Provably Secure Group Key Management Approach Based upon Hyper-Sphere

Shaohua Tang, *Member, IEEE*, Lingling Xu, Niu Liu, Xinyi Huang, Jintai Ding, and Zhiming Yang

Abstract—Secure group communication systems have become increasingly important for many emerging network applications. An efficient and robust group key management approach is indispensable to a secure group communication system. Motivated by the theory of hyper-sphere, this paper presents a new group key management approach with a group controller (GC). In our new design, a hyper-sphere is constructed for a group and each member in the group corresponds to a point on the hyper-sphere, which is called the member's private point. The GC computes the central point of the hyper-sphere, intuitively, whose "distance" from each member's private point is identical. The central point is published such that each member can compute a common group key, using a function by taking each member's private point and the central point of the hyper-sphere as the input. This approach is provably secure under the pseudo-random function (PRF) assumption. Compared with other similar schemes, by both theoretical analysis and experiments, our scheme (1) has significantly reduced memory and computation load for each group member; (2) can efficiently deal with massive membership change with only two re-keying messages, i.e., the central point of the hyper-sphere and a random number; and (3) is efficient and very scalable for large-size groups.

Index Terms—Group communication, key management, hyper-sphere, pseudo-random function (PRF), provable security

1 Introduction

With the rapid development of Internet technology and the popularization of multicast, group-oriented applications, such as video conference, network games, and video on demand, etc., are playing important roles. How to protect the communication security of these applications are becoming more and more significant. Generally speaking, a secure group communication system should not only provide data confidentiality, user authentication, and information integrity, but also accommodate perfect scalability. Without any doubt, a secure, efficient, and robust group key management approach is essential to a secure group communication system.

Our Contributions. This paper presents a secure group key management approach based on the properties of hyper-sphere. In mathematics, a hyper-sphere is a generalization of the surface of an ordinary sphere to arbitrary dimension. The distance from any point on the hyper-sphere to the central point of the hyper-sphere is identical. Inspired by this principle, a secure group key management scheme is designed. The most significant advantages of the proposed approach are the reduction of user storage, user computation, and the amount of update information due to re-keying. The group key is updated periodically to protect its secrecy. Each key is completely independent from any previously used and future keys. A formal security proof for our scheme is given under the assumption of pseudo-random functions (PRF).

Organization. The remainder of this paper is organized as follows. A brief survey of relevant schemes on secure group key management is described in Section 2. Preliminaries and the security model are given in Section 3. Our proposed secure group key management approach is presented in Section 4. Formal analysis of the security and performance are given in Section 5. Comparisons with other schemes are presented in Section 6. Finally, Section 7 summarizes the major contributions of this paper.

2 A Brief Survey of Related Work

There are various approaches on key management for secure group communication. Rafaeli and Hutchison [1] presented a comprehensive survey on this area. Existing schemes can be divided into three different categories: centralized, distributed and decentralized schemes.

In a centralized system, there is an entity Group Controller (GC) managing the whole group [1]. Some typical schemes in this category include Group Key Management Protocol (GKMP) [2], Secure Lock (SL) [3], Logical Key Hierarchy (LKH) [4], etc. The Group Key Management Protocol [2] is a direct extension from unicast to multicast communication. It is assumed that there exists a secure channel between the GC and every group member. Initially, the GC selects a group key K_0 and distributes this key to all group members via the secure channel. Whenever a member joins in the group, the GC selects a new group key K_N, encrypts the new group key with the old group key yielding $K' = E_{K_0}(K_N)$, and then broadcasts K' to the group members. Moreover, the GC sends K_N to the joining member via

- *S. Tang, L. Xu, N. Liu, and Z. Yang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong 510640, P.R. China. E-mail: shtang@ieee.org, csllxu@scut.edu.cn, niuliu83@gmail.com, yzm52703@163.com.*
- *X. Huang is with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, Fujian, P.R. China. E-mail: xyhuang@fjnu.edu.cn.*
- *J. Ding is with the Department of Mathematical Sciences, University of Cincinnati, OH. E-mail: jintai.ding@uc.edu.*

Manuscript received 14 Apr. 2013; revised 29 Sept. 2013; accepted 30 Dec. 2013. Date of publication 15 Jan. 2014; date of current version 14 Nov. 2014. Recommended for acceptance by D. Xuan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2013.2297917

the secure channel between the GC and the new member. Obviously, the solution is not scalable [1]. The Secure Lock scheme [3] takes advantage of Chinese Remainder Theorem (CRT) to construct a secure lock to combine all re-keying messages into a single message while the group key is updated. However, CRT is a time-consuming operation. As mentioned in [3], the SL scheme is efficient only when the number of users in a group is small, since the time to compute the lock and the length of the lock (hence the transmission time) is proportional to the number of users. The Logical Key Hierarchy scheme [4] adopts tree structure to organize keys. The GC maintains a virtual tree, and the nodes in the tree are assigned keys. The key held by the root of the tree is the group key. The internal nodes of the tree hold key encryption keys (KEK). Keys at leaf nodes are possessed by individual members. Every member is assigned the keys along the path from its leaf to the root. When a member joins or leaves the group, its parent node's KEK and all KEKs held by nodes in the path to the root should be updated. The number of keys which need to be changed for a joining or leaving is $\mathcal{O}(\log_2 n)$ and the number of encryptions is $\mathcal{O}(2 \times \log_2 n)$. If there are a great deal of members need to join or leave the group, then the re-keying overhead will increase proportionally to the number of members changed. There are some other schemes that adopt tree structures, for example, One-way Function Tree (OFT) [5], One-way Function Chain Tree (OFCT) [6], Hierarchical α-ary Tree with Clustering [7], Efficient Large-Group Key [8], etc.

In distributed key management schemes, there is no explicit GC and the key generation can be either contributory or done by one of the members [1]. Some typical schemes include: Burmester and Desmedt Protocol [9], Group Diffie-Hellman key exchange [10], Octopus Protocol [11], Conference Key Agreement [12], Distributed Logical Key Hierarchy [13], Distributed One-way Function Tree [14], Diffie-Hellman Logical Key Hierarchy [15], [16], Distributed Flat Table [17], etc. Recent references paid more attentions to contributory and collaborative group key agreement [18], [19], [20], [21], [22], [23], etc. Recently, the concepts of asymmetric group key agreement (ASGKA) and contributory broadcast encryption (CBE) were proposed [24], [25], [26]. An asymmetric group key agreement protocol [24] lets the group members negotiate a shared encryption key instead of a common secret key. The encryption key is accessible to attackers and corresponds to different decryption keys, each of which is only computable by one group member. A contributory broadcast encryption [25] enables a group of members negotiate a common public encryption key while each member holds a decryption key. A hybrid key management paradigm was proposed by Wu et al. [26] to integrate group key agreement and traditional broadcast encryption, which enables a remote sender broadcast securely to any intended subgroup chosen in an ad hoc way on seeing the public keys of the members.

In the decentralized architectures, the large group is split into small subgroups. Different controllers are used to manage each subgroup [1]. Some typical schemes include: Scalable Multicast Key Distribution [27], Iolus [28], Dual-Encryption Protocol [29], MARKS [30], Cipher Sequences [31], Kronos [32], Intra-Domain Group Key Management [33], Hydra [34], etc.

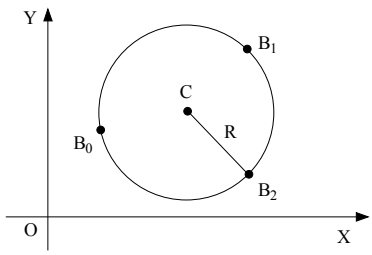


Fig. 1. A one-sphere or a circle in a plane.

Secure group key management approaches can be applied to a lot of application areas. For example: wireless/mobile network [35], [36], [37], [38], [39], [40], wireless sensor network [41], storage area networks [42], etc.

3 Preliminaries

In this section, we first briefly describe the concept of hyper-sphere, and present some syntax used throughout this paper. Then we define Pseudo-Random Function, and describe the security model under which we prove the security of our group key management protocol.

3.1 N-Dimensional Hyper-Sphere

For any natural number $N \in \mathbb{N}$, an N-dimensional hyper-sphere or an N-sphere is a generalization of the surface of an ordinary sphere to arbitrary dimension. In particular, a zero-sphere is a pair of points on a line, a one-sphere illustrated in Fig. 1 is a circle in a plane, and a two-sphere is an ordinary sphere in three-dimensional space. Spheres of dimension $N > 2$ are sometimes called hyper-spheres.

3.1.1 Hyper-Sphere in Euclidean Space

In mathematics, an N-sphere of radius $r \in \mathbb{R}$ with a central point $\boldsymbol{C} = (c_0, c_1, \ldots, c_N) \in \mathbb{R}^{N+1}$ is defined as the set of points in $(N+1)$-dimensional Euclidean space which are at distance r from the central point $\boldsymbol{C}$. Any point $\boldsymbol{X} = (x_0, x_1, \ldots, x_N) \in \mathbb{R}^{N+1}$ on the hyper-sphere can be represented by the equation

$$(x_0 - c_0)^2 + (x_1 - c_1)^2 + \cdots + (x_N - c_N)^2 = r^2. \quad (1)$$

Any given $N+2$ points $\boldsymbol{A}_i = (a_{i,0}, a_{i,1}, \ldots, a_{i,N}) \in \mathbb{R}^{N+1}$, where $i = 0, 1, \ldots, N+1$, can uniquely determine a hyper-sphere as long as certain conditions are satisfied, which will be presented at the end of this subsection. By applying the coordinates of the points $\boldsymbol{A}_0, \boldsymbol{A}_1, \ldots, \boldsymbol{A}_{N+1}$ to (1), we can obtain a system of $N+2$ equations

$$\begin{cases} (a_{0,0} - c_0)^2 + (a_{0,1} - c_1)^2 + \cdots + (a_{0,N} - c_N)^2 = r^2, \\ (a_{1,0} - c_0)^2 + (a_{1,1} - c_1)^2 + \cdots + (a_{1,N} - c_N)^2 = r^2, \\ \qquad\qquad \cdots\cdots \\ (a_{N+1,0} - c_0)^2 + (a_{N+1,1} - c_1)^2 + \cdots + (a_{N+1,N} - c_N)^2 = r^2. \end{cases} \quad (2)$$

By subtracting the jth equation from the $(j+1)$-th equation, where $j = 1, 2, \ldots, N+1$, we can obtain a system of linear equations with $N+1$ unknowns $c_0, c_1, \ldots, c_N$:

$$\begin{cases} 2(a_{0,0} - a_{1,0})c_0 + \cdots + 2(a_{0,N} - a_{1,N})c_N = \sum_{j=0}^{N} \left(a_{0,j}^2 - a_{1,j}^2\right), \\ \cdots\cdots \\ 2(a_{N,0} - a_{N+1,0})c_0 + \cdots + 2(a_{N,N} - a_{N+1,N})c_N \\ \qquad = \sum_{j=0}^{N} \left(a_{N,j}^2 - a_{N+1,j}^2\right). \end{cases} \tag{3}$$

If and only if the determinant of the coefficients in (3) is non-zero, this system of linear equations can have a unique solution $(c_0, c_1, \ldots, c_N)$. By applying the values of $c_0, c_1, \ldots, c_N$ to one of the equations in (2), we can obtain r^2.

3.1.2 Hyper-Sphere over Finite Field

We can extend the concept of Hyper-sphere to finite fields. For simplicity, the Galois field $GF(p)$ is adopted as the ground field, where p is a large prime number. However, the results can be easily extended to other forms of finite fields. For any given positive integer N, and vector $\mathbf{C} = (c_0, c_1, \ldots, c_N) \in GF(p)^{N+1}$, we define function

$$\boldsymbol{R} : GF(p)^{N+1} \rightarrow GF(p)$$

as

$$\boldsymbol{R}(\boldsymbol{X}) \equiv \|\boldsymbol{X} - \boldsymbol{C}\|^2 \pmod{p}, \tag{4}$$

where $\boldsymbol{X} = (x_0, x_1, \ldots, x_N) \in GF(p)^{N+1}$, and

$$\|\boldsymbol{X} - \boldsymbol{C}\|^2 \equiv (x_0 - c_0)^2 + (x_1 - c_1)^2 + \cdots + (x_N - c_N)^2 (\mathrm{mod}\, p).$$

Definition 1 (Hyper-Sphere over Finite Field). *For any given positive integer N, vector $\boldsymbol{C} = (c_0, c_1, \ldots, c_N) \in GF(p)^{N+1}$, and $\overline{R} \in GF(p)$, the **Hyper-Sphere** determined by $\boldsymbol{C}$ and $\overline{R}$ over the finite field $GF(p)$ is defined by*

$$\boldsymbol{R}(\boldsymbol{X}) \equiv \overline{R} \pmod{p}, \tag{5}$$

or

$$(x_0 - c_0)^2 + (x_1 - c_1)^2 + \cdots + (x_N - c_N)^2 \equiv \overline{R} \,(\mathrm{mod}\, p), \tag{6}$$

*where $\boldsymbol{C} = (c_0, c_1, \ldots, c_N)$ is called the **Central Point** of the hyper-sphere, and $\boldsymbol{X} = (x_0, x_1, \ldots, x_N) \in GF(p)^{N+1}$ is called a **Point** on the hyper-sphere.*

Specifically, a *Sphere over Finite Field* is a trivial case of hyper-sphere over finite field if $N = 2$ in (6), and a *Circle over Finite Field* is another special case of hyper-sphere over finite field if $N = 1$ in (6).

Given $N, \boldsymbol{C}$ and $\overline{R}$, we can find at least p^{N-1} different points on the hyper-sphere determined by $\boldsymbol{C}$ and $\overline{R}$. This fact is proven by Theorem 1 in the supplementary file, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.2297917, and a concrete procedure to find a point on the hyper-sphere is given by Algorithm 1 in the supplementary file, available in the online supplemental material.

3.2 Syntax

If $\kappa \in \mathbb{N}$, then 1^κ is the string consisting of κ ones. If $\mathtt{A}$ is a randomized algorithm, then $y \leftarrow \mathtt{A}(x)$ denotes the assignment to y of the output of $\mathtt{A}$ on input x when running with fresh random coins. We use the notation $u \leftarrow_R S$ to denote that u is chosen randomly from S, where the symbol R stands for "randomly". Unless noted, all algorithms are probabilistic polynomial-time (PPT) and we implicitly assume that they take an extra parameter 1^κ in their input, where κ is a security parameter. A function $\nu(\cdot) : \mathbb{N} \rightarrow [0, 1)$ is negligible[1] if for every polynomial $p(\cdot)$ there exists a $\kappa_c \in \mathbb{N}$ such that for all integers $\kappa > \kappa_c$ it holds that $\nu(\kappa) < \frac{1}{p(\kappa)}$.

3.3 Pseudo-Random Function

Let κ be a security parameter, $F^\kappa : Keys(F^\kappa) \times D \rightarrow R$ be a family of functions with input length $l_{in}(\kappa)$, output length $l_{out}(\kappa)$, and key length $l_{key}(\kappa)$, where $Keys(F^\kappa)$ stands for the key space of F^κ, D and R represent the input space and output space respectively. Let $Func : D \rightarrow R$ be the set of all functions from D to R. We adopt some expressions of pseudo-random function in [43], [44], and its definition is given as follows.

Definition 2 (Pseudo-Random Function). *We say that F^κ is a pseudo-random function (or PRF for short) if $F_K(x)$ is polynomial-time computable in κ, where $F_K \in F^\kappa$, $K \in Keys(F^\kappa)$ and $x \in D$, and for every PPT distinguisher $\mathcal{D}$ who is given access to an oracle for a function $g : D \rightarrow R$, where g can be chosen at random from $Func$ or is chosen at random from F^κ, the advantage $\mathbf{Adv}^{\mathbf{PRF}}_{F^\kappa,\mathcal{D}}$ is negligible in κ. $\mathbf{Adv}^{\mathbf{PRF}}_{F^\kappa,\mathcal{D}}$ is defined by indistinguishability of the following two experiments,*

Experiment $\mathbf{EXP}^{\mathbf{prf-1}}(\mathcal{D})$
$K \leftarrow_R Keys(F^\kappa)$
$b \leftarrow \mathcal{D}(F_K)$
return b

Experiment $\mathbf{EXP}^{\mathbf{prf-0}}(\mathcal{D})$
$g \leftarrow_R Func$
$b \leftarrow \mathcal{D}(g)$
return b

The advantage $\mathbf{Adv}^{\mathbf{PRF}}_{F^\kappa,\mathcal{D}}$ is defined as

$$\mathbf{Adv}^{\mathbf{PRF}}_{F^\kappa,\mathcal{D}} = \left| \mathbf{Prob}[\mathbf{EXP}^{\mathbf{prf-1}}(\mathcal{D}) = 1] - \mathbf{Prob}[\mathbf{EXP}^{\mathbf{prf-0}}(\mathcal{D}) = 1] \right|.$$

PRFAssumption. There exists no (t, ϵ)-PRF distinguisher in κ. In other words, for every probabilistic, polynomial-time, 0/1-valued distinguisher $\mathcal{D}$, who runs in time t, $\mathbf{Adv}^{\mathbf{PRF}}_{F^\kappa,\mathcal{D}} \leq \epsilon$ for any sufficiently small $\epsilon > 0$.

In our construction of group key management protocol, we specify a family of pseudo-random functions

1. The function $\nu(\kappa) = \frac{1}{\lambda^\kappa}$ is an instance of negligible functions, where λ is an integral constant and $\lambda > 1$.

$$F^{\kappa} : GF(p) \times GF(p) \rightarrow GF(p),$$

i.e.,

$$F^{\kappa} = \{f_a(\cdot) \,|\, a \in GF(p)\}.$$

The cardinalities of F^{κ} and $Func$ are p and p^p respectively.

3.4 Security Model

Usually, a group key management scheme includes some phases like initialization, adding members, removing members, massively adding and removing members, and periodically update. Communications at all phases in this paper are assumed to be authenticated, since we emphasize on the mechanism of group key management. An authentication channel can be achieved by incorporating a standard digital signature scheme into the protocol as mentioned in Section 5.1.

The objective of the adversary is to derive or guess the correct group key of the current session by using all information that it possesses and all capabilities that it owns. The adversary $\mathcal{A}$ is assumed to be a probabilistic polynomial-time adversary. Its capabilities usually include corrupting the deleted members, eavesdropping on communications, and modifying, inserting, deleting and replaying of transmitted messages. Since we assume the communications are authenticated, the adversary will fail to launch attacks by modifying, inserting, deleting, and replaying of messages. Then, the information that the adversary can possess includes all transmitted messages between the group controller GC and each individual member, and among group members, and expired group keys of previous sessions.

Our formal adversarial model described below is similar to the security model of Atallah et al. [45] and Dutta and Barua [18], where the adversary's capability is modeled by querying oracles to simulate real attacks.

Let $\mathcal{P} = \{U_1, U_2, \ldots, U_N\}$ be a set of N users or group members. At any point of time, any subset of $\mathcal{P}$ may decide to establish a session key via the group controller GC who is a trusted third party. We identify the execution of protocols for initial group key establishment, adding member, removing member, and periodically re-keying as different sessions. The adversarial model allows each user an unlimited number of instances of joining or leaving or re-keying. We assume that an adversary never participates as a user in the protocol. This adversarial model allows concurrent execution of the protocol. Let $G = \{U_1, \ldots, U_n\}$, and G is any non-empty subsets of $\mathcal{P}$.

We will require the following notations:

- $\mathbf{LS}_{GC}$ Long-term secret kept by the group controller GC.
- $\mathbf{LS}_U$ Long-term secret of user U.
- Π_U^i The ith instance of user U.
- $\mathbf{sk}_U^i$ Session key after execution of the protocol by Π_U^i.
- $\mathbf{sid}_U^i$ Session identity for instance Π_U^i. We set $\mathbf{sid}_U^i = \{(U_1, i_1), \ldots, (U_n, i_n)\}$ such that users $U_1, \ldots, U_n$ wish to agree upon a common key in a session using unused instances $\Pi_{U_1}^{i_1}, \ldots, \Pi_{U_n}^{i_n}$.
- $\mathbf{pid}_U^i$ Partner identity for instance Π_U^i, defined by $\mathbf{pid}_U^i = \{U_1, \ldots, U_n\}$, such that $(U_j, i_j) \in \mathbf{sid}_U^i$ for all $1 \leq j \leq n$, where i_j comes from $\mathbf{sid}_U^i$ defined above.
- $\mathbf{acc}_U^i$ 0/1-valued variable which is set to be 1 by Π_U^i upon normal termination of the session; and is set to be 0, otherwise.

In our setup we assume that each user U with instance Π_U^i knows his partners' identities $\mathbf{pid}_U^i$ in a session. Two instances $\Pi_{U_{j_1}}^{i_{j_1}}$ and $\Pi_{U_{j_2}}^{i_{j_2}}$ are *partnered* if $\mathbf{sid}_{U_{j_1}}^{i_{j_1}} = \mathbf{sid}_{U_{j_2}}^{i_{j_2}}$ and $\mathbf{acc}_{U_{j_1}}^{i_{j_1}} = \mathbf{acc}_{U_{j_2}}^{i_{j_2}} = 1$.

An adversary's interaction with principals in the network is modeled by allowing it to have access to the following oracles.

- $Execute(G)$. This query models passive attacks in which the attacker eavesdrops on honest execution of group key management protocol among unused instances $\Pi_{U_1}^{i_1}, \ldots, \Pi_{U_n}^{i_n}$ and outputs the transcript of the execution. A transcript consists of the messages that were exchanged during the honest execution of the protocol.
- $Send(U, i, m)$. This query models an active attack, in which the adversary $\mathcal{A}$ may intercept a message and then either modify it, create a new one or simply forward it to the intended participant. The output of the query is the reply (if any) generated by the instance Π_U^i upon receipt of message m.
- $Reveal(U, i)$. This query unconditionally outputs session key $\mathbf{sk}_U^i$ if it has previously been accepted by Π_U^i, otherwise a value $NULL$ is returned. This query models the misuse of the session keys, i.e., known session key attack.
- $Corrupt(U)$. This query outputs the long-term secret $\mathbf{LS}_U$ (if any) of user U. We say that user U_x is *honest* if and only if no query $Corrupt(U_x)$ has ever been made by the adversary. $Corrupt$(GC) is not allowed since GC is a trusted third party in the adversarial model we adopt.
- $Test(U, i)$. This query is allowed only once, at any time during the adversary's execution. A bit $b \in \{0, 1\}$ is chosen uniformly at random. The adversary is given $\mathbf{sk}_U^i$ if $b = 1$, and a random session key otherwise.

The adversary can ask $Execute$, $Reveal$ and $Corrupt$ queries several times, while $Test$ query is asked only once and on a *fresh* instance. We say that an instance $\Pi_{U_{x_0}}^i$ is *fresh* unless either the adversary, at a certain point, queried $Reveal(U_{x_0}, i)$ or $Reveal(U_{x_1}, j)$ with $(U_{x_1}, j) \in \mathbf{sid}_{U_{x_0}}^i$ or the adversary queried $Corrupt(U_{x_2})$ with $U_{x_2} \in \mathbf{pid}_{U_{x_0}}^i$. The adversary will fail to ask $Send$ query because communications are assumed to be authenticated.

Finally, the adversary outputs a guess bit b'. Such an adversary is said to win the game if $b' = b$, where b is the hidden bit used by $Test$ oracle.

Let $Succ$ denote the event that the adversary $\mathcal{A}$ wins the game for the protocol. We define

$$\mathbf{Adv} := \left| \mathbf{Prob}[\mathbf{Succ}] - \frac{1}{2} \right|$$

to be the advantage of the adversary $\mathcal{A}$ in attacking the protocol.

Definition 3. *We say that a group key management protocol is secure if for any PPT adversary $\mathcal{A}$ who makes q_E* **Execute** *queries, runs in time t and does not violate the freshness of the* **Test** *instance, the advantage* $\mathbf{Adv}(t)$ *is negligible in κ.*

4 The Proposed Scheme Based on Hyper-Sphere

4.1 The Proposed Approach

Inspired by the mathematical principle that any point on the hyper-sphere is at the same distance from the central point, a new secure group key management scheme is proposed.

Before the establishment of a group, the group controller GC chooses a large prime number p and a family of pseudo-random function $F^\kappa = \{f_K : GF(p) \times GF(p) \to GF(p)\}$ which is described in Section 3.3, and publishes them to the public. Hereafter, all computations are conducted over the finite field $GF(p)$.

Intuitively, a hyper-sphere is constructed for the group, and each member in the group corresponds to a point on the hyper-sphere. The GC, who manages the group initialization and membership change operations, computes the central point $\mathbf{C}$ of the hyper-sphere and publishes it to the public. Then each member can calculate $\overline{R}$ via (5) or (6). Therefore, the value $K = (\overline{R} - \|\boldsymbol{C}\|^2)(\mathrm{mod}\, p)$ can be assigned as the group key, which can be computed by all members of the group. Any illegitimate user cannot calculate this value without the knowledge of the legitimate private point, therefore cannot derive the group key.

Our group key management approach includes the phases of initialization, adding members, removing members, massively adding and removing members, and periodically update.

4.1.1 Initialization

The GC lets the first user U_1 join the group at the initialization phase, including the following steps.

Step 1. The GC selects two different two-dimensional private points $\boldsymbol{S}_0 = (s_{00}, s_{01}) \in GF(p)^2$ and $\boldsymbol{S}_1 = (s_{10}, s_{11}) \in GF(p)^2$ at random, and keeps them secret.

Step 2. After authenticating U_1, the GC chooses a two-dimensional private point $\boldsymbol{A}_1 = (a_{10}, a_{11})$ at random for the user U_1, where $a_{10} \neq 0$, $a_{11} \neq 0$ and $a_{10} \neq a_{11}$. The GC stores the point $\boldsymbol{A}_1$ securely and transmits it to the user U_1 via a secure channel.

$\boldsymbol{A}_1$ is the private information of U_1, and should be kept secret by both the member U_1 and the GC.

Step 3. The GC selects a random number $u \in GF(p)$ and computes:

$$b_{00} = f_{s_{00}}(u), b_{01} = f_{s_{01}}(u),$$
$$b_{10} = f_{s_{10}}(u), b_{11} = f_{s_{11}}(u),$$
$$b_{20} = f_{a_{10}}(u), b_{21} = f_{a_{11}}(u),$$

where $f_{s_{00}}(\cdot)$, $f_{s_{01}}(\cdot)$, $f_{s_{10}}(\cdot)$, $f_{s_{11}}(\cdot)$, $f_{a_{10}}(\cdot)$, and $f_{a_{11}}(\cdot)$ are instances of pseudo-random function $f_a(\cdot)$.

Then the GC constructs new points $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$:

$$\boldsymbol{B}_0 = (b_{00}, b_{01}), \boldsymbol{B}_1 = (b_{10}, b_{11}), \boldsymbol{B}_2 = (b_{20}, b_{21}).$$

If

$$2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11}) \not\equiv 0 \ (\mathrm{mod}\ p), \tag{7}$$

go to Step 4; otherwise, the GC repeats Step 3.

Notice that the condition in (7) can guarantee that the points $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$ can uniquely determine a circle in two-dimensional space.

Step 4. The GC establishes a hyper-sphere, herein a circle, in two-dimensional space using the above points $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$. Suppose that the central point of the hyper-sphere is $\mathbf{C} = (c_0, c_1) \in GF(p)^2$. By applying points $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$ to (5) or (6), the GC can construct the following system of equations:

$$\begin{cases} (b_{00} - c_0)^2 + (b_{01} - c_1)^2 \equiv \overline{\boldsymbol{R}} \ (\mathrm{mod}\ p), \\ (b_{10} - c_0)^2 + (b_{11} - c_1)^2 \equiv \overline{\boldsymbol{R}} \ (\mathrm{mod}\ p), \\ (b_{20} - c_0)^2 + (b_{21} - c_1)^2 \equiv \overline{\boldsymbol{R}} \ (\mathrm{mod}\ p). \end{cases} \tag{8}$$

By subtracting the first equation from the second one, and subtracting the second equation from the third one, we can obtain a system of linear equations with two unknowns c_0 and c_1:

$$\begin{cases} 2(b_{00} - b_{10})c_0 + 2(b_{01} - b_{11})c_1 \equiv b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 \ (\mathrm{mod}\ p), \\ 2(b_{10} - b_{20})c_0 + 2(b_{11} - b_{21})c_1 \equiv b_{10}^2 + b_{11}^2 - b_{20}^2 - b_{21}^2 \ (\mathrm{mod}\ p). \end{cases} \tag{9}$$

The condition in (7) guarantees that (9) has one and only one solution (c_0, c_1). Then the central point $\boldsymbol{C} = (c_0, c_1)$ of the hyper-sphere is determined.

Step 5. The GC delivers the public re-keying information $\mathbf{C}$ and u to the member U_1.

Step 6. The member U_1 can calculate the group key by using its private point $\mathbf{A}_1 = (a_{10}, a_{11})$ along with the public information $\boldsymbol{C} = (c_0, c_1)$ and u:

$$\begin{aligned} K &= (\overline{R} - \|\boldsymbol{C}\|^2)(\mathrm{mod}\, p) \\ &= (b_{20}^2 + b_{21}^2 - 2b_{20}c_0 - 2b_{21}c_1)(\mathrm{mod}\, p) \\ &= (f_{a_{10}}^2(u) + f_{a_{11}}^2(u) - 2f_{a_{10}}(u)c_0 - 2f_{a_{11}}(u)c_1) \ (\mathrm{mod}\, p), \end{aligned} \tag{10}$$

where $\boldsymbol{C}$ is the central point of the hyper-sphere, and $\|\boldsymbol{C}\|^2 = c_0^2 + c_1^2$.

Notice that in order to keep our scheme clear and simple, the dimension of the constructed hyper-sphere is designed to equal the number of the group members. Therefore, a one-sphere or a circle is constructed if the condition in (7) is satisfied, since the first member U_1 is enrolled in the group at this phase.

4.1.2 Adding Members

Suppose that there are $n - m$ members in the group before the enrollment of new members, where $n > 0$ and $n > m \geq 0$. Now there are m new members want to join the group. After new members are admitted, there will be n members in the group, which can be denoted by $U_{i_1}, U_{i_2}, \ldots, U_{i_n}$. The steps are as follows.

Step 1. After the new user U_{i_x} is authenticated, the GC selects unique two-dimensional private point $\boldsymbol{A}_{i_x} = (a_{i_x,0}, a_{i_x,1}) \in GF(p)^2$ for each new member U_{i_x}, where $a_{i_x,0} \neq 0$, $a_{i_x,1} \neq 0$, $a_{i_x,0} \neq a_{i_x,1}$, and $x = (n-m)+1, (n-m)+2, \ldots, n$.

The points $\boldsymbol{A}_{i_x}$ should satisfy $\boldsymbol{A}_{i_x} \neq \boldsymbol{A}_{i_y}$ if $x \neq y$, where $1 \leq x, y \leq n$.

Step 2. The GC sends the point $\boldsymbol{A}_{i_x}$ to the user U_{i_x} via a secure channel.

The point A_{i_x} is the private information of U_{i_x}, and should be kept secret by both the member U_{i_x} and the GC.

Step 3. The GC selects a different random number $u \in GF(p)$, and computes

$$b_{00} = f_{s_{00}}(u), b_{01} = f_{s_{01}}(u),$$
$$b_{10} = f_{s_{10}}(u), b_{11} = f_{s_{11}}(u).$$

For $j = 2, 3, \ldots, n+1$, the GC computes

$$b_{j0} = f_{a_{i_{j-1},0}}(u), b_{j1} = f_{a_{i_{j-1},1}}(u),$$

where $f_{a_{i_{j-1},0}}(\cdot)$ is an instance of pseudo-random function $f_a(\cdot)$.

Then the GC constructs new points $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_{n+1}$:

$$\boldsymbol{B}_0 = (b_{00}, b_{01}), \boldsymbol{B}_1 = (b_{10}, b_{11}), \boldsymbol{B}_2 = (b_{20}, b_{21}),$$
$$\cdots\cdots$$
$$\boldsymbol{B}_{n+1} = (b_{n+1,0}, b_{n+1,1}).$$

If the condition

$$\begin{aligned}&(2(b_{00}-b_{10}) \cdot 2(b_{11}-b_{21}) - 2(b_{10}-b_{20}) \cdot 2(b_{01}-b_{11})) \\ &\times \prod_{t=3}^{n+1} (-2b_{t1}) \not\equiv 0 \pmod p\end{aligned} \tag{11}$$

satisfies, go to Step 4; otherwise, the GC repeats Step 3.

Step 4. The GC expands each $\boldsymbol{B}_j$ to become an $(n+1)$-dimensional point $\boldsymbol{V}_j$. Then the GC constructs an n-dimensional hyper-sphere based on the set of points $\boldsymbol{V}_0, \boldsymbol{V}_1, \ldots, \boldsymbol{V}_{n+1}$. Suppose that the central point of the hyper-sphere is $\mathbf{C} = (c_0, c_1, \ldots, c_n) \in GF(p)^{n+1}$.

Step 4.1. The GC expands each $\boldsymbol{B}_j$ to become an $(n+1)$-dimensional point $\boldsymbol{V}_j$.

For $j = 0, 1$, and 2, the point $\boldsymbol{B}_j$ is supplemented $(n-1)$ zeros to become $\boldsymbol{V}_j$, i.e.,

$$\boldsymbol{V}_0 = (b_{00}, b_{01}, 0, \ldots, 0),$$
$$\boldsymbol{V}_1 = (b_{10}, b_{11}, 0, \ldots, 0),$$
$$\boldsymbol{V}_2 = (b_{20}, b_{21}, 0, \ldots, 0).$$

For $j = 3, 4, \cdots, n+1$, let

$$\boldsymbol{V}_3 = (b_{30}, 0, b_{31}, 0, \ldots, 0),$$
$$\cdots\cdots$$
$$\boldsymbol{V}_j = (b_{j0}, 0, \ldots, 0, b_{j1}, 0, \ldots, 0),$$
$$\cdots\cdots$$
$$\boldsymbol{V}_{n+1} = (b_{n+1,0}, 0, \ldots, 0, b_{n+1,1}),$$

where the number of 0 between b_{j0} and b_{j1} is $(j-2)$, and there are $(n+1-j)$ zeros supplemented after b_{j1}.

Step 4.2. The GC constructs a system of equations about the hyper-sphere by applying the set of points $\boldsymbol{V}_0$, $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{n+1}$ to (5) or (6):

$$\begin{cases}(b_{00}-c_0)^2 + (b_{01}-c_1)^2 + (0-c_2)^2 + \cdots + (0-c_n)^2 = \overline{R}, \\ (b_{10}-c_0)^2 + (b_{11}-c_1)^2 + (0-c_2)^2 + \cdots + (0-c_n)^2 = \overline{R}, \\ (b_{20}-c_0)^2 + (b_{21}-c_1)^2 + (0-c_2)^2 + \cdots + (0-c_n)^2 = \overline{R}, \\ (b_{30}-c_0)^2 + (0-c_1)^2 + (b_{31}-c_2)^2 + \cdots + (0-c_n)^2 = \overline{R}, \\ \cdots\cdots \\ (b_{n+1,0}-c_0)^2 + (0-c_1)^2 + \cdots\cdots + (b_{n+1,1}-c_n)^2 = \overline{R}.\end{cases} \tag{12}$$

By subtracting the jth equation from the $(j+1)$th equation in (12), where $j = 1, 2, \ldots, n$, we can obtain a system of linear equations with $(n+1)$ unknowns $c_0, c_1, \ldots,$ and c_n.

$$\begin{bmatrix} 2(b_{00}-b_{10}) & 2(b_{01}-b_{11}) & 0 & \ldots & 0 \\ 2(b_{10}-b_{20}) & 2(b_{11}-b_{21}) & 0 & \ldots & 0 \\ 2(b_{20}-b_{30}) & 2b_{21} & -2b_{31} & \ldots & 0 \\ & & \cdots\cdots & \ldots & \\ 2(b_{n0}-b_{n+1,0}) & 0 & \ldots & \ldots & -2b_{n+1,1} \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \ldots \\ c_n \end{bmatrix} = \begin{bmatrix} b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 \\ b_{10}^2 + b_{11}^2 - b_{20}^2 - b_{21}^2 \\ b_{20}^2 + b_{21}^2 - b_{30}^2 - b_{31}^2 \\ \cdots \\ b_{n0}^2 + b_{n1}^2 - b_{n+1,0}^2 - b_{n+1,1}^2 \end{bmatrix}. \tag{13}$$

Let matrix

$$\boldsymbol{Y} = \begin{bmatrix} 2(b_{00}-b_{10}) & 2(b_{01}-b_{11}) & 0 & \ldots & 0 \\ 2(b_{10}-b_{20}) & 2(b_{11}-b_{21}) & 0 & \ldots & 0 \\ 2(b_{20}-b_{30}) & 2b_{21} & -2b_{31} & \ldots & 0 \\ & & \cdots\cdots & \ldots & \\ 2(b_{n0}-b_{n+1,0}) & 0 & \ldots & \ldots & -2b_{n+1,1} \end{bmatrix}$$

and vectors

$$\boldsymbol{C}^T = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \ldots \\ c_n \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 \\ b_{10}^2 + b_{11}^2 - b_{20}^2 - b_{21}^2 \\ b_{20}^2 + b_{21}^2 - b_{30}^2 - b_{31}^2 \\ \cdots \\ b_{n0}^2 + b_{n1}^2 - b_{n+1,0}^2 - b_{n+1,1}^2 \end{bmatrix},$$

where $\boldsymbol{C}^T$ denotes the transpose of $\boldsymbol{C}$.

Then (13) can be expressed in the matrix and vector form

$$\boldsymbol{Y} \times \boldsymbol{C}^T = \boldsymbol{Z}. \tag{14}$$

The condition in (11) guarantees that (13) or (14) has one and only one solution $\boldsymbol{C}^T = \boldsymbol{Y}^{-1} \times \boldsymbol{Z}$. Then the central point $\boldsymbol{C} = (c_0, c_1, \ldots, c_n)$ of the hyper-sphere is determined.

Step 5. The GC multicasts the public re-keying information $\boldsymbol{C}$ and u to all group members $U_{i_1}, U_{i_2}, \ldots, U_{i_n}$.

Step 6. Each group member U_{i_x} can calculate the group key by using its private point $A_{i_x}(a_{i_x,0}, a_{i_x,1})$ along with the public information $\boldsymbol{C} = (c_0, c_1, \ldots, c_n)$ and u:

$$\begin{aligned}K &= \left(\overline{R} - \|\boldsymbol{C}\|^2\right) (\mathrm{mod}\, p) \\ &= \left(b_{x+1,0}^2 + b_{x+1,1}^2 - 2b_{x+1,0}c_0 - 2b_{x+1,1}c_{i_x}\right) (\mathrm{mod}\, p) \\ &= \left(f_{a_{i_x,0}}^2(u) + f_{a_{i_x,1}}^2(u) - 2f_{a_{i_x,0}}(u)c_0 - 2f_{a_{i_x,1}}(u)c_{i_x}\right) \pmod p,\end{aligned} \tag{15}$$

where $\boldsymbol{C}$ is the central point of the hyper-sphere, and $\|\boldsymbol{C}\|^2 = c_0^2 + c_1^2 + \cdots + c_n^2$.

4.1.3 Removing Members

Suppose that there are $n + w$ members in the group before membership exclusion, where $n > 0$ and $w \geq 0$. Now there are w members want to leave the group, then there will be n members in the group after w users leave. Suppose that the set of remaining members in the group is $\{U_{i_1}, U_{i_2}, \ldots, U_{i_n}\}$ after removing members. The steps are as follows.

Step 1. The GC deletes the leaving members' private two-dimensional points.

Step 2. The GC's private two-dimensional points $\boldsymbol{S}_0$ and $\boldsymbol{S}_1$, and the remaining members' private points $\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \ldots, \boldsymbol{A}_{i_n}$ should be stored securely by the GC.

The following steps are the same as Steps 3-6 in the "Adding Members" phase, i.e., the GC re-selects a new random number $u \in GF(p)$ and constructs new points $\boldsymbol{B}_0, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_{n+1}$ in Step 3. Then the GC constructs a new hyper-sphere in Step 4, and publishes the new random number u and the new central point $\boldsymbol{C}$ of the hyper-sphere in Step 5. Finally, each group member can calculate the new group key by using its private point in Step 6.

4.1.4 Massively Adding and Removing Members

This phase manipulates the situation that a lot of members join and other members leave the group at the same time in batch mode. Suppose that there are $n + w - m$ members in the group before membership change, where $n > 0$ and $w \geq 0$, $n + w > m \geq 0$. Now there are w members want to leave, and m new members want to join the group simultaneously. After the membership update, there will be n members in the group. The steps are as follows.

Step 1. The GC deletes the leaving members' private two-dimensional points, and let new users join in at the same time. After new user U_{i_x} is authenticated, the value of i_x is assigned as the identifier of the new joining members, where $x = (n - m) + 1, (n - m) + 2, \ldots, n$.

The GC selects unique two-dimensional point $\boldsymbol{A}_{i_x} = (a_{i_x,0}, a_{i_x,1}) \in GF(p)^2$ as U_{i_x}'s private information, where $a_{i_x,0} \neq 0, a_{i_x,1} \neq 0$, and $a_{i_x,0} \neq a_{i_x,1}$. The private points $\boldsymbol{A}_{i_x}$ should satisfy $\boldsymbol{A}_{i_x} \neq \boldsymbol{A}_{i_y}$ if $x \neq y$, where $1 \leq x, y \leq n$.

Step 2. The GC sends the private point A_{i_x} to the user U_{i_x} via a secure channel.

The point $\boldsymbol{A}_{i_x}$ is the private information of U_{i_x}, and should be kept secret by both the member U_{i_x} and the GC.

Other steps are for w members to leave the group, which are the same as Steps 3-6 described in the "Adding Members" phase. By executing Steps 3 to 6, the GC re-selects a new random number $u \in GF(p)$, constructs a new hyper-sphere, and publishes the new random number u and the new central point $\boldsymbol{C}$ of the hyper-sphere. Then each group member can calculate a new group key.

4.1.5 Periodically Update

If the group key is not updated within a period of time, the GC will start periodically update procedure to renew the group key to safeguard the secrecy of group communication. The GC needs to re-select a new random number $u \in GF(p)$, then construct a new hyper-sphere, and publish the new random number u and the new central point of the hyper-sphere. These steps are the same as Steps 3-6 in "Adding Members" phase.

TABLE 1
Performance Requirements by the GC and Each Member

	Storage (bits)	Computation	Re-keying Messages	
			Number	Size(bits)
GC	$2 \times (n+2) \times L$	$O(n)$	2	$(n+2) \times L$
Member	$2 \times L$	$2H + 4M + 5A$	0	0

Notation for Table 1:
n : number of members in the group
L : the length of the prime p in bits
H : average time required by an f function
M : average time required by a modular multiplication
A : average time required by a modular addition

5 Security and Performance Analysis

5.1 Security Analysis

We will show that our group key management protocol is secure under PRF assumption and assuming that communications are authenticated. The authenticated communications can be achieved by incorporating a standard digital signature scheme into the protocol. More concretely, we can authenticate each transfered message via a digital signature during each protocol execution. Similar in [18], Dutta and Barua first presented a secure unauthenticated group key agreement protocol, and then proved that an authenticated group key agreement protocol can be obtained by authenticating each message via a signature during each protocol execution.

Two lemmas and a theorem, i.e., Lemma 2, Lemma 3, and Theorem 4, are given and proven mathematically in the supplementary file, available in the online supplemental material. Theorem 4 declares that an adversary is unable to distinguish a real group key generated by our scheme with a randomly chosen faked key. That is to say, by using all information that it possesses and all capabilities that it owns, the advantage of the adversary to derive the correct group key is equivalent to guessing the key randomly, whose success probability is $\frac{1}{p}$ since the cardinality of the key space is p. Therefore, our scheme is formally proven to be secure under PRF assumption.

5.2 Performance Analysis

Suppose that the length of the prime p in binary expression is L bits. Table 1 shows the performance requirements by both the GC and each member.

Storage. Each member needs to store its two-dimensional private point. The GC should store all members' two-dimensional private points. Then the storage for each member is $2 \times L$ bits, and the storage for the GC is $2 \times (n + 2) \times L$ bits.

Computation. The major computation by each member is to calculate the group key via (15), which includes two computations of f function, four modular multiplications and five modular additions over finite field. The computation for the GC is to solve a system of linear equations. Since the coefficient matrix in (13) can easily be converted to a lower

TABLE 2
Storage and Computation Required by the GC

Size of group	Storage (bytes)	Computation (ms)	
		Adding Members	Removing Members
10	384	0.06	0.06
100	3,264	0.4	0.4
1,000	32,064	0.7	0.7
10,000	320,064	7.7	7.7
100,000	3,200,064	85.2	85.2

TABLE 3
Storage and Computation Required by Each Member

Size of group	Storage (bytes)	Computation (ms)	
		Adding Members	Removing Members
10	32	0.00564	0.00564
100	32	0.00564	0.00564
1,000	32	0.00564	0.00564
10,000	32	0.00564	0.00564
100,000	32	0.00564	0.00564

triangular matrix, the computation complexity of solving $(c_0, c_1, \ldots, c_n)$ from (13) is $\mathcal{O}(n)$.

Number and size of re-keying message. The total number of re-keying messages is two, including the central point of the hyper-sphere and the random number u. The size of re-keying messages is $(n+2) \times L$ bits.

Batch processing. If there are a lot of users join and leave the group simultaneously, only one batch processing is needed.

5.3 Experiments

While f can be any computationally efficient function assumable to be pseudo-random, we instantiate it by a cryptographic hash function to ease the comparison. Our experimental test bed for the GC is a 2.33 GHz Intel Xeon quad-core dual-processor PC server with 4 GB memory and running Linux, and the platform for the member is a HP XW4600 Workstation with 2.33 GHz Intel dual-processor and 2 GB memory and running Linux. C/C++ programming language is adopted to compose the software to simulate the behavior of the GC and members. We choose $L = 128$ bits, which denotes the length of the prime p in binary form, then we compute the average cost of the GC and each member. The time was averaged over 20 distinct runs of the experiments, and the difference among the same experiments is less than 2 percent. The summary of the experimental results are presented in Tables 2 and 3.

In Table 2, the first column represents the size of the group; the second, the storage for the computation, and the third and fourth, the computation time. For a large group $n = 100{,}000$, the GC takes 85.2 ms = 0.0852 seconds to process member adding or removing. We can observe from this experimental data that the GC can manage a large group efficiently.

Table 3 shows that the storage and the computation cost does not increase at all for each group member even when the group size increases, which is very desirable.

Our experimental results confirm that our scheme is very scalable and very efficient for large groups.

6 Comparison with Related Work

Our scheme falls into the category of centralized systems, therefore we will compare our scheme with some typical centralized key management schemes. A summary of the comparison results are presented in Tables 4 and 5.

Group Key Management Protocol is a simple extension from unicast to multicast, but not scalable and very inefficient. Table 4 clearly shows that our scheme outperforms GKMP.

The Logical Key Hierarchy scheme can be considered to be the representative of tree-based schemes, including OFT [5], OFCT [6], Hierarchical α-ary Tree with Clustering [7], Efficient Large-Group Key [8], etc. Hence, we compare our scheme with LKH only, but the results are similar to other tree-based schemes.

The advantages of our scheme over the LKH are as follows: 1) our scheme is scalable for massive membership change; 2) the number of re-keying messages is $\mathcal{O}(1)$ in our scheme, but is $\mathcal{O}(\log_2 n)$ in LKH; 3) the computation complexity of each member is $\mathcal{O}(1)$ in our scheme, but is $\mathcal{O}(\log_2 n)$ in LKH.

TABLE 4
Feature and Computation Complexity Comparison among Existing Schemes

	GKMP	LKH	Secure Lock	Our Scheme
Major principle adopted	Encryption	Tree structure	Chinese Remainder Theorem	Hyper-sphere
Efficient for very large group	No	Yes	No	Yes
Scalable to massively adding and removing members	No	No	Yes	Yes
Number of re-keying messages	n	$O(\log_2 n)$	$O(1)$	$O(1)$
Member computation complexity	$O(1)$ decryptions	$O(\log_2 n)$ decryptions	$O(1)$ decryptions and modular operations	$O(1)$ simple operations
GC computation complexity	$O(n)$ encryptions	$O(\log_2 n)$ encryptions	$O(n)$ encryptions and modular operations	$O(n)$ simple operations

TABLE 5
GC's Computation Comparison between Secure Lock and Our Scheme

	Secure Lock	Our Scheme
Computation complexity	$E \cdot O(n) + M \cdot O(2n) + A \cdot O(n) + R \cdot O(2n)$	$H \cdot O(2n) + M \cdot O(2n) + A \cdot O(4n) + R \cdot O(n)$
Difference between schemes	$E \cdot O(n) + R \cdot O(n)$	$2H \cdot O(n) + 3A \cdot O(n)$

Notation for TABLE 5:
n : number of members in the group
M : average time required by a modular multiplication over $GF(p)$
A : average time required by a modular addition over $GF(p)$
E : average time required by a symmetric encryption
H : average time required by a hash function
R : average time required by a multiplication reverse over $GF(p)$

The major differences between our scheme and LKH are: 1) the principles behind are different: hyper-sphere is adopted in our scheme, but tree structure is adopted in LKH; 2) The computation complexity by the GC in our scheme is $\mathcal{O}(n)$ simple operations, but the one in LKH is $\mathcal{O}(2\log_2 n)$ encryptions. In average conditions, the computation of simple operations can be faster than encryptions.

Notice that tree structure can also be adopted by our scheme to divide the members into different sub-trees and to further speed up our scheme. We will explore this direction in our future research.

The Secure Lock is based on Chinese Remainder Theorem, which is a time-consuming operation. Hence, the SL scheme is applicable only for small groups [3].

There are some similarities between the SL and our scheme: 1) both schemes can be regarded as flat structure, that is, no hierarchical structures such as tree structures are adopted; 2) the numbers of re-keying messages in both schemes are $\mathcal{O}(1)$; 3) the computation complexity by each member in both schemes are also $\mathcal{O}(1)$; 4) the computation complexity by the GC in both schemes are $\mathcal{O}(n)$.

Table 5 compares the computation complexity by the GC in the SL and our scheme. The one in the SL is based on an optimized CRT [3]. The first row presents the computation complexity, and the second row shows the difference of computation complexity of two schemes by omitting the identical items in the first row. The complexity differences are: $E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$ in the SL, and $2H \cdot \mathcal{O}(n) + 3A \cdot \mathcal{O}(n)$ in our scheme, where n is the number of members in the group, E, R, H and A are the average time required by encryption, modular multiplication reverse, f function, and modular addition, respectively. Usually, we can choose a pseudo-random function f that can be computed very fast, so $E > 2H$. Modular reverse operation is much slower than the addition operation over finite field, thus $R \gg 3A$, and then

$$E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n) \gg 2H \cdot \mathcal{O}(n) + 3A \cdot \mathcal{O}(n),$$

or

$$E \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(2n) + A \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(2n)$$

$$\gg H \cdot \mathcal{O}(2n) + M \cdot \mathcal{O}(2n) + A \cdot \mathcal{O}(4n) + R \cdot \mathcal{O}(n).$$

Hence, the computation of our scheme is much faster than that of SL.

Therefore, the advantages of our scheme over the ones of the SL include: 1) our scheme is efficient for very large group; 2) the performance by each member and the GC in our scheme is much better than the ones in SL.

Our scheme belongs to the category of centralized systems. Thus some common disadvantages of the centralized ones, like the group controller being a single point of failure, are also employed by our scheme. The failure of the group controller could compromise the system completely. This is one main disadvantage compared with distributed or decentralized schemes. However, some techniques to prevent the failure of single point, for example [46], can be adopted to alleviate this disadvantage. In addition, our scheme can be a fundamental component to construct some decentralized schemes by combining other techniques.

7 Conclusions

In this paper, we study the problem of group key management from a very different angle than before. A new secure group key management scheme based on hyper-sphere is constructed, where each member in the group corresponds to a private point on the hyper-sphere and the group controller computes the central point of the hyper-sphere, intuitively, whose "distance" from each member's private point is identical. The central point is published, and each member can compute a common group key using a function by taking each member's private point and the central point of the hyper-sphere as the input. Our new approach is formally proved secure under the pseudo-random function assumption.

The advantages of our scheme include: (1) public re-keying messages can be broadcasted or multicasted to group members; (2) it is very efficient and scalable for large-size groups and can deal with massive membership change efficiently with only two re-keying messages, i.e., the central point of the hyper-sphere and a random number; (3) both the storage and the computation overhead of each member is significantly reduced, which is independent of the group size; and (4) the GC's storage and computation cost is also acceptable: the storage and computation overhead increases linearly with the group size.

The performance estimations are further confirmed by our experiments. For example, in the case of a group of size $n = 100{,}000$, the storage cost for each member's private information is 32 bytes, the time for each member to compute the group key is 0.000564 ms or 5.64×10^{-7} seconds, and the time for the GC to process membership change is 85.2 ms or 8.52×10^{-4} seconds on a 2.33 GHz Intel Xeon quad-core dual-processor PC server.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. U1135004, 61170080, 61202466, 61202450), Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011), High-level Talents Project of Guangdong Institutions of Higher Education (2012), Fok Ying Tung Education Foundation (Grant No. 141065), Ph.D. Programs Foundation of Ministry of Education of China (Grant NO. 20123503120001), and Distinguished Young Scholars Fund of Department of Education (JA13062), Fujian Province, China. The authors are most grateful for the constructive advice on the revision of the manuscript from the anonymous reviewers.

References

[1] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, vol. 35, no. 3, pp. 309-329, 2003.

[2] H. Harney, "Group Key Management Protocol (GKMP) Specification," *RFC 2003*, http://tools.ietf.org/rfc/rfc2093.txt, 1997.

[3] G.-H. Chiou and W.-T. Chen, "Secure Broadcasting Using the Secure Lock," *IEEE Trans. Software Eng.*, vol. 15, no. 8, pp. 929-934, Aug. 1989.

[4] C.K. Wong, M. Gouda, and S.S. Lam, "Secure Group Communications Using Key Graphs," *IEEE/ACM Trans. Networking*, vol. 8, no. 1, pp. 16-30, Feb. 2000.

[5] A.T. Sherman and D.A. McGrew, "Key Establishment in Large Dynamic Groups Using One-Way Function Trees," *IEEE Trans. Software Eng.*, vol. 29, no. 5, pp. 444-458, May 2003.

[6] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," *Proc. IEEE 18th Ann. Joint Conf. Computer and Comm. Soc.*, vol. 2, pp. 708-716, Mar. 1999.

[7] R. Canetti, T. Malkin, and K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," *Proc. Advances in Cryptology: 17th Int "l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT' 99)*, pp. 459-474, 1999.

[8] A. Penrig, D. Song, and J.D. Tygar, "ELK, A New Protocol for Efficient Large-Group Key Distribution," *Proc. IEEE Symp. Security and Privacy*, pp. 247-262, 2001.

[9] M. Burmester and Y. Desmedt, "A Secure and Efficient Conference Key Distribution System (Extended Abstract)," *Proc. Advances in Cryptology: Workshop the Theory and Application of Cryptographic Techniques (EUROCRYPT '94)*, pp. 275-286, 1995.

[10] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," *Proc. Third ACM Conf. Computer and Comm. Security*, pp. 31-37, 1996.

[11] K. Becker and U. Wille, "Communication Complexity of Group Key Distribution," *Proc. Fifth ACM Conf. Computer and Comm. Security*, pp. 1-6, 1998.

[12] C. Boyd, "On Key Agreement and Conference Key Agreement," *Proc. Information Security and Privacy: Second Australasian Conf.*, pp. 294-302, 1997.

[13] O. Rodeh, K. Birman, and D. Dolev, "Optimized Group Rekey for Group Communications Systems," *Proc. Symp. Network and Distributed System Security*, 2000.

[14] L. Dondeti, S. Mukherjee, and A. Samal, "A Distributed Group Key Management Scheme for Secure Many-to-Many Communication," Technical Report PINTL-TR-207-99, Department of Computer Science, Univ. of Maryland, 1999.

[15] A. Perrig, "Efficient Collaborative Key Management Protocols for Secure Autonomous Group Communication," *Proc. Int'l Workshop Cryptographic Techniques and E-Commerce*, pp. 192-202, 1999.

[16] Y. Kim, A. Perrig, and G. Tsudik, "Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups," *Proc. Seventh ACM Conf. Computer and Comm. Security*, pp. 235-244, 2000.

[17] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey Framework: Versatile Group Key Management," *IEEE J. Selected Areas in Comm.*, vol. 17, no. 9, pp. 1614-1631, Sept. 1999.

[18] R. Dutta and R. Barua, "Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting," *IEEE Trans. Information Theory*, vol. 54, no. 5, pp. 2007-2025, May 2008.

[19] W. Yu, Y. Sun, and K.J.R. Liu, "Optimizing Rekeying Cost for Contributory Group Key Agreement Schemes," *IEEE Trans. Dependable and Secure Computing*, vol. 4, no. 3, pp. 228-242, July-Sept. 2007.

[20] P.P.C. Lee, J.C.S. Lui, and D.K.Y. Yau, "Distributed Collaborative Key Agreement and Authentication Protocols for Dynamic Peer Groups," *IEEE/ACM Trans. Networking*, vol. 14, no. 2, pp. 263-276, Apr. 2006.

[21] Y. Mao, Y. Sun, M. Wu, and K.J.R. Liu, "JET: Dynamic Join-Exit-Tree Amortization and Scheduling for Contributory Key Management," *IEEE/ACM Trans. Networking*, vol. 14, no. 5, pp. 1128-1140, Oct. 2006.

[22] Y. Amir, C. Nita-Rotaru, S. Stanton, and G. Tsudik, "Secure Spread: An Integrated Architecture for Secure Group Communication," *IEEE Trans. Dependable and Secure Computing*, vol. 2, no. 3, pp. 248-261, Aug. 2005.

[23] Y. Amir, Y. Kim, C.Nita-Rotaru, J.L. Schultz, S. Stanton, and G. Tsudik, "Secure Group Communication Using Robust Contributory Key Agreement," *IEEE Trans. Parallel and Distributed Systems*, vol. 15, no. 5, pp. 468-480, May 2004.

[24] Q. Wu, Y. Mu, W. Susilo, B. Qin, and J.Domingo Ferrer, "Asymmetric Group Key Agreement," *Proc. Advances in Cryptology: Workshop the Theory and Application of Cryptographic Techniques (EUROCRYPT'09)*, pp. 153-170, 2009.

[25] Q. Wu, B. Qin, L. Zhang, J. Domingo Ferrer, and O. Farrà, "Bridging Broadcast Encryption and Group Key Agreement," *Proc. Advances in Cryptology: 17th Int'l Conf. The Theory and Application of Cryptology and Information Security (ASIACRYPT '11)*, pp. 143-160, 2011.

[26] Q. Wu, B. Qin, L. Zhang, and J.Domingo-Ferrer, J. Manjn, "Fast Transmission to Remote Cooperative Groups: A New Key Management Paradigm," *IEEE/ACM Trans. Networking*, vol. 21, no. 2, pp. 621-633, Apr. 2013.

[27] A. Ballardie, "Scalable Multicast Key Distribution," *RFC1949*, 1996.

[28] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," *Proc. ACM SIGCOMM'97*, pp. 277-288, 1997.

[29] L.R. Dondeti, S. Mukherjee, and A. Samal, "Scalable Secure One-to-Many Group Communication Using Dual Encryption," *Computer Comm.*, vol. 23, no. 17, pp. 1681-1701, 2000.

[30] B. Briscoe, "MARKS: Zero Side Effect Multicast Key Management Using Arbitrarily Revealed Key Sequences," *Proc. First Int'l Workshop Networked Group Comm.*, pp. 301-320, 1999.

[31] R. Molva and A. Pannetrat, "Scalable Multicast Security in Dynamic Groups," *Proc. Sixth ACM Conf. Computer and Comm. Security*, pp. 101-112, 1999.

[32] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Re-Keying Approach for Secure Multicast," *Proc. IEEE Symp. Security and Privacy*, pp. 215-228, May 2000.

[33] B. DeCleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhang, "Secure Group Communications for Wireless Networks," *Proc. IEEE Military Comm. Conf. Comm. for Network-Centric Operations: Creating the Information Force (MILCOM '01)*, vol. 1, pp. 113-117, 2001.

[34] S. Rafaeli and D. Hutchison, "Hydra: A Decentralised Group Key Management," *Proc. 11th IEEE Int'l Workshops Enabling Technologies: Infrastructure for Collaborative Enterprises*, pp. 62-67, 2002.

[35] B. Rong, H.-H. Chen, Y. Qian, K. Lu, R.Q. Hu, and S. Guizani, "A Pyramidal Security Model for Large-Scale Group-Oriented Computing in Mobile Ad Hoc Networks: The Key Management Study," *IEEE Trans. Vehicular Technology*, vol. 58, no. 1, pp. 398-408, Jan. 2009.

[36] Q. Gu, P. Liu, W.-C. Lee, and C.-H. Chu, "KTR: An Efficient Key Management Scheme for Secure Data Access Control in Wireless Broadcast Services," *IEEE Trans. Dependable and Secure Computing*, vol. 6, no. 3, pp. 188-201, July-Sept. 2009.

[37] X. Yi, C.-K. Siew, C.H. Tan, and Y. Ye, "A Secure Conference Scheme for Mobile Communications," *IEEE Trans. Wireless Comm.*, vol. 2, no. 6, pp. 1168-1177, Nov. 2003.

[38] J. Salido, L. Lazos, and R. Poovendran, "Energy and Bandwidth-Efficient Key Distribution in Wireless Ad Hoc Networks: A Cross-Layer Approach," *IEEE/ACM Trans. Networking*, vol. 15, no. 6, pp. 1527-1540, Dec. 2007.

[39] Y. Sun, W. Trappe, and K.J.R. Liu, "A Scalable Multicast Key Management Scheme for Heterogeneous Wireless Networks," *IEEE/ACM Trans. Networking*, vol. 12, no. 4, pp. 653-666, Aug. 2004.

[40] X. Yi, C.-K. Siew, and C.H. Tan, "A Secure and Efficient Conference Scheme for Mobile Communications," *IEEE Trans. Vehicular Technology*, vol. 52, no. 4, pp. 784-793, Aug. 2003.

[41] K. Ren, W. Lou, B. Zhu, and S. Jajodia, "Secure and Efficient Multicast in Wireless Sensor Networks Allowing Ad Hoc Group Formation," *IEEE Trans. Vehicular Technology*, vol. 58, no. 4, pp. 2018-2029, May 2009.

[42] Y. Kim, M. Narasimha, and G. Tsudik, "Secure Group Key Management for Storage Area Networks," *IEEE Comm. Magazine*, vol. 41, no. 8, pp. 92-99, Aug. 2003.

[43] M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom Functions Revisited: The Cascade Construction and Its Concrete Security," *Proc. 37th Ann. Symp. Foundations of Computer Science*, pp. 514-523, 1996.

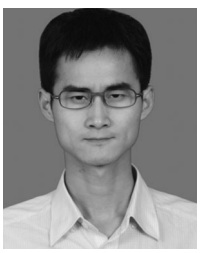
[44] S. Goldwasser and M. Bellare, "Lecture Notes on Cryptography," *Cryptography and Computer Security*. MIT, 2008.

[45] M.J. Atallah, M. Blanton, N. Fazio, and K.B. Frikken, "Dynamic and Efficient Key Management for Access Hierarchies," *ACM Trans. Information System Security*, vol. 12, no. 3, article 18, Jan. 2009.

[46] S.K. Srivastava, *Method for Overcoming the Single Point of Failure of the Central Group Controller in a Binary Tree Group Key Exchange*, US, Patent 7,260,716, Aug. 2007.

Shaohua Tang received the BSc and MSc degrees in applied mathematics, and the PhD degree in communication and information system all from the South China University of Technology, in 1991,1994, and 1998, respectively. He was a visiting scholar with North Carolina State University, and a visiting professor with the University of Cincinnati, Ohio. He has been a full professor with the School of Computer Science and Engineering, South China University of Technology since 2004. His current research interests include information security, networking, and information processing. He is a member of the IEEE and the IEEE Computer Society.

Lingling Xu received the BSc and MSc degrees in mathematics both from Shandong University, China, in 2005 and 2008, respectively, and the PhD degree in communication and information system from Sun Yat-sen University, in 2011. She is currently an assistant professor with the School of Computer Science and Engineering, South China University of Technology. Her current research interests include cryptography and cloud computing.



Niu Liu received the BS degree in mathematics from the University of Electronic Science and Technology, in 2005, and the MS degree in mathematics from the South China University of Technology in 2008. He is currently working toward the PhD degree in computer science at the South China University of Technology. His research interests include group key management protocols, distributed systems, cryptography, and network security.

Xinyi Huang received the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Computer Science, Fujian Normal University, China. His research interests include cryptography and information security. He has published more than 60 research papers in refereed international conferences and journals. His work has been cited more than 1,000 times at Google Scholar. He is in the editorial board of the *International Journal of Information Security* and has served as the program/general chair or program committee member in more than 40 international conferences.

Jintai Ding received the PhD degree from Yale University, in 1995. He was a lecturer at Kyoto University, Japan, from 1995 to 1998. He has been a full professor at the Department of Mathematical Sciences of University of Cincinnati,Ohio since 2006. His research interests include cryptography, computational algebra, and information security. He was a Humboldt fellow from 2006 to 2007.

Zhiming Yang received the BSc degree in computer science from Fuzhou University, China, in 2007, and the MSc degree in computer science from the South China University of Technology in 2010. His research interests include cryptography and information security.