Provably Secure Group Key Management Approach Based upon Hyper-sphere

Shaohua Tang[1,2], Lingling Xu[1], Niu Liu[1], Jintai Ding[2,3], and Zhiming Yang[1]

[1] School of Computer Science & Engineering,
South China University of Technology, Guangzhou, China
shtang@IEEE.org, csshtang@scut.edu.cn
[2] Department of Mathematical Sciences, University of Cincinnati, OH, USA
jintai.ding@mail.uc.edu
[3] Dept. of Applied Math., South China University of Technology, China
jintai.ding@gmail.com

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.

Keywords: 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 while 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 pseudo-random function.

Organization. The remainder of this paper is organized as follows. A brief survey of some related schemes on secure group key management is described in Section 2. Some preliminaries and security model are given in Section 3. The proposed secure group key management approach is presented in Section 4. Security is formally proven, and performance is discussed in Section 5. Comparisons with related work 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 the key management for secure group communication. Rafaeli and Hutchison [30] 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 GC (Group Controller) controlling the whole group [30]. Some typical schemes in this category include Group Key Management Protocol (GKMP) [19, 20], Secure Lock (SL) [12], Logical Key Hierarchy (LKH) [41], etc. The Group Key Management Protocol (GKMP) [19, 20] 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 and encrypts the new group key with the old group key yielding $K' = E_{K_0}(K_N)$ then broadcasts K' to the group members. Moreover, the GC sends K_N to the joining member via the secure channel between the GC and the new member. Obviously, the solution is not scalable [30]. The Secure Lock (SL) scheme [12] takes advantage of Chinese Remainder Theorem (CRT) to construct a secure lock to combine all the re-keying messages into a single message while the group key is updated. However, CRT is a time-consuming operation. As mentioned in [12], 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 (LKH) scheme [41] 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 $O(\log_2 n)$ and the number of encryptions is $O(2 \times \log_2 n)$. 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, OFT (One-way Function Tree) [37], OFCT (One-way Function Chain Tree) [10], Hierarchical α-ary Tree with Clustering [11], Efficient Large-Group Key [29], etc.

In the distributed schemes, there is no explicit GC and the key generation can be either contributory or done by one of the members [30]. Some typical schemes include: Burmester and Desmedt Protocol [9], Group Diffie-Hellman key exchange [38], Octopus Protocol [5], Conference Key Agreement [7], Distributed Logical Key Hierarchy [34], Distributed One-way Function Tree [16], Diffie-Hellman Logical Key Hierarchy [28, 21], Distributed Flat Table [40], etc. Recent references paid more attentions to contributory and collaborative group key agreement [14, 46, 24, 25, 1, 2], etc. Recently, the concepts of asymmetric group key agreement and contributory broadcast encryption were proposed [42, 43]. An asymmetric group key agreement (ASGKA) protocol [42] 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 (CBE) [43] enables a group of members negotiate a common public encryption key while each member holds a decryption key.

In the decentralized architectures, the large group is split into small subgroups. Different controllers are used to manage each subgroup [30]. Some typical schemes include: Scalable Multicast Key Distribution [4], Iolus [26], Dual-Encryption Protocol [15], MARKS [8], Cipher Sequences [27], Kronos [36], Intra-Domain Group Key Management [13], Hydra [31], etc.

The secure group key management approaches can be applied to a lot of application areas. For example: wireless/mobile network [33, 18, 44, 35, 39, 45], wireless sensor network [32], storage area networks [22], etc.

3 Preliminaries

In this section, we briefly introduce the concept of hyper-sphere, and present some syntax used throughout this paper. Then we define Pseudo-Random Function (PRF), and describe the security model in 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, an 0-sphere is a pair of points on a line, an 1-sphere illustrated in Fig. 1 is a circle in a plane, and an 2-sphere is an ordinary sphere in three-dimensional space. Spheres of dimension $N > 2$ are sometimes called hyper-spheres.

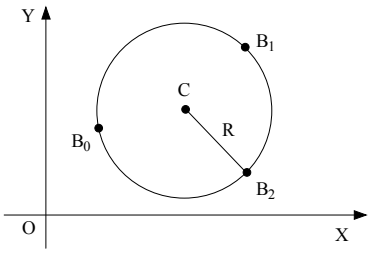

Fig. 1. An 1-sphere or a circle in a plane

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

$$(x_0 - c_0)^2 + (x_1 - c_1)^2 + \ldots + (x_N - c_N)^2 = r^2. \tag{1}$$

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

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

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

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

If and if only the determinant of the coefficients in (3) is non-zero, this system of linear equations can have 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.

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 $\boldsymbol{C} = (c_0, c_1, \ldots, c_N) \in GF(p)^{N+1}$, we define function

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

as

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

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

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

For a given $\overline{R} \in GF(p)$, the hyper-sphere determined by $\overline{R}$ and $\boldsymbol{C}$ is defined by

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

or

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

Notice that only $\overline{R}$ is needed in our scheme, and the square-root of $\overline{R}$ over $GF(p)$ is never required throughout this paper. The square-root may not always be a valid operation over $GF(p)$.

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 run with fresh random coins. We use the notation $u \leftarrow_R S$ to denote that u is chosen randomly from S. 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 $\nu : \mathbb{N} \rightarrow [0, 1]$ is negligible if for all $c \in \mathbb{N}$ there exists a $\kappa_c \in \mathbb{N}$ such that $\nu(\kappa) < \kappa^{-c}$ for all $\kappa > \kappa^c$.

3.3 Pseudo-Random Function (PRF)

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 a set of all functions from D to R. We adopt some expressions of pseudo-random function in [6, 17], and its definition is given as follows.

Definition 1 (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 $\mathbf{EXP^{prf-1}}(\mathcal{D})$	Experiment $\mathbf{EXP^{prf-0}}(\mathcal{D})$
$K \leftarrow_R Keys(F^\kappa)$	$g \leftarrow_R Func$
$b \leftarrow \mathcal{D}(F_K)$	$b \leftarrow \mathcal{D}(g)$
return b	*return b*

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

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

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

In our construction of group key management protocol, we specify a family of pseudo-random functions $F^\kappa : GF(p) \times GF(p) \rightarrow GF(p)$, i.e. $F^\kappa = \{f_a(\cdot) \mid a \in GF(p)\}$. 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.

Our adversarial model described below is similar to the formal security model of Atallah *et al.* [3] and Dutta *et al.* [14]. Let $\mathcal{P} = \{U_1, U_2, \cdots, U_N\}$ 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. The interaction between the adversary $\mathcal{A}$ and the protocol users occur only by querying oracles, which models the adversary's capabilities in real attacks. Let G, G_1, and G_2 be three user sets such that $G \cap G_1 = \phi$ and $G_2 \subseteq G$. More precisely, let $G = \{(\overline{U}_1, i_1), ..., (\overline{U}_n, i_n)\}$, $G_1 = \{(\overline{U}_{n+1}, i_{n+1}), ..., (\overline{U}_{n+k}, i_{n+k})\}$, $G_2 = \{(\overline{U}_{j_1}, i_{j_1}), ..., (\overline{U}_{j_k}, i_{j_k})\}$, where $\{\overline{U}_1, ..., \overline{U}_n\}$ is any non-empty subset 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.
$\mathbf{\Pi}^i_U$ The i-th instance of user U.
$\mathbf{sk}^i_U$ Session key after execution of the protocol by $\mathbf{\Pi}^i_U$.
$\mathbf{sid}^i_U$ Session identity for instance $\mathbf{\Pi}^i_U$. We set $\mathbf{sid}^i_U = G = \{(U_1, i_1), \cdots, (U_n, i_n)\}$ such that $(U, i) \in G$ and users $U_1, \cdots, U_n$ wish to agree upon a common key in a session using unused instances $\mathbf{\Pi}^{i_1}_{U_1}, \cdots, \mathbf{\Pi}^{i_n}_{U_n}$.
$\mathbf{pid}^i_U$ Partner identity for instance $\mathbf{\Pi}^i_U$, defined by $\mathbf{pid}^i_U = \{U_1, \cdots, U_n\}$, such that $(U_j, i_j) \in \mathbf{sid}^i_U$ for all $1 \le j \le n$, where i_j comes from $\mathbf{sid}^i_U$ defined above.
$\mathbf{acc}^i_U$ 0/1-valued variable which is set to be 1 by $\mathbf{\Pi}^i_U$ upon normal termination of the session and 0, otherwise.

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

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 $\mathbf{\Pi}^{i_1}_{U_1}, ..., \mathbf{\Pi}^{i_n}_{U_n}$ 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 $\mathbf{\Pi}^i_U$ upon receipt of message m.
- **Reveal**(U, i) : This query unconditionally outputs session key $\mathbf{sk}^i_U$ if it has previously been accepted by $\mathbf{\Pi}^i_U$, 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 the 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 $\mathbf{sk}^i_U$ if $b = 1$, and a random session key otherwise.

Throughout the paper, we assume that all communications in the group key management protocol are authenticated. 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 $\mathbf{\Pi}^i_{U_{x_0}}$ is *fresh* unless either the adversary, at a certain point, queried **Reveal**(U_{x_0}, i) or **Reveal**(U_{x_1}, j) with $(U_{x_1}, j) \in \mathbf{sid}^i_{U_{x_0}}$ or the adversary queried **Corrupt**(U_{x_2}) with $U_{x_2} \in \mathbf{pid}^i_{U_{x_0}}$.

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

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

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

Definition 2. *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 $F^\kappa = \{f_K : GF(p) \times GF(p) \rightarrow GF(p)\}$ 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 $\boldsymbol{C}$ of the hyper-sphere and publishes it to the public. Then each member can calculate $\overline{R}$ via (5) or (6). Therefore, the value $K = (\overline{R} - \|\boldsymbol{C}\|^2) \mod p$ 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.

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 2-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 an 2-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:

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

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

$$b_{20} = f_{a_{10}}(u), b_{21} = f_{a_{11}}(u).$$

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

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

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 2-dimensional space.

Step 4) The GC establishes a hyper-sphere, herein a circle, in 2-dimensional space using the above points $\boldsymbol{B}_0$, $\boldsymbol{B}_1$, and $\boldsymbol{B}_2$. Suppose the central point of the hyper-sphere is $\boldsymbol{C} = (c_0, c_1) \in GF(p)$. 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:

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

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

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

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 $\boldsymbol{C}$ and u to the member U_1 via open channel.

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

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

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, an 1-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.

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 $U_{i_1}, U_{i_2}, \cdots, U_{i_n}$. The steps are as follows.

Step 1) After the new user U_x is authenticated, the GC selects unique 2-dimensional private point $\boldsymbol{A}_x = (a_{x0}, a_{x1}) \in GF(p)^2$ for each new member U_x, where $a_{x0} \neq 0$, $a_{x1} \neq 0$, $a_{x0} \neq a_{x1}$, and $x = (n-m)+1, (n-m)+2, \cdots, n$.

The points $\boldsymbol{A}_x$ should satisfy $\boldsymbol{A}_i \neq \boldsymbol{A}_j$ if $i \neq j$, where $1 \leq i, j \leq n$.

Step 2) The GC sends the point $\boldsymbol{A}_x$ to the user U_x via a secure channel.

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

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

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

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

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

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

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

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

$$\cdots\cdots$$

$$\boldsymbol{B}_{n+1} = (b_{n+1,0}, b_{n+1,1}).$$

If the condition

$$(2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11})) \times \prod_{t=3}^{n+1} (-2b_{t1}) \not\equiv 0 \mod p \quad (11)$$

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 $\boldsymbol{V}_0, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_{n+1}$. Suppose that the central point of the hyper-sphere is $\boldsymbol{C} = (c_0, c_1, \cdots, c_n) \in GF(p)^{n+1}$.

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.,

$$\boldsymbol{V}_0 = (b_{00}, b_{01}, 0, \cdots, 0),$$

$$\boldsymbol{V}_1 = (b_{10}, b_{11}, 0, \cdots, 0),$$

$$\boldsymbol{V}_2 = (b_{20}, b_{21}, 0, \cdots, 0).$$

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

$$\boldsymbol{V}_3 = (b_{30}, 0, b_{31}, 0, \cdots, 0),$$

$$\cdots\cdots$$

$$\boldsymbol{V}_j = (b_{j0}, 0, \cdots, 0, b_{j1}, 0, \cdots, 0),$$

$$\cdots\cdots$$

$$\boldsymbol{V}_{n+1} = (b_{n+1,0}, 0, \cdots, 0, b_{n+1,1}),$$

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 the system of equations about the hyper-sphere by applying the set of points $\boldsymbol{V}_0, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_{n+1}$ to (5) or (6):

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

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

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

Let matrix

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

and vectors

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

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 $\boldsymbol{C} = (c_0, c_1, \cdots, c_n)$ of the hyper-sphere is determined.

Step 5) The GC multicasts $\boldsymbol{C}$ and u to all the group members $U_{i_1}, U_{i_2}, \cdots, U_{i_n}$ via open channel.

Step 6) Each group member U_x can calculate the group key by using its private point $A_x(a_{x0}, a_{x1})$ along with the public information $\boldsymbol{C} = (c_0, c_1, \cdots, c_n)$ and u :

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

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

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 the set of remaining members in the group is $\{U_{i_1}, U_{i_2}, \cdots, U_{i_n}\}$ after removing members. The steps are as follows.

Step 1) The GC deletes the leaving members' private 2-dimensional points.

Step 2) The GC's private 2-dimensional points $\boldsymbol{S}_0$ and $\boldsymbol{S}_1$, and the remaining members' private points $\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2}, \cdots, \boldsymbol{A}_{i_n}$ 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 $\boldsymbol{B}_0, \boldsymbol{B}_1, \cdots, \boldsymbol{B}_{n+1}$ 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.

Massively Adding and Removing Members. This subsection 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 v 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 2-dimensional points, and let new users join in at the same time. After new user U_x is authenticated, the value of x is assigned as the identifier of the new joining members, where $x = (n - m) + 1, (n - m) + 2, \cdots, n$.

The GC selects unique 2-dimensional point $\boldsymbol{A}_x = (a_{x0}, a_{x1}) \in GF(p)$ as U_x's private information, where $a_{x0} \neq 0, a_{x1} \neq 0$, and $a_{x0} \neq a_{x1}$. The private points $\boldsymbol{A}_x$ should satisfy $\boldsymbol{A}_i \neq \boldsymbol{A}_j$ if $i \neq j$, where $1 \leq i, j \leq n$.

Step 2) The GC sends the private point $\boldsymbol{A}_x$ to the user U_x via a secure channel.

The point $\boldsymbol{A}_x$ is the private information of U_x, and should be kept secret by both the member U_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 Step 3 to Step 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 the new group key.

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.

5 Security and Performance Analysis

5.1 Security Analysis

We will show (in Theorem 1) that our group key management protocol is secure, supposed that all communications are authenticated. The proof is similar to the way to prove the security of the unauthenticated protocol by Dutta-Barua[14] and Mikhall et al.[3]. In our security model, the adversary $\mathcal{A}$ can access five oracles, i.e., **Execute**, **Reveal**, **Corrupt**, **Send** and **Test**. The **Send** query may be ignored by $\mathcal{A}$ because all communications are assumed to be authenticated. Some notations, such as F^{κ}, $Func$, p and $GF(p)$, are defined in Section 3.

Theorem 1. *Our protocol is secure under PRF assumption, and the adversary's advantage* $\mathrm{Adv}(t)$ *satisfies the following:*

$$\mathrm{Adv}(t) \leq (2n+4) \times (q_E \times \mathrm{Adv}^{\mathrm{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}),$$

where q_E is the number of **Execute** *queries that the adversary can call and run in time t, and $t^{(1)} = t + O(n^3)M + O(n)H$, in which n is the number of members in the group, M is the average time required to perform multiplication over $GF(p)$, and H is the average time to compute f.*

Proof. Let $\mathcal{A}$ be an adversary for the group key management protocol. By using this, we can construct an algorithm $\mathcal{D}$ that will distinguish between random and pseudo-random functions. Assume that $\mathcal{A}$ will make q_E **Execute** queries, and choose r_{th} session as the **Test** session. And assume that $\mathcal{D}$ correctly guessed the **Test** session r. Then, when $\mathcal{A}$ makes **Execute** query, (except for the r_{th} session), $\mathcal{D}$ follows the real protocol. When $\mathcal{A}$ makes **Reveal** or **Corrupt** oracle (other than for the r_{th} session), $\mathcal{D}$ sends $\mathcal{A}$ all the corresponding information as in a real interaction.

$$\mathbf{Real} := \left\{ \begin{array}{l} u \leftarrow_R GF(p): \\ \quad b_{00} = f_{s_{00}}(u), b_{01} = f_{s_{01}}(u), \\ \quad b_{10} = f_{s_{10}}(u), b_{11} = f_{s_{11}}(u), \\ \quad \text{for} \quad j = 2, 3, \cdots, n+1: \\ \qquad b_{j0} = f_{a_{j-1,0}}(u), b_{j1} = f_{a_{j-1,1}}(u); \\ V_0 = (b_{00}, b_{01}, 0, \cdots, 0), \\ V_1 = (b_{10}, b_{11}, 0, \cdots, 0), \\ V_2 = (b_{20}, b_{21}, 0, \cdots, 0). \\ \text{for} \quad j = 3, 4, \cdots, n+1: \\ \quad V_j = (b_{j0}, 0, \cdots, b_{j1}, 0, \cdots, 0); \\ \boldsymbol{Y} = \begin{bmatrix} 2(b_{00}-b_{10}) & 2(b_{01}-b_{11}) & 0 & ... & 0 \\ 2(b_{10}-b_{20}) & 2(b_{11}-b_{21}) & 0 & ... & 0 \\ 2(b_{20}-b_{30}) & 2b_{21} & -2b_{31} & ... & 0 \\ & & ..... & ... & \\ 2(b_{n0}-b_{n+1,0}) & 0 & ... & ... & -2b_{n+1,1} \end{bmatrix}; \\ \boldsymbol{Z} = \begin{bmatrix} b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 \\ b_{10}^2 + b_{11}^2 - b_{20}^2 - b_{21}^2 \\ b_{20}^2 + b_{21}^2 - b_{30}^2 - b_{31}^2 \\ ... \\ b_{n0}^2 + b_{n1}^2 - b_{n+1,0}^2 - b_{n+1,1}^2 \end{bmatrix}; \\ \boldsymbol{C}^T = (c_0, c_1, \cdots, c_{n+1})^T = \boldsymbol{Y}^{-1} \times \boldsymbol{Z}; \\ \overline{R} = \|\boldsymbol{V}_i - \boldsymbol{C}\|^2; \\ T = \{u; \boldsymbol{C}\} \\ K = \overline{R} - \|\boldsymbol{C}\|^2. \end{array} \right\},$$

$$\mathbf{Fake}^{(0,0)} := \left\{ \begin{array}{l} u \leftarrow_R GF(p): \\ \quad b_{00} = g_{00}(u), b_{01} = f_{s_{01}}(u), \\ \quad b_{10} = f_{s_{10}}(u), b_{11} = f_{s_{11}}(u), \\ \quad \text{for} \quad j = 2, 3, \cdots, n+1: \\ \qquad b_{j0} = f_{a_{j-1,0}}(u), b_{j1} = f_{a_{j-1,1}}(u); \\ \text{the rest are the same as the ones in } \mathbf{Real}. \end{array} \right\}.$$

In the rest of the proof, we will assume that $\mathcal{D}$ correctly guessed the **Test** session. Since such a priori guess is correct with $1/q_E$ chance, this affects the exact security of the reduction proof by a factor of q_E.

As a stepping stone, we first define distributions **Real** and $\mathbf{Fake}^{(0,0)}$ above for transcript/session key pair (T, K) as follows, where **Real** is the real execution scenario of our protocol while $f_{s_{00}}$ is replaced with a truly random function g_{00} in $\mathbf{Fake}^{(0,0)}$. Similarly, we can define the distributions $\mathbf{Fake}^{(0,1)}$, ..., $\mathbf{Fake}^{(n+1,0)}$, $\mathbf{Fake}^{(n+1,1)}$. For $i = 0, 1, \ldots, n+1$, $\mathbf{Fake}^{(i,1)}$ is the same as $\mathbf{Fake}^{(i,0)}$ except that let $b_{i1} = g_{i1}(u)$ where g_{i1} is a truly random function, and $\mathbf{Fake}^{(i+1,0)}$ is the same as $\mathbf{Fake}^{(i,1)}$ except that let $b_{i+1,0} = g_{i+1,0}(u)$ where g_{i0} is a truly random function. Finally, the distribution $\mathbf{Fake}^{(n+1,1)}$ (we

denote as **Fake** hereafter) is described as follows, where for $i = 1, 2, \ldots, n+1$, g_{i0} and g_{i1} are all truly random functions

$$\mathbf{Fake} := \left\{ \begin{array}{l} u \leftarrow_R GF(p); \\ \text{for} \quad i = 0, 1, \cdots, n+1 : \\ \quad b_{i0} = g_{i0}(u), b_{i1} = g_{i1}(u); \\ \text{the rest are the same as the ones in } \mathbf{Real}. \end{array} \right\}.$$

Due to the PRF assumption, we can obtain from Lemma 1 below that

$$\begin{array}{l} |\mathbf{Prob}[(T, K) \leftarrow \mathbf{Real} : \mathcal{A}(T, K) = 1] \\ -\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake}^{(0,0)} : \mathcal{A}(T, K) = 1]| \\ \qquad \leq \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}, \end{array} \tag{1}$$

where t is the running time of $\mathcal{A}$, $t^{(1)} = t + O(n^3)M + O(n)H$, n is the number of members in the group, M is the average time required to perform multiplication over $GF(p)$, and H is the average time to compute f.

Similarly, for $i = 0, 1, \ldots, n+1$, we can further conclude that

$$\begin{array}{l} |\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake}^{(i,0)} : \mathcal{A}(T, K) = 1] \\ -\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake}^{(i,1)} : \mathcal{A}(T, K) = 1]| \\ \qquad \leq \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}, \end{array} \tag{2}$$

and

$$\begin{array}{l} |\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake}^{(i,1)} : \mathcal{A}(T, K) = 1] \\ -\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake}^{(i+1,0)} : \mathcal{A}(T, K) = 1]| \\ \qquad \leq \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}. \end{array} \tag{3}$$

From equation (2) and (3), we have

$$\begin{array}{l} |\mathbf{Prob}[(T, K) \leftarrow \mathbf{Real} : \mathcal{A}(T, K) = 1] \\ -\mathbf{Prob}[(T, K) \leftarrow \mathbf{Fake} : \mathcal{A}(T, K) = 1]| \\ \quad \leq (2n+4)(\mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}). \end{array} \tag{4}$$

Furthermore, from Lemma 2, the success probability of $\mathcal{A}$ in distinguishing between the keys from the distribution **Fake** and keys randomly chosen from $GF(p)$ is just $\frac{1}{2}$. That is,

$$|\mathbf{Prob}[(T, K_0) \leftarrow \mathbf{Fake}; K_1 \leftarrow_R GF(p); b \leftarrow_R \{0, 1\} : \mathcal{A}(T, K_b) = b] = \tfrac{1}{2}. \tag{5}$$

Hence by Lemmas 1 and 2, we can conclude that

$$\begin{array}{l} \quad |\mathbf{Prob}[(T, K_0) \leftarrow \mathbf{Real}, K_1 \leftarrow_R GF(p), b \leftarrow_R \{0, 1\} : \mathcal{A}(T, K_b) = b] - \frac{1}{2}| \\ = |\mathbf{Prob}[(T, K_0) \leftarrow \mathbf{Real}, K_1 \leftarrow_R GF(p), b \leftarrow_R \{0, 1\} : \mathcal{A}(T, K_b) = b] \\ \quad -\mathbf{Prob}[(T, K_0) \leftarrow \mathbf{Fake}, K_1 \leftarrow_R GF(p), b \leftarrow_R \{0, 1\} : \mathcal{A}(T, K_b) = b]| \\ \leq (2n+4) \times (\mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}). \end{array} \tag{6}$$

We assumed that $\mathcal{D}$ correctly guessed the **Test** session above, which affects the exact security of the reduction proof by a factor of q_E. Finally we conclude that the adversary's advantage is negligible under the pseudo-random function assumption,

$$\mathbf{Adv}(t) \leq (2n+4) \times (q_E \times \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \tfrac{1}{p^{p-1}}). \tag{7}$$

Lemma 1. *For any algorithm $\mathcal{A}$ running in time t, we have the following where $t^{(1)} = t + O(n^3)M + O(n)H$:*

$$\begin{aligned} &|\mathbf{Prob}[(T,K) \leftarrow \mathbf{Real} : \mathcal{A}(T,K) = 1] \\ &-\mathbf{Prob}[(T,K) \leftarrow \mathbf{Fake}^{(0,0)} : \mathcal{A}(T,K) = 1]| \\ &\leq \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \tfrac{1}{p^{p-1}}. \end{aligned}$$

Proof. We construct a distinguisher $\mathcal{D}$ by using $\mathcal{A}$ which on an input $g_1 \in Func$. $\mathcal{D}$ first generates a pair (T, K) according to the distribution $Dist'$ described below which depends on g_1, then runs $\mathcal{A}$ on (T, K) and outputs whatever $\mathcal{A}$ outputs.

$$\mathbf{Dist}' := \left\{ \begin{array}{l} u \leftarrow_R GF(p): \\ \quad b_{00} = g_1(u), b_{01} = f_{s_{01}}(u), \\ \quad b_{10} = f_{s_{10}}(u), b_{11} = f_{s_{11}}(u), \\ \quad \text{for} \quad j = 2, 3, \cdots, n+1: \\ \quad\quad b_{j0} = f_{a_{j-1,0}}(u), b_{j1} = f_{a_{j-1,1}}(u); \\ \text{the rest are the same as the ones in } \mathbf{Real}. \end{array} \right\}.$$

Define set $E_1 = \{g \mid g \in Func \backslash F^\kappa\}$. The distribution **Real** and distribution $\{(T, K) : g_1 \in F^\kappa; (T, K) \leftarrow \mathbf{Dist}'(g_1)\}$ are statistically equivalent. On the other hand, the distribution $\mathbf{Fake}_{(00)}$ and the distribution $\{(T, K) : g_1 \in E_1; (T, K) \leftarrow \mathbf{Dist}'(g_1)\}$ are statistically equivalent but for a factor of $\frac{p}{p^p}$ since g_1 is not in F^κ. These two distributions are statistically equivalent by the definition of **PRF**,

$$\begin{aligned} &|\mathbf{Prob}[(T,K) \leftarrow \mathbf{Real} : \mathcal{A}(T,K) = 1] - \mathbf{Prob}[(T,K) \leftarrow \mathbf{Fake}^{(0,0)} : \mathcal{A}(T,K) = 1]| \\ &\leq |\mathbf{Prob}[g_1 \leftarrow_R F^\kappa : \mathcal{D}(g_1) = 1] - \mathbf{Prob}[g_1 \leftarrow_R E_1 : \mathcal{D}(g_1) = 1]| + \frac{|F^\kappa|}{|Func|} \\ &\leq \mathbf{Adv}^{\mathbf{PRF}}_{GF(p)}(t^{(1)}) + \frac{1}{p^{p-1}}. \end{aligned}$$

The time required to perform $n \times n$ matrix inversion and $n \times n$ matrix multiplying an n-dimensional vector operation in $GF(p)$ are $O(n^3)M$ and $O(n^2)M$ respectively. There are $2n+3$ computations of f in $\mathbf{Dist}'$. Hence $t^{(1)}$ is basically equal to $t+O(n^3)M+O(n)H$. □

Lemma 2. *For any computationally unbounded adversary $\mathcal{A}$, we have*

$$\mathbf{Prob}[(T, K_0) \leftarrow \mathbf{Fake}; K_1 \leftarrow_R GF(p); b \leftarrow_R \{0,1\} : \mathcal{A}(T, K_b) = b] = \frac{1}{2}.$$

Proof. We have $T = \{u; \boldsymbol{C}\}$, $K = \overline{R} - \|\boldsymbol{C}\|^2$ and $\boldsymbol{C}^T = (c_0, c_1, \cdots, c_{n+1})^T = \boldsymbol{Y}^{-1} \times \boldsymbol{Z}$. Because **Test** is allowed to call *Fresh* session for only once, no player in this session is corrupted, so $a_{2,0}, a_{2,1}, \cdots, a_{n+1,0}, a_{n+1,1}$ are kept secret and unknown to $\mathcal{A}$. And $\boldsymbol{C} = \boldsymbol{Y}^{-1} \times \boldsymbol{Z}$ is independent from u, where all elements b_{j0}, b_{j1} in both $\boldsymbol{Y}$ and $\boldsymbol{Z}$ are chosen randomly. Thus K is also independent from u and is a random value in $GF(p)$. $\mathcal{A}$ gets no information on both K_0 and K_1, therefore the probability of guessing the bit b correctly is exactly $\frac{1}{2}$.

□

Above we present the static security of our scheme. In the phases of **Adding Members** and **Removing Members**, when new users join the group or members leave the group, GC establishes the new group key as in the phase of **Initialization** by re-selecting new random value $u \in GF(p)$. So both the new users who join the group in **Adding Members** and members who leaves the group in **Removing Members** cannot obtain any information about the previous group key.

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 2-dimensional private point only. The GC should store all members' 2-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 (13) or (14), 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 triangular matrix, the computation complexity of solving $(c_0, c_1, \cdots, c_n)$ from (13) is $O(n)$.

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 2.33GHz Intel Xeon quad-core dual-processor PC server with 4GB memory and running Linux, and the platform for the member is a HP XW4600 Workstation with 2.33GHz Intel dual-processor and 2GB memory and

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

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%. The summary of the experimental results are presented in Table 2 and Table 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 $n = 100000$, 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.

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

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 Table 4 and Table 5.

GKMP (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 with respect to both secrecy and performance.

The LKH (Logical Key Hierarchy) scheme can be considered to be the representative of tree-based schemes, including OFT [37], OFCT [10], Hierarchical α-ary Tree with Clustering [11], Efficient Large-Group Key [29], 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 $O(1)$ in our scheme, but is $O(\log_2 n)$ in LKH; 3) the computation complexity of each member is $O(1)$ in our scheme, but is $O(\log_2 n)$ in LKH.

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 $O(n)$ simple operations, but the one in LKH is $O(2\log_2 n)$ encryptions. In average conditions, the computation of simple operations can be faster than encryptions.

Table 4. Feature and Computation Complexity Comparison among Schemes

	GKMP	LKH	Secure Lock	This Paper
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	This Paper
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	E : average time required by a symmetric encryption
M : average time required by a modular multiplication over $GF(p)$	H : average time required by a hash function
A : average time required by a modular addition over $GF(p)$	R : average time required by a multiplication reverse over $GF(p)$

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 SL (Secure Lock) is based on Chinese Remainder Theorem (CRT), which is a time-consuming operation. Hence, the SL scheme is applicable only for small groups [12].

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 $O(1)$; 3) the computation complexity by each member in both schemes are also $O(1)$; 4) the computation complexity by the GC in both schemes are $O(n)$.

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 [12]. 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: $E \cdot O(n) + R \cdot O(n)$ in the SL, and $2H \cdot O(n) + 3A \cdot O(n)$ 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 over finite field is a time-consuming computation, thus $R \gg 3A$, and then

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

or

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

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

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 can be adopted to weaken 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 (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, 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 (PRF) assumption.

The advantages of our scheme include: (1) the re-keying messages can be broadcasted or multicasted via open channel, and the secure channel is required only once when new users register to join in the group for the first time; (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 $n = 100000$, 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.

Acknowledgement

This paper is financially supported by the National Natural Science Foundation of China under Grant No. U1135004 and 61170080, and Guangdong Province Universities and Colleges Pearl River Scholar Funded Scheme (2011), and Guangzhou Metropolitan Science and Technology Planning Project under grant No. 2011J4300028, and High-level Talents Project of Guangdong Institutions of Higher Education (2012), and the Fundamental Research Funds for the Central Universities under Grant No. 2009ZZ0035 and 2011ZG0015, and Guangdong Provincial Natural Science Foundation of under grant No. 9351064101000003.

References

1. Y. Amir, C. Nita-Rotaru, S. Stanton, and G. Tsudik, "Secure spread: an integrated architecture for secure group communication," IEEE Transactions on Dependable and Secure Computing, vol. 2, no. 3, pp. 248-261, 2005.
2. Y. Amir, Y. Kim, C. Nita-Rotaru, J. L. Schultz, J. Stanton, and G. Tsudik, "Secure group communication using robust contributory key agreement," IEEE Transactions on Parallel and Distributed Systems, vol. 15, no.5, pp. 468-480, 2004.
3. M. Atallah, M. Blanton, N. Fazio, and K. Frikken, "Dynamic and efficient key management for access hierarchies," ACM Trans. Inf. Syst. Secur., vol. 12, no. 3, pp. 1-43, 2009.
4. A. Ballardie, "Scalable multicast key distribution," RFC 1949, 1996.
5. C. Becker, U. Wille, "Communication complexity of group key distribution," In Proceedings of the 5th ACM Conference on Computer and Communications Security, San Francisco, Calif., ACM, New York, pp. 1-6, Nov. 1998.
6. M. Bellare, R. Canetti, and H. Krawczyk, "Pseudorandom functions revisited; the cascade construction and its concrete security," In Proceedings of the 37th Annual Symposium on Foundations of Computer Science. IEEE Computer Society, pp. 514 - 523, 1996.
7. C. Boyd, "On key agreement and conference key agreement," In Proceedings of the Information Security and Privacy: Australasian Conference, Lecture Notes in Computer Science, vol. 1270. Springer-Verlag, New York, pp. 294-302, 1997.
8. B. Briscoe, "MARKS: Multicast key management using arbitrarily revealed key sequences," In Proceedings of the 1st International Workshop on Networked Group Communication, Pisa, Italy, pp. 301-320, Nov. 1999.
9. M. Burmester, and Y. Desmedt, "A secure and efficient conference key distribution system (extended abstract)," In Advances in Cryptology-EUROCRYPT 94, A. D. Santis, Ed., Lecture Notes in Computer Science, vol. 950. Springer- Verlag, New York, pp. 275-286, 1994.
10. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," In Proceedings of the IEEE INFOCOM, New Yok, pp. 708-716 , Mar. 1999.
11. R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," In Advances in Cryptology-EUROCRYPT 1999, Lecture Notes in Computer Science, vol. 1599, Springer- Verlag, New York, pp. 459-474, 1999.
12. G.-H. Chiou, and W.-T. Chen, "Secure broadcasting using the secure lock," IEEE Transactions on Software Engineering, vol. 15, no. 8, pp. 929-934, Aug. 1989.
13. B. Decleene, L. Dondeti, S. Griffin, T. Hardjono, D. Kiwior, J. Kurose, D. Towsley, S. Vasudevan, and C. Zhand, "Secure group communications for wireless networks," In Proceedings of the MILCOM, pp. 113 - 117, June 2001.
14. R. Dutta, and R. Barua, "Provably secure constant round contributory group key agreement in dynamic setting," IEEE Transactions On Information Theory, vol. 54, no. 5, pp. 2007-2025, May 2008.
15. L. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," Comput. Commun., vol. 23, no. 17, pp. 1681-1701, Nov. 1999.
16. L. Dondeti, S. Mukherjee, and A. Samal, "A distributed group key management scheme for secure many-to-many communication," Tech. Rep. PINTL-TR-207-99, Department of Computer Science, University of Maryland, 1999.
17. S. Goldwasser, and M. Bellare, "Lecture notes on cryptography," Summer cource cryptography and computer security at MIT, 2008.
18. Q. Gu, P. Liu, W. C. Lee, and C. H. Chu, "KTR: An efficient key management scheme for secure data sccess control in wireless broadcast services," IEEE Transactions on Dependable and Secure Computing, vol. 6, no. 3, pp. 188-201, 2009.

19. H. Harney, C. Muckenhirn, and T. Rivers, "Group key management protocol (GKMP) specification," RFC 2093, July 1997. [Online]. Available: http://tools.ietf.org/rfc/rfc2093.txt
20. H. Harney, C. Muckenhirn, and T. Rivers, "Group key management protocol (GKMP) architecture," RFC 2094, July 1997. [Online]. Available: http://tools.ietf.org/rfc/rfc2094.txt.
21. Y. Kim, A. Perrig, and G. Tsudik, "Simple and fault-tolerant key agreement for dynamic collaborative groups," In Proceedings of the 7th ACM conference on Computer and Communications security, pp. 235-244, 2000.
22. Y. Kim, M. Narasimha, and G. Tsudik, "Secure group key management for storage area networks," IEEE Communications Magazine, vol. 41, no. 8, pp. 92-99, 2003.
23. Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," IEEE Transactions on Computers, vol. 53, no. 5, pp. 905-921, 2004.
24. 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 Transactions on Networking, vol. 14, no. 2, pp. 263-276, 2006.
25. 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 Transactions on Networking, vol. 14, no. 5, pp. 1128-1140, Oct. 2006.
26. S. Mittra, "Iolus: A framework for scalable secure multicasting," In Proceedings of the ACM SIGCOMM, ACM, New York, vol. 27, no. 4, pp. 277-288, Sept. 1997.
27. R. Molva, and A. Pannetrat, "Scalable multicast security in dynamic groups," In Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, ACM, New York, 101-112, Nov. 1999.
28. A. Perrig, "Efficient collaborative key management protocols for secure autonomous group communication," In Proceedings of the International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC'99), Hong Kong, China, M. Blum and C H Lee, Eds. City University of Hong Kong Press, Hong Kong, China, pp. 192-202, July 1999.
29. A. Perrig, D. Song, and J. Tygar, "ELK, a new protocol for efficient large-group key distribution," In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, pp. 247-262, May 2001.
30. S. Rafaeli, and D. Hutchison, "A survey of key management for secure group communication," ACM Comput.Surv., vol. 35, no. 3, pp. 309-329, Sept. 2003.
31. S. Rafaeli, and D. Hutchison, "Hydra: A decentralised group key management," In Proceedings of the 11th IEEE International WETICE: Enterprise Security Workshop, A. Jacobs, Ed., Pittsburgh, Pa., IEEE Computer Society Press, Los Alamitos, Calif, pp. 62-67, June 2002.
32. K. Ren, W. Lou, B. Zhu and S. Jajodia, "Secure and efficient multicast in wireless sensor networks allowing Ad hoc group formation," IEEE Transactions on Vehicular Technology, vol. 58, no. 4, pp. 2018-2029, 2009.
33. B. Rong, H. Chen, Y. Qian, K. Lu, R. Hu, and S. Guizani, "A pyramidal security model for large-scale group-oriented computing in mobile Ad hoc networks: The key management study," IEEE Transactions on Vehicular Technology, vol. 58, no. 1, pp. 398-408, January 2009.
34. O. Rodeh, K. Birman, and D. Dolev, "Optimized group rekey for group communication systems," In Network and Distributed System Security, San Diego, Calif., Feb. 2000.
35. J. Salido, L. Lazos, and R. Poovendran, "Energy and bandwidth-efficient key distribution in wireless Ad hoc networks: a cross-layer approach," IEEE/ACM Transactions on Networking, vol. 15, no. 6, pp. 1527-1540, 2007.
36. S. Setia, S. Koussih, and S. Jajodia, "Kronos: A scalable group re-keying approach for secure multicast," In Proceedings of the IEEE Symposium on Security and Privacy, Oakland Calif., IEEE Computer Society Press, Los Alamitos, Calif, pp. 215-228, May 2000.

37. A.T. Sherman and D.A McGrew, "Key establishment in large dynamic groups using one-way function trees," IEEE Transactions on Software Engineering, vol. 29, no. 5, pp. 444-458, May 2003.
38. M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," In SIGSAC Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, ACM, New York, pp. 31-37, Mar. 1996.
39. Y. Sun, W. Trappe, and K. J. R. Liu, "A scalable multicast key management scheme for heterogeneous wireless networks," IEEE/ACM Transactions on Networking, vol. 12, no. 4, pp. 653-666, Aug. 2004.
40. M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner, "The VersaKey framework: Versatile group key management," IEEE J. Sel. Areas Commun. , vol. 17, no. 9, pp. 1614-1631, Sept. 1999.
41. C.K. Wong, M. Gouda, and S.S.Lam, "Secure group communications using key graphs," IEEE/ACM Transactions on Networking, vol. 8, no. 1, pp 16-30, Feb. 2000.
42. Q.H. Wu, Y. Mu, W. Susilo, B. Qin, and J. Domingo-Ferrer, "Asymmetric group key agreement," In Advances in Cryptology-EUROCRYPT 2009, Antoine Joux , Lecture Notes in Computer Science, vol. 5479. Springer- Verlag, Heidelberg, pp. 153-170, 1994.
43. Q.H. Wu, B. Qin, L. Zhang, J. Domingo-Ferre, and O. Farrs, "Bridging broadcast encryption and group key agreementm," In Advances in Cryptology-ASIACRYPT 2011, D. Lee and X.Y. Wang, Lecture Notes in Computer Science, vol. 7073. Springer- Verlag, Heidelberg, pp. 143-160, 2011.
44. X. Yi, C. K. Siew, C. H. Tan, and Y. Ye, "A secure conference scheme for mobile communications," IEEE Transactions on Wireless Communications, vol. 2, no. 6, pp. 1168-1177, 2003.
45. X. Yi, C. K. Siew, and C. H. Tan, "A secure and efficient conference scheme for mobile communications," IEEE Transactions on Vehicular Technology, vol. 52, no. 4, pp. 784-793, 2003.
46. W. Yu, Y. Sun, and K. J. R. Liu, "Optimizing rekeying cost for contributory group key agreement schemes," IEEE Transactions On Dependable and Secure Computing, vol. 4, no. 3, pp. 228-242, 2007.

A Toy Example

A toy example is given to illustrate the procedure of massive membership change in our group key management approach based upon hyper-sphere.

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

At the initiazation stage, the GC selects two different 2-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.

Now suppose the set of members in the current group is $\{U_1, U_2, U_3, U_4\}$. The members U_2 and U_4 want to leave the group, and new users U_5 and U_6 want to join the group. The following steps can support massively adding and removing of members.

Step 1) The GC deletes the private points $\boldsymbol{A}_2 = (a_{20}, a_{21})$ and $\boldsymbol{A}_4 = (a_{40}, a_{41})$ of the leaving members.

After the new users U_5 and U_6 are authenticated, the GC assigns ID=5 and ID=6 to the new members U_5 and U_6 respectively.

The GC selects unique 2-dimensional points $\boldsymbol{A}_5 = (a_{50}, a_{51})$ and $\boldsymbol{A}_6 = (a_{60}, a_{61})$ as the private information of U_5 and U_6 respectively.

Now the set of private points of the group members is $\{\boldsymbol{A}_1, \boldsymbol{A}_3, \boldsymbol{A}_5, \boldsymbol{A}_6\}$, and the subscripts of the private points are: $i_1 = 1, i_2 = 3, i_3 = 5$, and $i_4 = 6$. The points $\boldsymbol{A}_x$ should also satisfy $\boldsymbol{A}_y \neq \boldsymbol{A}_z$ if $y \neq z$, where $y, z \in \{1, 3, 5, 6\}$.

Step 2) The GC sends the point $\boldsymbol{A}_x$ to the member U_x via a secure channel, where $x \in \{5, 6\}$.

Step 3) The GC chooses a random number u, and computes:

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

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

$$b_{20} = f_{a_{i_1,0}}(u) = f_{a_{10}}(u), b_{21} = f_{a_{i_1,1}}(u) = f_{a_{11}}(u),$$

$$b_{30} = f_{a_{i_2,0}}(u) = f_{a_{30}}(u), b_{31} = f_{a_{i_2,1}}(u) = f_{a_{31}}(u),$$

$$b_{40} = f_{a_{i_3,0}}(u) = f_{a_{50}}(u), b_{41} = f_{a_{i_3,1}}(u) = f_{a_{51}}(u),$$

$$b_{50} = f_{a_{i_4,0}}(u) = f_{a_{60}}(u), b_{51} = f_{a_{i_4,1}}(u) = f_{a_{61}}(u).$$

The GC then constructs points $\boldsymbol{B}_0, \boldsymbol{B}_1, \cdots, \boldsymbol{B}_5$:

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

$$\boldsymbol{B}_2 = (b_{20}, b_{21}), \boldsymbol{B}_3 = (b_{30}, b_{31}),$$

$$\boldsymbol{B}_4 = (b_{40}, b_{41}), \boldsymbol{B}_5 = (b_{50}, b_{51}).$$

If the condition

$$(2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11})) \times \prod_{t=3}^{5} (-2b_{t1}) \not\equiv 0 \mod p \quad (16)$$

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

Step 4) The GC expands $\boldsymbol{B}_0, \boldsymbol{B}_1, \boldsymbol{B}_2, \boldsymbol{B}_3, \boldsymbol{B}_4$, and $\boldsymbol{B}_5$ to become 5-dimensional points:

$$\boldsymbol{V}_0 = (b_{00}, b_{01}, 0, 0, 0),$$

$$\boldsymbol{V}_1 = (b_{10}, b_{11}, 0, 0, 0),$$

$$\boldsymbol{V}_2 = (b_{20}, b_{21}, 0, 0, 0),$$

$$\boldsymbol{V}_3 = (b_{30}, 0, b_{31}, 0, 0),$$

$$\boldsymbol{V}_4 = (b_{40}, 0, 0, b_{41}, 0),$$

$$\boldsymbol{V}_5 = (b_{50}, 0, 0, 0, b_{51}).$$

The GC is now going to establishe a 4-dimensional hyper-sphere based on the set of points $\boldsymbol{V}_0, \boldsymbol{V}_1, \cdots, \boldsymbol{V}_5$. Suppose the central point of the hyper-sphere is $\boldsymbol{C} = (c_0, c_1, \cdots, c_4)$. The GC then constructs the set of equations about the hyper-sphere:

$$\begin{cases} (b_{00}-c_0)^2+(b_{01}-c_1)^2+(0-c_2)^2+(0-c_3)^2+(0-c_4)^2 \equiv \overline{R} \mod p, \\ (b_{10}-c_0)^2+(b_{11}-c_1)^2+(0-c_2)^2+(0-c_3)^2+(0-c_4)^2 \equiv \overline{R} \mod p, \\ (b_{20}-c_0)^2+(b_{21}-c_1)^2+(0-c_2)^2+(0-c_3)^2+(0-c_4)^2 \equiv \overline{R} \mod p, \\ (b_{30}-c_0)^2+(0-c_1)^2+(b_{31}-c_2)^2+(0-c_3)^2+(0-c_4)^2 \equiv \overline{R} \mod p, \\ (b_{40}-c_0)^2+(0-c_1)^2+(0-c_2)^2+(b_{41}-c_3)^2+(0-c_4)^2 \equiv \overline{R} \mod p, \\ (b_{50}-c_0)^2+(0-c_1)^2+(0-c_2)^2+(0-c_3)^2+(b_{51}-c_4)^2 \equiv \overline{R} \mod p. \end{cases} \tag{17}$$

Let matrix

$$\boldsymbol{Y} = \begin{bmatrix} 2(b_{00}-b_{10}) & 2(b_{01}-b_{11}) & 0 & 0 & 0 \\ 2(b_{10}-b_{20}) & 2(b_{11}-b_{21}) & 0 & 0 & 0 \\ 2(b_{20}-b_{30}) & 2b_{21} & -2b_{31} & 0 & 0 \\ 2(b_{30}-b_{40}) & 0 & 2b_{31} & -2b_{41} & 0 \\ 2(b_{40}-b_{50}) & 0 & 0 & 2b_{41} & -2b_{51} \end{bmatrix}$$

and vectors

$$\boldsymbol{C}^T = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}, \quad \boldsymbol{Z} = \begin{bmatrix} b_{00}^2+b_{01}^2-b_{10}^2-b_{11}^2 \\ b_{10}^2+b_{11}^2-b_{20}^2-b_{21}^2 \\ b_{20}^2+b_{21}^2-b_{30}^2-b_{31}^2 \\ b_{30}^2+b_{31}^2-b_{40}^2-b_{41}^2 \\ b_{40}^2+b_{41}^2-b_{50}^2-b_{51}^2 \end{bmatrix}.$$

By subtracting the j-th equation from the $(j+1)$-th equation in (17), where $j = 1, 2, \cdots, 5$, we can get a system of linear equations with 5 unknowns $c_0, c_1, \cdots, c_4$, which can be expressed in the matrix and vector form

$$\boldsymbol{Y} \times \boldsymbol{C}^T = \boldsymbol{Z} \quad . \tag{18}$$

The condition in (16) in Step 3 guarantees that (18) has one and only one solution $\boldsymbol{C}^T = \boldsymbol{Y}^{-1} \times \boldsymbol{Z}$. Then the central point $\boldsymbol{C} = (c_0, c_1, \cdots, c_4)$ of the hyper-sphere is determined.

Step 5) The GC multicasts $\boldsymbol{C}$ and u to all group members U_1, U_3, U_5, and U_6 via open channel.

Step 6) Each group member can calculate the new group key.

The member U_1 can calculate the group key by using its private point $\boldsymbol{A}_1 = (a_{10}, a_{11})$ along with the public information $\boldsymbol{C} = (c_0, c_1, \cdots, c_4)$ and u, and the third equation in (17):

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

Similarly, the member U_3 can calculate the group key by using its private point $A_3(a_{30}, a_{31})$ along with the public information $\boldsymbol{C} = (c_0, c_1, \cdots, c_4)$ and u, and the forth equation in (17):

$$K = \overline{R} - \|\boldsymbol{C}\|^2 = b_{30}^2 + b_{31}^2 - 2b_{30}c_0 - 2b_{31}c_2$$

$$= (f_{a_{30}}(u))^2 + (f_{a_{31}}(u))^2 - 2f_{a_{30}}(u)c_0 - 2f_{a_{31}}(u)c_2.$$

For users U_5 and U_6, the computation procedures are similar to that of members U_1 and U_3. Finally, all the group members can re-construct the same hyper-sphere and calculate the same group key $K = \overline{R} - \|\boldsymbol{C}\|^2$.