neq \mathcal{M}_i^*$. Therefore, $participant_i$ is detected as a malicious participant. The second is that $participant_h$ is asked to submit $(N, ID_h, r_h, \mathcal{M}_x, x \in E_h - h)$ to the TPA. Because of the honesty of $participant_h$, the TPA finds $\mathcal{M}_h^g = \mathcal{A}_h$ and $\mathcal{M}_h = \mathcal{M}_h^*$. In conclusion, an honest participant will never be removed by the TPA.

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

According to Theorem 3, an honest participant will not be removed from the conference, whereas a malicious participant will be detected and removed from the conference. In addition, after some malicious participants are removed from the conference, the common conference key could not be derived because some messages are missing for generating the conference key. Then, the positions of malicious participants should be replaced by volunteers to ensure that the protocol performs well. A volunteer is a participant in a conference who helps real participants complete some calculations and transfer information. Moreover, during the key agreement process, $\mathcal{M}_i$ of the volunteer is set as 1, which can make our protocol perform well. Therefore, the protocol can not only resist different key attacks from malicious participants but also provide the property of fault tolerance.

The proposed protocol can effectively support secure and efficient group data sharing in cloud computing, which is described as follows. In the initial phase, the system parameters are generated by the TPA. Then, the TPA dis-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

tributes the parameters to the clients who want to achieve data sharing in the cloud. In the key agreement phase, in Round 1, each client in the same group selects a secret key r_i and calculates $\mathcal{M}_i = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)$. In addition, $Y_i = H_2(ID_i)$, $T_i = X_i \cdot \hat{e}(\mathcal{G}, w_i r_i \mathcal{S}_i)$ and a time stamp t_i are used to support authentication services, where $X_i = Y_i^{d_i}$ and $w_i = H_2(\mathcal{M}_i, t_i)$. Subsequently, each client receives messages $D_j = \{Y_j, (\mathcal{M}_j)^{e_i}, T_j, t_j\}$ from their intended counterparts based on the structure of the SBIBD. In Round 2, each client receives messages $\mathcal{E}_{j,i} = \{Y_j, (C_{j,i})^{e_i}, (M_j)^{e_i}, T_j, t_j\}$ from their intended counterparts. According to Theorem 1, after the two rounds of information exchange, every client within a group generates a common conference key. In a group, the correctness and validity of the common conference key are guaranteed by the fault tolerance property of the protocol. In addition, the clients in the group can dynamically update the key by restarting the protocol. In Section 6 and Section 7, the presented protocol has already been proven to be secure against both passive attacks and active attacks, and the communication complexity and the computational complexity of our protocol is only $O(v\sqrt{v})$ and $O(vm^2)$, respectively.

6 Security Analysis

The security of our protocol is based on the ECDLP [29] and the BDH assumption [10]. In this section, we prove that our protocol is secure against passive attacks and active attacks.

6.1 Security Against Passive Attacks

In our protocol with v participants, a participant and a volunteer in the protocol are a probabilistic polynomial-time Turing machine, as is an adversary. A passive adversary is the person who attempts to learn information about the conference key by eavesdropping on the multicast channel. Note that an adversary has access to the system parameters $\{\mathcal{G}, P_{PUB}, H_2(ID_i) | 0 \leq i \leq v-1\}$ and the public key (e_i, n) for $participant_i$. In contrast, the secret key d_i for $participant_i$ cannot be deduced since it is hard to solve the integer factorization problem. In addition, the ephemeral key r_i for $participant_i$ is prevented from the adversary due to the ECDLP and the BDH assumption. According to [10], if $X \approx_{poly} Y$, then the presented protocol is secure against passive attacks. $X \approx_{poly} Y$ represents that two tuples of random variables $X = \{\mathcal{G}, P_{PUB}, H_2(ID_i), \hat{e}(\mathcal{G}, \sum e_i r_i \mathcal{S}_i) | 0 \leq i \leq v-1\}$ and $Y = \{\mathcal{G}, P_{PUB}, H_2(ID_i), y | 0 \leq i \leq v-1, y \in Z_q^*\}$ are polynomially indistinguishable, where $y \in Z_q^*$ is a randomly chosen number. More precisely, 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 $Q(l)$ [30]. Here, $l \in Z^+$ is a security parameter in our key agreement protocol, which can determine the size of p defined in Definition 1. All algorithms run in probabilistic polynomial time with l as an input.

Theorem 4. *If the condition of $X_i \approx_{poly} Y_i$ holds for all $participant_i$, then $X \approx_{poly} Y$.*

Proof. Let $X_i = \{\mathcal{G}, P_{PUB}, H_2(ID_i), \hat{e}(\mathcal{G}, \sum e_i r_i \mathcal{S}_i)\}$ and $Y_i = \{\mathcal{G}, P_{PUB}, H_2(ID_i), y_i\}$.

$$
\begin{aligned}
X &= (\mathcal{G}, P_{PUB}, H_2(ID_i), \hat{e}(\mathcal{G}, \sum e_i r_i \mathcal{S}_i) | 0 \leq i \leq v-1) \\
&= (\mathcal{G}, P_{PUB}, H_2(ID_0) ... H_2(ID_{v-1}), \hat{e}(\mathcal{G}, \sum e_0 r_0 \mathcal{S}_0) \cdot \\
&\quad ... \hat{e}(\mathcal{G}, \sum e_{v-1} r_{v-1} \mathcal{S}_{v-1})) \\
&= \prod_{i=0}^{v-1} X_i \\
Y &= (\mathcal{G}, P_{PUB}, H_2(ID_i), y | 0 \leq i \leq v-1, y \in Z_q^*) \\
&= \prod_{i=0}^{v-1} Y_i
\end{aligned}
$$

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, $\prod_{i=0}^{v-1} X_i \approx_{poly} \prod_{i=0}^{v-1} Y_i$. It implies that $X \approx_{poly} Y$. □

6.2 Security Against Active Attacks

In an active attack, an adversary not only learns information about the conference key but also replays, forges and delays the messages. To resist active attacks, desired properties for a practical key agreement protocol typically include the following.

Key comprise impersonation

Our protocol can withstand the key comprise impersonation attack, in which the adversary impersonates a legal conferee (e.g. $participant_j$) to $participant_i$ with the long-term secret key ($\mathcal{S}_i$) of $participant_i$. In our protocol, long-term secret keys of participants are independent of each other with respect to real identities of participants. Therefore, with the long-term secret key ($\mathcal{S}_i$), the adversary still cannot learn any information about long-term secret keys of other participants. In addition, signatures produced by participants are tied with a time stamp. Thus, the adversary cannot be authenticated by replaying the signature of a legal participant later. Moreover, the signature of $participant_i$ is encrypted by his public key e_i. Since no polynomial algorithm has been found for solving the factorization problem, the adversary having no access to d_i cannot forge or decrypt the signature of a legal participant.

Known session key

The known session key prevents the session key held by a fresh participant [11] from being compromised by an adversary, even if the adversary has learned some previous session keys. In the presented protocol, the ephemeral secret key r_i is selected by $participant_i$ in each session randomly, which makes every value of r_i equally likely. Therefore, session keys calculated as $\mathcal{K} = \hat{e}(\mathcal{G}, \sum_{i=0}^{v-1} e_i r_i \mathcal{S}_i)$ are independent in each session such that the adversary cannot learn any information about the session key of a fresh participant. He cannot do any better than a guess.

Perfect forward security

A protocol offers perfect forward security if the compromising of long-term keys ($\mathcal{S}_i$) during the communication among multiple participants cannot result in the compro-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

mising of the previous session key ($\mathcal{K}_{pre}$). In our protocol, the previous session key is computed as

$$\mathcal{K}_{pre} = \hat{e}(\mathcal{G}, \sum_{i=0}^{v-1} e_i r_{i\_pre} \mathcal{S}_i).$$

Even if the long-term keys ($\mathcal{S}_i$) are compromised by an adversary, the adversary who has no access to the previous ephemeral secret key (r_{i_pre}) cannot generate the previous session key. Note that the security of the previous ephemeral key (r_{i_pre}) is based on the ECDLP and the BDH assumption. Therefore, the presented protocol provides perfect forward security.

Different key attacks

In accordance with Theorem 3, in the fault detection phase, a malicious participant who attempts to delay or destruct the conference will be removed from the conference by the TPA. Therefore, the proposed protocol can resist different key attacks.

Key confirmation

If a participant is assured that its counterparts actually have possession of a particular secret key, the protocol provides key confirmation. In our protocol, with respect to the fault detection phase in Section 5, each participant can ensure that its counterparts actually have possession of a common conference key $\mathcal{K}$. Therefore, the presented protocol can provide key confirmation.

Moreover, the presented protocol can resist denial of service attacks. In the fault detection phase, a participant should be removed by the TPA if he did not resend the fault report within $\triangle t$ or the failure occurrence exceeds a threshold τ. Note that the presented protocol is contributory. Unlike the El Gamal one-pass protocol where only one of the parties contributes a fresh exponent, each participant in our protocol equally contributes to the common conference key and guarantees the freshness of the key.

7 Performance Analysis and Evaluation

7.1 Performance Analysis

Generally, the performance of a key agreement protocol consists of communicational and computational efficiency. In each round of our protocol, each participant has to receive k messages from the intended k participants according to a $(v, k+1, 1)$-design of the SBIBD. Then, each participant has to perform some operations such as point multiplications, pairing computations, and so forth. Computational complexity is composed of pairing computations, point multiplications and modular exponentiations, whereas communication complexity is composed of the number of participants and the number of message exchanges.

Let P_i denote the total point multiplications of $participant_i$, M_i represent the total modular exponentiations of $participant_i$ and W_i imply the total Weil pairings computed by $participant_i$. In Round 1, $participant_i$ needs to compute $e_i r_i \mathcal{S}_i$, $w_i r_i \mathcal{S}_i$ and two Weil pairings $\mathcal{M}_i = \hat{e}(\mathcal{G}, e_i r_i \mathcal{S}_i)$, $\hat{e}(\mathcal{G}, w_i r_i \mathcal{S}_i)$. Thus, we have $P_i = 2$, $W_i = 2$. After receiving some messages from $participant_j$, $participant_i$ decrypts $\mathcal{M}_j = [(\mathcal{M}_j)^{e_i}]^{d_i}, j \in E_i - \{i\}$ by his secret key d_i. The number of messages received by $participant_i$ is k. Hence, k modular exponentiations are needed, namely, $M_i = k$. Furthermore, $participant_i$ needs to compute $2k$ modular exponentiations $T_j^{e_j}$ and $\mathcal{M}_j^{w_j^*}$ for the purpose of ensuring his counterparts. In summary, in Round 1, $M_i = 3k$. In Round 2, to obtain $C_{j,i} = (C_{j,i}^{e_i})^{d_i}$, $participant_i$ needs to compute k modular exponentiations. In addition, $3k$ modular exponentiations are required to obtain $\mathcal{M}_j, T_j^{e_j}, \mathcal{M}_j^{w_j^*}$ for the purpose of providing authentication services. Therefore, in Round 2, the number of modular exponentiations is $M_i = 4k$. In terms of communication overhead, every participant needs to receive k messages in each round based on the group data sharing model of a $(v, k+1, 1)$-design. Thus, the number of message exchanges of $participant_i$ is $2k$.

In our protocol, the calculation of the point multiplication, the pairing computation and the modular exponentiation is over the supersingular elliptic curve, which is defined in Definition 1. Thus, the computational complexities of the point multiplications and the pairing computation are $O(m)$ and $O(m^2)$, respectively. Here, m is the extension degree of the finite field F_{p^m}.

Essentially, in the presented protocol with v participants, the total numbers of point multiplications and Weil pairing computations in the protocol are $P = 2v$ and $W = 2v$, respectively. Additionally, the total number of modular exponentiations is $M = 7kv$. The communication complexity and the computational complexity in the protocol are $O(vk)$ and $O(2vm^2 + 2vm)$, respectively. Moreover, in accordance with the basic equation of a BIBD defined in Definition 3, we have $\lambda(v-1) = r(k-1)$. Note that the presented protocol is based on the $(v, k+1, 1)$-design of an SBIBD. Thus, we have $\lambda = 1$ and $r = k$. In this case, $k \approx \sqrt{v}$. Therefore, the communication complexity of our protocol is $O(v\sqrt{v})$, and the computational complexity is $O(vm^2)$. Note that compared to the protocol in paper [9], our protocol is more efficient since we adopt the $(v, k+1, 1)$-design of an SBIBD such that λ can reach its minimum value of one ($\lambda = 1$), where λ is a parameter in the SBIBD. In paper [9], when $\lambda > 1$, k is approximately equal to $\sqrt{\lambda v}$ and the communication complexity of the protocol is $O(v\sqrt{\lambda v})$. In addition, one more modular exponentiation is required for each participant. The detailed comparison results are shown in TABLE 2.

7.2 Performance Evaluation

To study the performance of our scheme, we provide an experimental evaluation of the proposed scheme†. Our experiments are simulated by using C programming language with the pairing-based cryptography (PBC) library and the GUN multiple precision arithmetic (GMP) library on a VMware Workstation machine with Intel Core i5-3210 processors running at 2.50 GHz and 2 G memory, Ubuntu 12.04 X64.

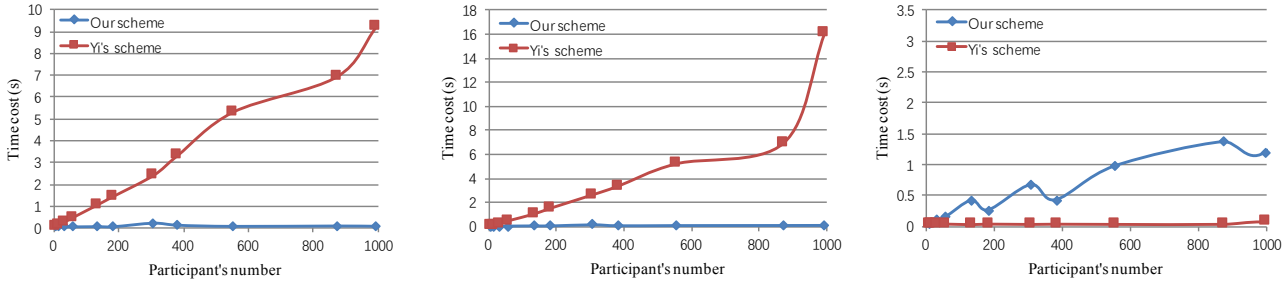
†. Source codes of the simulation have been uploaded to IEEE Xplore + Code Ocean. They are named as
"Efficiency comparison for different phases (v2)" with DOI "10.24433/CO.eea19cea-ca33-4f3c-b641-f46fb7b79253",
"Efficiency comparison for multiple participants (v2)" with DOI "10.24433/CO.a433f2e9-2003-45d1-b519-98bf3aec28dc", and
"Efficiency comparison for different simulation times (v2)" with DOI "10.24433/CO.6b08d728-fc8b-4af7-8f1d-cae7c28af097".

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

TABLE 2: Comparison results

	Yi's protocol	**Shen *et al.*'s protocol**	**Our protocol**
Type of msgs distribution	Broadcast	Multicast	Multicast
Type of communication model	Centralized	Decentralized	Decentralized
The number of participants	n	7	n
No. of Weil pairing computation per $participant_i$	$2(n-1)$	2	2
No. of point multiplication per $participant_i$	6	2	2
No. of modular exponentiation per $participant_i$	$2n$	$7n\sqrt{\lambda n}$	$7n\sqrt{n}$
Total computation cost	$2n^2(W_i)+7n(P_i)+2n(M_i)$	$2n(W_i)+2n(P_i)+(7n\sqrt{\lambda n}+n)(M_i)$	$2n(W_i)+2n(P_i)+7n\sqrt{n}(M_i)$
Communication complexity	$O(n^2)$	$O(n\sqrt{\lambda n})$	$O(n\sqrt{n})$
Computational complexity	$O(n^2m^2)$	$O(nm^2)$	$O(nm^2)$

*n: Participant's number, m: Extension degree of the finite field F_{p^m},P_i: Point multiplications of $participant_i$, M_i: Modular exponentiations of $participant_i$, W_i: Weil pairings computed by $participant_i$, λ: Parameter in the SBIBD.



(a) Initial phase (b) Key agreement phase (c) Authentication phase

Fig. 4: Efficiency comparison for different phases.

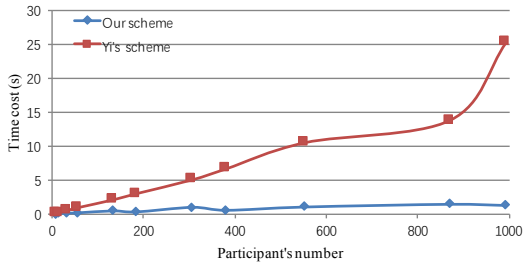


Fig. 5: Efficiency comparison for multiple participants.

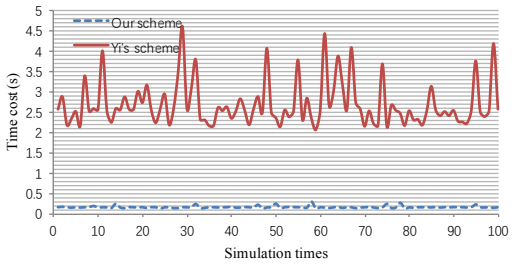


Fig. 6: Efficiency comparison for different simulation times.

The simulation consists of two parts. In the first part, we present a comparative simulation analysis between Yi's scheme [7] and our scheme with respect to the time cost for each participant in different phases, which is illustrated in Fig. 4. It can be seen that the time cost increases with the number of participants. On the one hand, simulation results in Fig. 4(a) and Fig. 4(b) indicate that our scheme is much more efficient than Yi's scheme in both initial phase and key agreement phase. On the other hand, in Fig. 4(c), the time cost of our scheme is slightly higher than that of Yi's scheme. The reason is that, for each participant, 4 point multiplications are required in Yi's scheme, while k modular exponentiations are required in our scheme during the authentication phase. However, we argue that, in terms of the total computational cost for each participant, our scheme is much more efficient than Yi's scheme, which is illustrated in Fig. 5.

In the second part, we focus on analyzing the total computational cost for each participant of Yi's scheme and our scheme with respect to different participants and different simulation times. It is clearly seen from Fig. 5 that our scheme is superior to Yi's scheme. Note that the computational cost of Yi's scheme continuously increases with the growth of the participant's number n, while the computational cost of our scheme increases slightly with a prime number k (Here, $n=k^2+k+1$). It is concluded that our scheme is much more efficient than Yi's scheme, which makes our scheme more practical for key agreement in the cloud environment. In addition, in Fig. 6, we present the efficiency comparison of Yi's scheme and our scheme with different simulation times, where the participants number is fixed as 133. Note that taking advantage of the SBIBD in our scheme, $k=11$ when the participant's number is 133. Firstly, in the initial phase, Yi's scheme requires to compute

2 point multiplications and 132 weil pairings, while our scheme only needs 2 point multiplications, 2 weil pairings and 11 modular exponentiations. Secondly, in the key agreement phase, Yi's scheme requires 132 weil pairings, while our scheme only needs 33 modular exponentiations. Finally, in the authentication phase, Yi's scheme requires 4 point multiplications, while our scheme needs 33 modular exponentiations. Through the simulation, we can conclude that the time cost of our scheme is much smaller than that of Yi's scheme with different simulation times. In addition, it is easily observed that the performance of our scheme is more stable than Yi's scheme.

8 Conclusion

As a development in the technology of the Internet and cryptography, group data sharing in cloud computing has opened up a new area of usefulness to computer networks. With the help of the conference key agreement protocol, the security and efficiency of group data sharing in cloud computing can be greatly improved. Specifically, the outsourced data of the data owners encrypted by the common conference key are protected from the attacks of adversaries. Compared with conference key distribution, the conference key agreement has qualities of higher safety and reliability. However, the conference key agreement asks for a large amount of information interaction in the system and more computational cost. To combat the problems in the conference key agreement, the SBIBD is employed in the protocol design.

In this paper, we present a novel block design-based key agreement protocol that supports group data sharing in cloud computing. Due to the definition and the mathematical descriptions of the structure of a $(v, k+1, 1)$-design, multiple participants can be involved in the protocol and general formulas of the common conference key for $participant_i$ are derived. Moreover, the introduction of volunteers enables the presented protocol to support the fault tolerance property, thereby making the protocol more practical and secure. In our future work, we would like to extend our protocol to provide more properties (e.g., anonymity, traceability, and so on) to make it appliable for a variety of environments.

Acknowledgments

The authors would like to thank the editors and anonymous reviewers for their constructive feedback and insightful suggestions that helped to significantly improve the quality of this work.

This work is supported by the National Science Foundation of China under Grant No. 61672295, No. 61373169, No. 61572379, No. 61501333 and No. U1405254, the State Key Laboratory of Information Security under Grant No. 2017-MS-10, the 2015 Project of six personnel in Jiangsu Province under Grant No. R2015L06, the CICAEET fund, and the PAPD fund.

References

[1] L. Zhou, V. Varadharajan, and M. Hitchens, "Cryptographic role-based access control for secure cloud data storage systems," *Information Forensics and Security IEEE Transactions on*, vol. 10, no. 11, pp. 2381–2395, 2015.

[2] F. Chen, T. Xiang, Y. Yang, and S. S. M. Chow, "Secure cloud storage meets with secure network coding," in *IEEE INFOCOM*, 2014, pp. 673–681.

[3] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Systems Journal*, pp. 1–10, 2015.

[4] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[5] J. Shen, H. Tan, S. Moh, I. Chung, and J. Wang, "An efficient rfid authentication protocol providing strong privacy and security," *Journal of Internet Technology*, vol. 17, no. 3, p. 2, 2016.

[6] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," *Designs Codes and Cryptography*, vol. 28, no. 2, pp. 119–134, 2010.

[7] X. Yi, "Identity-based fault-tolerant conference key agreement," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 3, pp. 170–178, 2004.

[8] R. Barua, R. Dutta, and P. Sarkar, "Extending joux's protocol to multi party key agreement (extended abstract)." *Lecture Notes in Computer Science*, vol. 2003, pp. 205–217, 2003.

[9] J. Shen, S. Moh, and I. Chung, "Identity-based key agreement protocol employing a symmetric balanced incomplete block design," *Journal of Communications and Networks*, vol. 14, no. 6, pp. 682–691, 2012.

[10] B. Dan and M. Franklin, "Identity-based encryption from the weil pairing," *Siam Journal on Computing*, vol. 32, no. 3, pp. 213–229, 2003.

[11] S. Blakewilson, D. Johnson, and A. Menezes, "Key agreement protocols and their security analysis," in *IMA International Conference on Cryptography and Coding*, 1997, pp. 30–45.

[12] I. Chung and Y. Bae, "The design of an efficient load balancing algorithm employing block design," *Journal of Applied Mathematics and Computing*, vol. 14, no. 1, pp. 343–351, 2004.

[13] O. Lee, S. Yoo, B. Park, and I. Chung, "The design and analysis of an efficient load balancing algorithm employing the symmetric balanced incomplete block design." *Information Sciences*, vol. 176, no. 15, pp. 2148–2160, 2006.

[14] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *Journal of Computer Security*, vol. 19, no. 5, pp. 79–88, 2011.

[15] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014.

[16] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1–1, 2015.

[17] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1–1, 2016.

[18] S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, "Encryption policies for regulating access to outsourced data," *Acm Transactions on Database Systems*, vol. 35, no. 2, pp. 78–78, 2010.

[19] H. Guo, Z. Li, Y. Mu, and X. Zhang, "Cryptanalysis of simple three-party key exchange protocol," *Computers and Security*, vol. 27, no. 1-2, pp. 16–21, 2008.

[20] Z. Tan, "An enhanced three-party authentication key exchange protocol for mobile commerce environments," *Journal of Communications*, vol. 5, no. 5, pp. 436–443, 2010.

[21] Y. M. Tseng, "An efficient two-party identity-based key exchange protocol." *Informatica*, vol. 18, no. 1, pp. 125–136, 2007.

[22] A. Shamir, "Identity-based cryptosystems and signature schemes," *Lecture Notes in Computer Science*, vol. 21, no. 2, pp. 47–53, 1985.

[23] E. Bresson, O. Chevassut, D. Pointcheval, and J. J. Quisquater, "Provably authenticated group diffie-hellman key exchange," *Acm Transactions on Information and System Security*, vol. 10, no. 3, pp. 89–92, 2001.

[24] D. R. Stinson, *Combinatorial designs: constructions and analysis*. Springer Science and Business Media, 2007.

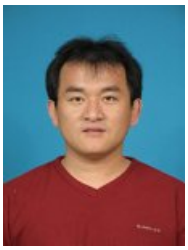
[25] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Transactions on Information Forensics and Security*, 2017, doi: 10.1109/TIFS.2017.2705620.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2017.2725953, IEEE Transactions on Dependable and Secure Computing

[26] B. Lamacchia, K. Lauter, and A. Mityagin, "Stronger security of authenticated key exchange," in *International Conference on Provable Security*, 2007, pp. 1–16.
[27] O. Hasan, L. Brunie, E. Bertino, and N. Shang, "A decentralized privacy preserving reputation protocol for the malicious adversarial model," *Information Forensics and Security IEEE Transactions on*, vol. 8, no. 6, pp. 949–962, 2013.
[28] L.-K. Hua, *Introduction to number theory*. Springer Science and Business Media, 2012.
[29] W. Stallings, "Cryptography and network security: Principles and practice," *International Annals of Criminology*, vol. 46, no. 4, pp. 121–136, 2008.
[30] M. Steiner, G. Tsudik, and M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 8, pp. 769–780, 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 late 2012, he has been a professor at Nanjing University of Information Science and Technology, Nanjing, China. His research interests include public key cryptography, secure data sharing and data auditing in cloud.

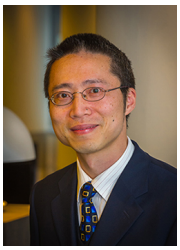
Tianqi Zhou received the B.E. degree from Nanjing University of Information Science and Technology, Nanjing, China, in 2016. She is currently a postgraduate with the School of Nanjing University of Information Science and Technology, Nanjing, China. Her research interests include computer and network security, security systems and cryptography.



Debiao He received the Ph.D. degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009. Currently, he is a professor with the State Key Laboratory of Software Engineering, Computer School, Wuhan University. His research interests include cryptography and information security, in particular, cryptographic protocols.

Yuexin Zhang received his B.S. degree from the Department of Physics and Electronic Information Engineering, Inner Mongolia Normal University, China, in 2010. and the M.S. degree from the School of Mathematics and Computer Science, Fujian Normal University, China, in 2013. He is currently pursuing the Ph.D. degree in Computer Science at Deakin University, Melbourne, Australia. His research focuses on network security.

Xingming Sun received his B.S. in mathematics from Hunan Normal University, China, in 1984, M.E. in computing science from Dalian University of Science and Technology, China, in 1988, and Ph.D. in computing science from Fudan University, China, in 2001. He is currently a professor in School of Computer and Software, Nanjing University of Information Science and Technology, China. His research interests include network and information security, digital watermarking, digital forensic, database security, and natural language processing.



Yang Xiang received the Ph.D. degree in computer science from Deakin University, Australia. He is currently a Dean at the Digital Research & Innovation Capability Platform, Swinburne University of Technology. He is the director of the Network Security and Computing Lab (NSCLab). His research interests include network and system security, distributed systems, and networking. In particular, he is currently leading his team developing active defense systems against large-scale distributed network attacks. He is the chief investigator of several projects in network and system security, funded by the Australian Research Council (ARC). He has published more than 130 research papers in many international journals and conferences, such as IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, IEEE Transactions on Information Security and Forensics, and IEEE Journal on Selected Areas in Communications. Two of his papers were selected as the featured articles in the April 2009 and the July 2013 issues of IEEE Transactions on Parallel and Distributed Systems. He has published two books, Software Similarity and Classification (Springer) and Dynamic and Advanced Data Mining for Progressing Technological Development (IGI-Global). He has served as the program/general chair for many international conferences such as ICA3PP 12/11, IEEE/IFIP EUC 11, IEEE TrustCom 13/11, IEEE HPCC 10/09, IEEE ICPADS 08, and NSS 11/10/09/08/07. He has been the PC member for more than 60 international conferences in distributed systems, networking, and security. He serves as the associate editor of IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Security and Communication Networks (Wiley), and the editor of Journal of Network and Computer Applications. He is the coordinator, Asia for IEEE Computer Society Technical Committee on Distributed Processing (TCDP). He is a senior member of the IEEE.