Secure Group Key Management Approach Based upon N-dimensional Hyper-sphere

Shaohua Tang[1,2], 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] Department of Applied Mathematics, South China University of Technology, China

Abstract. Secure group communication systems become more and more important in many emerging network applications. For a secure group communication system, an efficient and robust group key management approach is essential. In this paper, a new group key management approach with a group controller GC using the theory of hyper-sphere is developed, where a hyper-sphere is constructed for the group and each member in the group corresponds to a point on the hyper-sphere, which can be called the member's private point. The GC computes the central point of the hyper-sphere, whose distance from each member's private point is identical. The central point is published and each member can compute a common group key via the square of the radius. The security relies on the fact that any illegitimate user cannot calculate this value without the legitimate vector, therefore cannot derive the group key. This approach is secure and its backward and forward secrecy can be guaranteed. The performance of our approach is analyzed to demonstrate its advantages in comparison with others, which include: 1) it requires both small memory and little computations for each group member; 2) it can handle massive membership change efficiently with only two re-keying messages, i.e., the central point of the hyper-sphere and a random number; 3) it is very efficient and very scalable for large size groups. Our experiments confirm these advantages and the implementation of our prototype presents very satisfactory performance for large size groups.

Keywords: Group communication, key management, hyper-sphere, 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 also 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 own perfect scalability. It is shown that a secure, efficient, and robust group key management approach is essential to a secure group communication system.

A new secure group key management approach based on the properties of N-dimensional hyper-sphere is proposed in this paper. In mathematics, N-dimensional hyper-sphere or a N-sphere, is a generalization of the surface of an ordinary sphere to arbitrary dimension. In particular, a 0-sphere is a pair of points on a line, a 1-sphere is a circle in the plane, and a 2-sphere is an ordinary sphere in three-dimensional space. Spheres of dimension $N > 2$ are sometimes called hyper-spheres. For any natural number N, a N-sphere of radius R is defined as the set of points in $(N + 1)$-dimensional Euclidean space which are at distance R from a central point. Based on the above mathematical principle, a secure group key management scheme is designed. The advantages of the proposed approach include 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 previous used and future keys. The backward and forward secrecy can be guaranteed.

The rest of this paper is organized as follows. Some related schemes on secure group key management are described in Section II. The proposed secure group key management approach and a toy example is presented in Section III. Security and performance analysis is discussed in Section IV. Finally, Section V summarizes the major contributions of this paper.

2 A Brief Survey of Related Work

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

In a centralized system, there is one entity GC (Group Controller) controlling the whole group [1]. Some typical schemes in this category include Group Key Management Protocol (GKMP)[2], [3], Secure Lock (SL)[4], Logical Key Hierarchy (LKH)[5], etc. The Group Key Management Protocol (GKMP) [2], [3] 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 $K' = E_{K_N}(K_0)$ then broadcasts K' to the group. 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, and there is no solution for keeping the forward secrecy property when a member leaves the group except to recreate an entirely new group without that member [1]. The Secure Lock (SL)

scheme [4] takes advantage of Chinese Remainder Theorem (CRT) to construct a secure lock to combine all the re-keying messages into one while the group key is updated. However, CRT is a time-consuming operation. As mentioned in [4], 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 [5] 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 changed. 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 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) [6], OFCT (One-way Function Chain Tree) [7], Hierarchical α-ary Tree with Clustering [8], Efficient Large-Group Key [9], etc. They can be regarded as the similarity or improvement of LKH.

In the distributed architectures, 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[12], Group Diffie-Hellman key exchange[13], Octopus Protocol[14], Conference Key Agreement[15], Distributed Logical Key Hierarchy[16], Distributed One-way Function Tree[17], Diffie-Hellman Logical Key Hierarchy[18], [10], Distributed Flat Table [19], etc. Recent references paid more attentions to contributory and collaborative group key agreement, for example: [20], [21], [22], [23], [24], [25], etc.

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[26], Iolus[27], Dual-Encryption Protocol[28], MARKS[29], Cipher Sequences[30], Kronos[31], Intra-Domain Group Key Management[32], Hydra[33], etc.

The secure group key management approaches can be applied to a lot of application areas. For example: wireless/mobile network[34], [35], [37], [38], [39], [40], wireless sensor network[36], storage area networks[41], etc.

3 Proposed Scheme Based on Hyper-Sphere

3.1 Background Knowledge

In mathematics, a N-sphere of radius R is defined as the set of points in $(N+1)$-dimensional Euclidean space which are at distance R from a central point. Any point $X(x_0, x_1, \ldots, x_N)$ on the hyper-sphere can be represented by the equation

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

where $C(c_0, c_1, \ldots, c_N)$ is the central point, and R is the radius.

Any given $(N+2)$ points $A_i(a_{i,0}, a_{i,1}, \ldots, a_{i,N})$ in $(N+1)$-dimensional space, 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 $A_0, A_1, \ldots, A_{N+1}$ 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.

3.2 Notation

The following notations are used throughout the remainder of this paper:

GC is the group controller, who controls the whole group and manages the group initialization, membership change operations.

p is a public large prime number chosen by the GC.

$GF(p)$ is the finite field of order p.

n is the number of members in the group.

h is a cryptographic hash function over $GF(p)$.

It is supposed that all the computations hereafter are over the finite field $GF(p)$, and p, n and h are public.

3.3 The Proposed Approach

Based on the mathematical principle that any point on the hyper-sphere is at the same distance R from the central point, a new secure group key management scheme is proposed. As illustrated in Fig. 1, a hyper-sphere is constructed for the group, and each member in the group corresponds to a point on the hyper-sphere and the point can be called the member's private point. The GC computes

the central point $C(c_0, c_1, \ldots, c_N)$ of the hyper-sphere and publishes it to the public. Then each member can calculate the square of the radius R^2 via (1). The value of $\|C\|^2 = ((c_0^2 + c_1^2 + \ldots + c_N^2) \mod p)$ and $((R^2 - \|C\|^2) \mod p)$ can also be computed by each member. Therefore, $K = ((R^2 - \|C\|^2) \mod p)$ can be assigned as the secret group key. Any illegitimate user cannot calculate this value without the knowledge of the legitimate private point, therefore cannot derive the group key.

Similar to other group key management scheme, our approach include some phases like initialization, adding members, removing members, massive adding and removing members simultaneously, and periodically group key update.

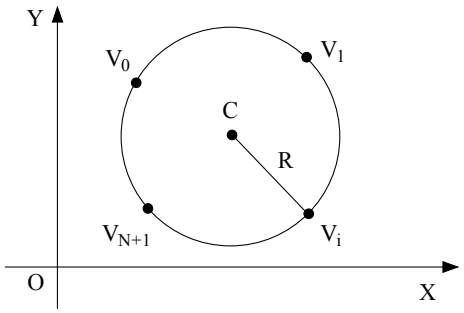

Fig. 1. Each member corresponds to a point V_i on the hyper-sphere.

Initialization The GC lets 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 $S_0(s_{00}, s_{01})$ and $S_1(s_{10}, s_{11})$, and keeps them secret. A large prime number p and a secure hash function h are also chosen and published to the public by the GC.

Step 2) After authenticating U_1, the GC chooses a 2-dimensional private point $A_1(a_{10}, a_{11})$ 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 $A_1(a_{10}, a_{11})$ securely and transmits A_1 to the user U_1 via a secure channel.

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 and computes:

$$b_{00} = h(h(s_{00}) \oplus u),$$

$$b_{01} = h(h(s_{01}) \oplus u),$$

$$b_{10} = h(h(s_{10}) \oplus u),$$

$$b_{11} = h(h(s_{11}) \oplus u),$$

$$b_{20} = h(h(a_{10}) \oplus u),$$

$$b_{21} = h(h(a_{11}) \oplus u).$$

Then the GC constructs new points B_0, B_1, and B_2:

$$B_0 = (b_{00}, b_{01}),$$

$$B_1 = (b_{10}, b_{11}),$$

$$B_2 = (b_{20}, b_{21}).$$

If

$$2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11}) \neq 0, \tag{4}$$

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

Note that the condition in (4) can guarantee that the points B_0, B_1, and 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 B_0, B_1, and B_2. Suppose the central point of the hyper-sphere is $C(c_0, c_1)$, and the square of the radius is R^2. The GC constructs the following system of equations:

$$\begin{cases} (b_{00} - c_0)^2 + (b_{01} - c_1)^2 = R^2 \\ (b_{10} - c_0)^2 + (b_{11} - c_1)^2 = R^2 \\ (b_{20} - c_0)^2 + (b_{21} - c_1)^2 = R^2 \end{cases} \tag{5}$$

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:

$$\begin{cases} 2(b_{00} - b_{10})c_0 + 2(b_{01} - b_{11})c_1 = b_{00}^2 + b_{01}^2 - b_{10}^2 - b_{11}^2 \\ 2(b_{10} - b_{20})c_0 + 2(b_{11} - b_{21})c_1 = b_{10}^2 + b_{11}^2 - b_{20}^2 - b_{21}^2 \end{cases} \tag{6}$$

The condition in (4) guarantees that (6) has one and only one solution (c_0, c_1). Then the central point $C(c_0, c_1)$ of the hyper-sphere is determined. Fig. 2 illustrates the hyper-sphere established by the GC.

Step 5) The GC delivers $C(c_0, c_1)$ 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 $A_1(a_{10}, a_{11})$ along with the public information $C(c_0, c_1)$ and u:

$$\begin{aligned} K = R^2 - \|C\|^2 &= b_{20}^2 + b_{21}^2 - 2b_{20}c_0 - 2b_{21}c_1 \\ &= (h(h(a_{10}) \oplus u))^2 + (h(h(a_{11}) \oplus u))^2 \\ &\quad -2h(h(a_{10}) \oplus u)c_0 - 2h(h(a_{11}) \oplus u)c_1, \end{aligned} \tag{7}$$

where R^2 is the square of the radius, C is the central point of the hyper-sphere, and $\|C\|^2 = c_0^2 + c_1^2$.

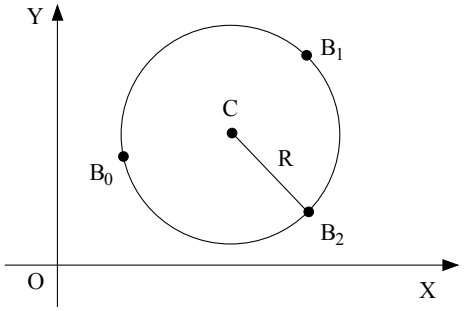

Fig. 2. A hyper-sphere established by the GC.

Adding Members Suppose there are $n - m$ members in the group, where $n > 2$ and $n > m \geq 0$, and there are m new members want to join the group. After new members are admitted, there will be n members in the group, and suppose that the set of members in the group is $\{U_{i_1}, U_{i_2}, \cdots, U_{i_n}\}$. The steps are as follows.

Step 1) After the new user U_x is authenticated, the GC selects unique 2-dimensional private point $A_x(a_{x0}, a_{x1})$ 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 A_x should satisfy $A_i \neq A_j$ if $i \neq j$, where $1 \leq i, j \leq n$.

Step 2) The GC sends the point A_x to the user U_x via a secure channel.

The point 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, and computes

$$b_{00} = h(h(s_{00}) \oplus u),$$

$$b_{01} = h(h(s_{01}) \oplus u),$$

$$b_{10} = h(h(s_{10}) \oplus u),$$

$$b_{11} = h(h(s_{11}) \oplus u).$$

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

$$b_{j0} = h(h(a_{i_{j-1},0}) \oplus u),$$

$$b_{j1} = h(h(a_{i_{j-1},0}) \oplus u).$$

Then the GC constructs new points $B_0, B_1, \cdots, B_{n+1}$:

$$B_0 = (b_{00}, b_{01}),$$

$$B_1 = (b_{10}, b_{11}),$$

$$B_2 = (b_{20}, b_{21}),$$

$$\cdots\cdots$$

$$B_{n+1} = (b_{n+1,0}, b_{n+1,1}).$$

If the condition

$$(2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11})) \times \prod_{t=3}^{n+1} (-2b_{t1}) \neq 0 \quad (8)$$

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

Step 4) The GC expands each B_j to become a $(n+1)$-dimensional point V_j. Then the GC constructs a n-dimensional hyper-sphere based on the set of points $V_0, V_1, \cdots, V_{n+1}$. Suppose the central point of the hyper-sphere is $C(c_0, c_1, \cdots, c_n)$, and the square of the radius is R^2.

Step 4.1) The GC expands each B_j to become a $(n+1)$-dimensional point V_j .

For $j = 0, 1$, and 2, the point B_j is supplemented $(n-1)$ zeros to become V_j, i.e.,

$$V_0 = (b_{00}, b_{01}, 0, \cdots, 0),$$
$$V_1 = (b_{10}, b_{11}, 0, \cdots, 0),$$
$$V_2 = (b_{20}, b_{21}, 0, \cdots, 0).$$

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

$$V_3 = (b_{30}, 0, b_{31}, 0, \cdots, 0),$$
$$\cdots\cdots$$
$$V_j = (b_{j0}, 0, \cdots, 0, b_{j1}, 0, \cdots, 0),$$
$$\cdots\cdots$$
$$V_{n+1} = (b_{n+1,0}, 0, \cdots, 0, b_{n+1,1}),$$

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 $V_0, V_1, \cdots, V_{n+1}$ to (1):

$$\begin{cases} (b_{00} - c_0)^2 + (b_{01} - c_1)^2 + (0 - c_2)^2 + \cdots + (0 - c_n)^2 = R^2 \\ (b_{10} - c_0)^2 + (b_{11} - c_1)^2 + (0 - c_2)^2 + \cdots + (0 - c_n)^2 = R^2 \\ (b_{20} - c_0)^2 + (b_{21} - c_1)^2 + (0 - c_2)^2 + \cdots + (0 - c_n)^2 = R^2 \\ (b_{30} - c_0)^2 + (0 - c_1)^2 + (b_{31} - c_2)^2 + \cdots + (0 - c_n)^2 = R^2 \\ \cdots\cdots \\ (b_{n+1,0} - c_0)^2 + (0 - c_1)^2 + (0 - c_2)^2 + \cdots + (b_{n+1,1} - c_n)^2 = R^2 \end{cases} \quad (9)$$

Let matrix

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

and vectors

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

By subtracting the j-th equation from the $(j+1)$-th equation in (9), where $j = 1, 2, \cdots, n$, we can get a system of linear equations with $(n+1)$ unknowns using the matrix and vector expression

$$Y \times X = Z \tag{10}$$

The condition in (8) guarantees that (10) has one and only one solution $(c_0, c_1, \cdots, c_n)$. Then the central point $C(c_0, c_1, \cdots, c_n)$ of the hyper-sphere is determined.

Step 5) The GC multicasts $C(c_0, c_1, \cdots, c_n)$ 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 $C(c_0, c_1, \cdots, c_n)$ and u :

$$\begin{aligned} K = R^2 - \|C\|^2 &= b_{x+1,0}^2 + b_{x+1,1}^2 - 2b_{x+1,0}c_0 - 2b_{x+1,1}c_{i_x} \\ &= (h(h(a_{i_x,0}) \oplus u))^2 + (h(h(a_{i_x,1}) \oplus u))^2 \\ &\quad -2h(h(a_{i_x,0}) \oplus u)c_0 - 2h(h(a_{i_x,1}) \oplus u)c_{i_x}, \end{aligned} \tag{11}$$

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

Removing Members Suppose that there are $(n+f)$ members in the group, where $n > 2$ and $f \geq 0$, and there are f members want to leave the group, then there will be n members in the group after f users leave. Suppose the set of remaining members in the group is $\{U_{i_1}, U_{i_2}, \cdots, U_{i_n}\}$ after members leave. 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 S_0 and S_1, and the remaining members' private points $A_{i_1}, A_{i_2}, \cdots, A_{i_n}$ 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 and constructs new points $B_0, B_1, \cdots, B_{n+1}$ in Step 3. Then the GC constructs a new hyper-sphere in Step 4, and publishes u and the central point of the hyper-sphere in Step 5. Finally, each group member can calculate the group key by using it private point in Step 6.

Massive 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 $n + w - v$ members in the group, where $n > 2$ and $w \geq 0$, $n + w > v \geq 0$. There are w members want to leave, and v new members want to join the group simultaneously. After the membership changes, 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 $x = (n - v) + 1, (n - v) + 2, \cdots, n$.

The GC selects unique 2-dimensional point $A_x(a_{x0}, a_{x1})$ as U_x's private information, where $a_{x0} \neq 0, a_{x1} \neq 0$, and $a_{x0} \neq a_{x1}$. The private points A_x should satisfy $A_i \neq A_j$ if $i \neq j$, where $1 \leq i, j \leq n$.

Step 2) The GC sends the private point A_x to the user U_x via a secure channel.

The point 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, constructs a new hyper-sphere, and publishes u and the central point of the hyper-sphere. Then each group member can calculate the 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, then construct a new hyper-sphere, and publish u and the central point of the hyper-sphere. These steps are the same as Steps 3 - 6 in "Adding Members" phase.

3.4 A Toy Example

A toy example is given to illustrate the procedure of massive membership change. 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 massive adding and removing of members.

Step 1) The GC deletes the private points $A_2(a_{20}, a_{21})$ and $A_4(a_{40}, a_{41})$ 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 $A_5(a_{50}, a_{51})$ and $A_6(a_{60}, a_{61})$ as the private information of U_5 and U_6 respectively.

Now the set of private points of the group members is $\{A_1, A_3, A_5, A_6\}$, and the subscripts of the private points are: $i_1 = 1, i_2 = 3, i_3 = 5$, and $i_4 = 6$. The points A_x should also satisfy $A_y \neq A_z$ if $y \neq z$, where $y, z \in \{1, 3, 5, 6\}$.

Step 2) The GC sends the point 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:

$$b_{00} = h(h(s_{00}) \oplus u),$$

$$b_{01} = h(h(s_{01}) \oplus u),$$

$$b_{10} = h(h(s_{10}) \oplus u),$$

$$b_{11} = h(h(s_{11}) \oplus u),$$

$$b_{20} = h(h(a_{i_1,0}) \oplus u) = h(h(a_{10}) \oplus u),$$

$$b_{21} = h(h(a_{i_1,1}) \oplus u) = h(h(a_{11}) \oplus u),$$

$$b_{30} = h(h(a_{i_2,0}) \oplus u) = h(h(a_{30}) \oplus u),$$

$$b_{31} = h(h(a_{i_2,1}) \oplus u) = h(h(a_{31}) \oplus u),$$

$$b_{40} = h(h(a_{i_3,0}) \oplus u) = h(h(a_{50}) \oplus u),$$

$$b_{41} = h(h(a_{i_3,1}) \oplus u) = h(h(a_{51}) \oplus u),$$

$$b_{50} = h(h(a_{i_4,0}) \oplus u) = h(h(a_{60}) \oplus u),$$

$$b_{51} = h(h(a_{i_4,1}) \oplus u) = h(h(a_{61}) \oplus u).$$

The GC then constructs points $B_0, B_1, \cdots, B_5$:

$$B_0 = (b_{00}, b_{01}),$$

$$B_1 = (b_{10}, b_{11}),$$

$$B_2 = (b_{20}, b_{21}),$$

$$B_3 = (b_{30}, b_{31}),$$

$$B_4 = (b_{40}, b_{41}),$$

$$B_5 = (b_{50}, b_{51}).$$

If the condition

$$(2(b_{00} - b_{10}) \cdot 2(b_{11} - b_{21}) - 2(b_{10} - b_{20}) \cdot 2(b_{01} - b_{11})) \times \prod_{t=3}^{5} (-2b_{t1}) \neq 0 \quad (12)$$

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

Step 4) The GC expands B_0, B_1, B_2, B_3, B_4, and B_5 to become 5-dimensional points:

$$V_0 = (b_{00}, b_{01}, 0, 0, 0),$$

$$V_1 = (b_{10}, b_{11}, 0, 0, 0),$$
$$V_2 = (b_{20}, b_{21}, 0, 0, 0),$$
$$V_3 = (b_{30}, 0, b_{31}, 0, 0),$$
$$V_4 = (b_{40}, 0, 0, b_{41}, 0),$$
$$V_5 = (b_{50}, 0, 0, 0, b_{51}).$$

The GC is now going to establishe a 4-dimensional hyper-sphere based on the set of points $V_0, V_1, \cdots, V_5$. Suppose the central point of the hyper-sphere is C, and the square of the radius is R^2. The GC then constructs the set of equations about the hyper-sphere:

$$\begin{cases} (b_{00}-c_0)^2 + (b_{01}-c_1)^2 + (0-c_2)^2 + (0-c_3)^2 + (0-c_4)^2 = R^2 \\ (b_{10}-c_0)^2 + (b_{11}-c_1)^2 + (0-c_2)^2 + (0-c_3)^2 + (0-c_4)^2 = R^2 \\ (b_{20}-c_0)^2 + (b_{21}-c_1)^2 + (0-c_2)^2 + (0-c_3)^2 + (0-c_4)^2 = R^2 \\ (b_{30}-c_0)^2 + (0-c_1)^2 + (b_{31}-c_2)^2 + (0-c_3)^2 + (0-c_4)^2 = R^2 \\ (b_{40}-c_0)^2 + (0-c_1)^2 + (0-c_2)^2 + (b_{41}-c_3)^2 + (0-c_4)^2 = R^2 \\ (b_{50}-c_0)^2 + (0-c_1)^2 + (0-c_2)^2 + (0-c_3)^2 + (b_{51}-c_4)^2 = R^2 \end{cases} \tag{13}$$

Let matrix

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

and vectors

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

By subtracting the j-th equation from the $(j+1)$-th equation in (13), 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

$$Y \times X = Z \quad . \tag{14}$$

The condition in (12) in Step 3 guarantees that (14) has one and only one solution $(c_0, c_1, \cdots, c_4)$. Then the central point $C(c_0, c_1, \cdots, c_4)$ of the hyper-sphere is solved.

Step 5) The GC multicasts $C(c_0, c_1, \cdots, c_4)$ 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 $A_1(a_{10}, a_{11})$ along with the public information $C(c_0, c_1, \cdots, c_4)$ and u, and the third equation in (13):

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

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 $C(c_0, c_1, \cdots, c_4)$ and u, and the forth equation in (13):

$$\begin{aligned} K = R^2 - \|C\|^2 &= b_{30}^2 + b_{31}^2 - 2b_{30}c_0 - 2b_{31}c_2 \\ &= (h(h(a_{30}) \oplus u))^2 + (h(h(a_{31}) \oplus u))^2 \\ &\quad -2h(h(a_{30}) \oplus u)c_0 - 2h(h(a_{31}) \oplus u)c_2 \end{aligned}$$

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 $K = R^2 - \|C\|^2$.

4 Security and Performance Analysis

4.1 Security Analysis

1) Any illegitimate user who is not the group member cannot derive the group key. The group key K is calculated via (11), which invokes the private point $A_x(a_{x0}, a_{x1})$ of the member U_x. Any illegitimate user, who is not the group member, cannot calculate K without the knowledge of legitimate private point $A_x(a_{x0}, a_{x1})$.

2) Forward and backward secrecy is provided. The points that determine the hyper-sphere are computed by invoking the random number u and hash function h. According to the properties of random number and hash function, the central point C and the square of the radius R^2 that the GC calculates each time are different and independent, and then the group key $K = R^2 - ||C||^2$ is different each time when group members join or leave. Even though the group key was exposed at one time, the outsider would not know the group key at another time due to the group key update mechanism and independency of group keys. Therefore, forward and backward secrecy can be guaranteed.

3) It is extremely difficult for attackers to derive the GC's secrets S_0 and S_1. The attacker might be a legitimate member or an illegitimate user. The scheme can be attacked by a single attacker, or by a lot of attackers who collude and attempt to get more information.

Suppose the attacker is a legitimate member who can have the knowledge of R^2 and $C(c_0, c_1, \cdots, c_n)$, and suppose the attacker wishes to calculate one of

the GC's secret S_0. The attacker might firstly attempt to solve b_{00} and b_{01} from the equation

$$(b_{00} - c_0)^2 + (b_{01} - c_1)^2 + (0 - c_2)^2 + ... + (0 - c_n)^2 = R^2, \tag{15}$$

where R^2 and $C(c_0, c_1, \cdots, c_n)$ are known to the attacker, and then attempt to derive s_{00} and s_{01} from b_{00} and b_{01}. The attacker has to solve two difficult problems at the same time: 1) can two unknowns b_{00} and b_{01} be solved from (15)? 2) can the values of s_{00} and s_{01} be derived from b_{00} and b_{01} ?

For the first problem, it is very difficult to uniquely solve two unknowns b_{00} and b_{01} from only one equation (15) if $R^2 \neq 0$.

For the second problem, since $b_{00} = h(h(s_{00}) \oplus u)$ and $b_{00} = h(h(s_{01}) \oplus u)$, the properties of cryptographic hash function guarantee that it is computationally infeasible to derive s_{00} and s_{01} by knowing b_{00}, b_{01} and u.

Similarly, to derive the GC's another secret S_1 is as difficult as to derive S_0 .

Even though a lot of attackers collude, what the attackers can get about the GC's secrets S_0 and S_1 are two equations from (9):

$$(b_{00} - c_0)^2 + (b_{01} - c_1)^2 + (0 - c_2)^2 + ... + (0 - c_n)^2 = R^2 \tag{16}$$

and

$$(b_{10} - c_0)^2 + (b_{11} - c_1)^2 + (0 - c_2)^2 + ... + (0 - c_n)^2 = R^2, \tag{17}$$

where

$$b_{00} = h(h(s_{00}) \oplus u),$$

$$b_{01} = h(h(s_{01}) \oplus u),$$

$$b_{10} = h(h(s_{10}) \oplus u),$$

$$b_{11} = h(h(s_{11}) \oplus u).$$

The situation is similar to that of single attacker. First, it is unable to solve four unknowns b_{00}, b_{01}, b_{10}, and b_{11} from the above two equations (16) and (17) if $R^2 \neq 0$. Second, the properties of cryptographic hash function guarantee that it is computationally infeasible to derive s_{00}, s_{01}, s_{10}, and s_{11} from $b_{00} = h(h(s_{00}) \oplus u)$, $b_{01} = h(h(s_{01}) \oplus u)$, $b_{10} = h(h(s_{10}) \oplus u)$, and $b_{11} = h(h(s_{11}) \oplus u)$ by knowing b_{00}, b_{01}, b_{10}, b_{11} and u.

If the attacker isn't a group member, since any illegitimate user will know less information than legitimate member, for example, the illegitimate user will not be able to compute R^2 , then it will be more difficult to derive the GC's secrets $S_0 = (s_{00}, s_{01})$ and $S_1 = (s_{10}, s_{11})$ in the above cases.

Therefore, it is extremely difficult for attackers to derive the GC's secrets $S_0(s_{00}, s_{01})$ and $S_1(s_{10}, s_{11})$.

4) It is extremely difficult for attackers to derive the member's private point A_x. Similar to the situation of attacking the GC' secrets, suppose the attacker is a legitimate member and wishes to calculate the member's private point

$A_x(a_{x0}, a_{x1})$. Let us take $x = 1$ as an example. The attacker might attempt to solve b_{20} and b_{21} from the equation

$$(b_{20} - c_0)^2 + (b_{21} - c_1)^2 + (0 - c_2)^2 + ... + (0 - c_n)^2 = R^2, \tag{18}$$

where R^2 and $C(c_0, c_1, \cdots, c_n)$ are known to the attacker, then attempt to derive a_{10} and a_{11} from b_{20} and b_{21}. Obviously, to solve two unknowns b_{20} and b_{21} from one equation (18) is difficult if $R^2 \neq 0$; it is computationally infeasible to derive a_{10} and a_{11} by knowing b_{20}, b_{21} and u, because the cryptographic hash function h is invoked.

Even though a lot of attackers collude, the attackers cannot get more information than (18) about the member's private point $A_x(a_{x0}, a_{x1})$.

If the attacker isn't a group member, since any illegitimate user will know less information than legitimate member, then it will be more difficult to derive the member's private point in the above cases.

Therefore, it is extremely difficult for attackers to derive the member's private point $A_x(a_{x0}, a_{x1})$.

5) Exhausting attack is extremely difficult. All the computations in our scheme are over the finite field $GF(p)$, as long as the number of elements in the field is larger than a certain constant, e.g., 2^{128} , then it will be very difficult to explore the group key by brute force attack.

4.2 Performance Analysis

Suppose the length of the prime p is L bits.

Storage. Each member needs only to store its 2-dimensional private point. 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 (11), which includes two h hash functions, 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 (10) can easily be converted to a lower triangular matrix, the computation complexity of solving $(c_0, c_1, \cdots, c_n)$ from (10) 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.

Table 1 shows the performance requirements by both the GC and each member.

4.3 Experiments

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

Table 1. Performance Requirements by the GC and each Member

	Storage (bits)	Computation	Re-keying Messages	
			Number	Size(bits)
GC	$2 \times (n+2) \times L$	$\mathcal{O}(n)$	2	$(n+2) \times L$
Member	$2 \times L$	$4H + 4M + 5A$	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 a h hash function
M : average time required by a modular multiplication
A : average time required by a modular addition

the member is a HP XW4600 Workstation with 2.33GHz Intel dual-processor and 2GB 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, and 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 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 processing 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 group.

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

5 Comparison with Related Work

A summary of the comparison results are presented in Table 4 and 5.

GKMP (Group Key Management Protocol) is a simple extension from unicast to multicast, but not scalable and very inefficient, which also requires strong symmetric encryption and decryption functions. The major disadvantage of GKMP is its lack of the forward secrecy property [1]. Table 4 clearly shows that our scheme overwhelms 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 [6], OFCT [7], Hierarchical α-ary Tree with Clustering [8], Efficient Large-Group Key [9], etc. Hence, we only compare our scheme with LKH, but the results are similar to other tree-based schemes.

The LKH scheme also requires strong symmetric encryption and decryption functions. The advantages of our scheme overwhelming the LKH are as follows: 1) our scheme requires no strong encryption function, except the secure channel when new users register to join in the group for the first time; 2) our scheme is scalable for massive membership change; 3) the number of re-keying messages is $\mathcal{O}(1)$ in our scheme, but is $\mathcal{O}(\log_2 n)$ in LKH; 4) the computation complexity of each member is $\mathcal{O}(1)$ in our scheme, but is $\mathcal{O}(\log_2 n)$ in LKH.

The major differences between our scheme and LKH are that: 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 much faster than encryptions.

Note 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 [4].

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

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
Secrecy	No	Yes	Yes	Yes
Strong encryption needed	Yes	Yes	Yes	No
Efficient for very large group	No	Yes	No	Yes
Scalable to massive adding and removing members	No	No	Yes	Yes
Number of re-keying messages	n	$\mathcal{O}(\log_2 n)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Member computation complexity	$\mathcal{O}(1)$ decryptions	$\mathcal{O}(\log_2 n)$ decryptions	$\mathcal{O}(1)$ decryptions and modular operations	$\mathcal{O}(1)$ simple operations
GC computation complexity	$\mathcal{O}(n)$ encryptions	$\mathcal{O}(\log_2 n)$ encryptions	$\mathcal{O}(n)$ encryptions and modular operations	$\mathcal{O}(n)$ simple operations

Table 5. GC's Computation Comparison between Secure Lock and our Scheme

	Secure Lock	This Paper
Computation complexity	$E \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(2n) + A \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(2n)$	$H \cdot \mathcal{O}(2n) + M \cdot \mathcal{O}(2n) + A \cdot \mathcal{O}(4n) + R \cdot \mathcal{O}(n)$
Difference between schemes	$E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$	$2H \cdot \mathcal{O}(n) + 3A \cdot \mathcal{O}(n)$

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 h hash function
R : average time required by a multiplication reverse over $GF(p)$

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 [4]. 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, h hash function, and modular addition, respectively. The hash function h 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

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

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 overwhelming the ones of the SL include: 1) our scheme requires no strong encryption function, except the secure channel when new users register to join in the group for the first time; 2) our scheme is efficient for very large group; 3) the performance by each member and the GC in our scheme is much better than the ones in SL.

6 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 schemes 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, whose distance between every member's private point is identical. The central point is published and each member can compute a common group key via the square of the radius of the hyper-sphere.

We demonstrate that our new approach is secure, and its backward and forward secrecy can be guaranteed. The security of our approach relies on the fact that any illegitimate user cannot compute the group key without the legitimate private point.

The advantages of our scheme include: 1) it is not necessary to invoke strong encryption algorithm except the secure channel for one time, the re-keying messages can be broadcasted or multicasted via open channel, and the secure channel is required once only when new users register to join in the group; 2) it is very efficient and scalable for large size groups, and can handle massive membership change efficiently with only two re-keying messages, i.e., the central point of the hyper-sphere and a random number; 3) the storage and the computation overhead of each member are both small, which will not increase as the group size grows; 4) the GC's storage and computation cost is also low, where the storage and computation overhead are increased only linearly with the growth of the group size.

The performance estimates are confirmed by our experiments. For example, in the case of a group of size n=100000, the storage for each member's private information is 32 bytes, and 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.

References

1. 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.
2. 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
3. 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
4. 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.
5. 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.

6. 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.
7. 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, Mar. 1999, pp. 708-716.
8. R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," In Advances in Cryptology-EUROCRYPT '99, New York, 1999, Lecture Notes in Computer Science, vol. 1599, pp. 459-474.
9. 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, May 2001, pp. 247-262.
10. 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.
11. 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.
12. 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.
13. 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, Mar. 1996, ACM, New York, pp. 31-37.
14. 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., Nov. 1998, ACM, New York.
15. 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, 294- 302.
16. 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.
17. 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.
18. 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, July 1999. M. Blum and C H Lee, Eds. City University of Hong Kong Press, Hong Kong, China, pp. 192-202.
19. 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.
20. 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.
21. 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.

22. 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.
23. 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.
24. 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.
25. Y. Amir, Y. Kim, C. Nita-Rotaru, J. L. Schultz, J. Stanton, 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.
26. A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, 1996.
27. S. Mittra, "Iolus: A framework for scalable secure multicasting," In Proceedings of the ACM SIGCOMM, vol. 27, 4 (New York, Sept. 1997) ACM, New York, pp. 277-288.
28. 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.
29. B. Briscoe, "MARKS: Multicast key management using arbitrarily revealed key sequences," In Proceedings of the 1st International Workshop on Networked Group Communication, Pisa, Italy, Nov. 1999.
30. R. Molva, and A. Pannetrat, "Scalable multicast security in dynamic groups," In Proceedings of the 6th ACM Conference on Computer and Communications Security, Singapore, Nov. 1999, ACM, New York, 101-112.
31. 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., May 2000, IEEE Computer Society Press, Los Alamitos, Calif.
32. 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, June 2001.
33. 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., June 2002, IEEE Computer Society Press, Los Alamitos, Calif.
34. 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.
35. 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 Transactions on Dependable and Secure Computing, vol. 6, no. 3, pp. 188-201, 2009.
36. 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.
37. 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.

38. 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.
39. 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.
40. X. Yi, C. K. Siew, and 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.
41. 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.