A Simple and Efficient New Group Key Management Approach Based on Linear Geometry

Shaohua Tang[1,2], Jintai Ding[2,3], and Yujun Liang[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. A new fundamental and secure group key management approach with a group controller GC using the theory of polynomial functions over a vector space over finite field is developed, where each member in the group corresponds to a vector in the vector space and the GC computes a central vector, whose inner product with every member's ID vector are identical. The central vector is published and each member can compute a common group key via inner product. 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 vector 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: key management, group communication, linear geometry, security

1 Introduction

With the rapid expansion of Internet technology and the popularization of multicast, group-oriented applications, such as video conference, network games, and video on demand, etc., play more and more important roles. How to protect the communication security of these applications is critical. A secure group communication system should not only provide data confidentiality, user authentication, and information integrity, but also good scalability. For a secure group communication system, a secure, efficient, and robust group key management approach is essential. Key management of secure group communication can be divided into three different categories: the first is called a centralized scheme with a central group controller (GC), the second is called a distributed scheme that without GC, and the third is called a decentralized scheme that can be regarded as the mix of previous two [1]. In this paper, we focus on the first category.

For the key management of secure group communication, there are many different approaches. In this paper, we will look the problem from a very different angle from before, namely a very fundamental mathematical point of view.

Let us assume that we have a GC and n members in our group. Clearly GC must assign each member an identifier to manage them, which we call V_1,...,V_n. From a mathematical point of view, we can assume that each identifier is a vector (which we call the ID vector) in a vector space over a finite field $\mathbb{F}$ with q elements, namely:

$$V_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,m}),$$

where each $v_{i,j}$ is an element in the finite field $\mathbb{F}$. Such ID vector should surely be kept secure for each member. To derive common key for the group, clearly the GC must develop a mechanism such that each member can use its information – the ID vector to derive the common key. Such a mechanism can be viewed as a functions f. Then we know this function must have the property:

$$f(V_1) = f(V_2) = \cdots = f(V_n) = k,$$

where k is the common key.

From the theory of finite field, we know that this function must be a polynomial function, since every function over a field is a polynomial function. Due to the efficiency consideration, we would like to start with the simplest function, namely the linear function. In this paper, we develop such a new secure group key management approach based on linear geometry, where the function f can be geometrically interpreted as the inner product of two vectors:

$$f(V_i) = X_i \times A^T,$$

where X_i is a vector derived from V_i, A is a vector selected by GC, and A^T is the transpose of the vector A.

Fig. 1 illustrates the geometrical interpretation of inner product of two vectors when we consider $\mathbb{F}$ to be the field of real number.

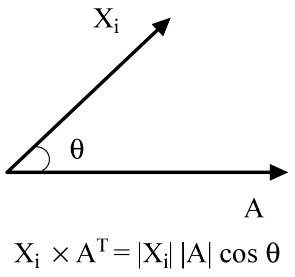

Fig. 1. The geometrical interpretation of inner product of two vectors over real number field

In this scheme, each key is completely independent from any previous used and future keys, therefore the backward and forward secrecy can be guaranteed.

We systematically analyze this construction. The new scheme has the advantages of efficient computations with small memory for each group member and efficient massive membership updates.

The rest of this paper is organized as follows. We first present a brief summary of related schemes on secure group key management in Section II. The proposed secure group key management approach is constructed in Section III. Section IV discusses and analyzes the security and the performance of new scheme. Experiments are also presented in Section IV. Finally, Section V summarizes the major contributions of this paper.

2 Previous 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.

Some typical schemes in the centralized category include Group Key Management Protocol (GKMP)[2, 3], Secure Lock (SL)[4], Logical Key Hierarchy (LKH)[5], etc. A brief survey of these schemes is summarized as follows. Hereafter, suppose n is the number of members in the group.

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 (Group Controller) 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, 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 $\mathcal{O}(2 \times \log_2 n)$. But 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 a-ary Tree with Clustering[8], Efficient Large-Group Key[9], etc. They can be regarded as the similarity or improvement of LKH.

The features of distributed schemes are that 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: [20–25], 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, 37–40], wireless sensor network[36], storage area networks[41], etc.

3 Approach Based on Linear Geometry

3.1 Notation

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

GC	the group controller, who manages the group initialization, member joining/leaving operations
$\mathbb{F}$	a finite field
g	a secure hash function over $\mathbb{F}$ without any algebra features, like SHA-1
n	number of members in the group
m	the dimension of group member's private vectors, and $2 \leq m \leq n$
r	a random number (public information)
A	central vector (public information)
V_i	member u_i's ID vector (private information)
$J^{(i)}$	member u_i's position set (private information)
k	the group key (secret information)

$\mathbb{F}$, g, n, m, r and A are public. V_i and $J^{(i)}$ are member u_i's private information. The group key k is a secret shared by all members. It is assumed that all computations hereafter are over the finite field $\mathbb{F}$.

3.2 Generation of Group Key

Suppose there are n members in the group, and the set of members are denoted by $U = \{u_1, u_2, ..., u_n\}$. The GC chooses an integer m, where $2 \leq m \leq n$. The parameter m should be fixed during the whole procedure of initial group key generation, adding members, removing members, and massive adding and removing. The process to construct the group key includes the following steps.

Step 1. For each member $u \in U$, the GC assigns an unique integer i as u's **ID**, where $1 \leq i \leq n$. For $i = 1, 2, ..., n$, the GC selects m random numbers $v_{i,1}, v_{i,2}, ..., v_{i,m} \in F$, and constructs m-dimensional vector $V_i = (v_{i,1}, v_{i,2}, \cdots, v_{i,m})$ over the finite field $\mathbb{F}$. The vectors $V_1, ..., V_n$ should satisfy $V_y \neq V_z$ if $y \neq z$, where $1 \leq y, z \leq n$. The vector V_i is called u_i's private vector (or **ID vector**).

The ID i and private vector V_i should be kept secret.

Step 2. The GC selects random number $r \in F$.

Step 3. The GC maps m-dimensional private vectors V_i to n-dimensional vectors X_i via the following computations.

For $i = 1, 2, ..., m$, the GC lets $j_1^{(i)} = 1$, $j_2^{(i)} = 2$, ..., and $j_m^{(i)} = m$.

For $i = m + 1, ..., n$, the GC lets $j_1^{(i)} = i$; randomly selects an integer $j_2^{(i)}$ satisfying $1 \leq j_2^{(i)} < i$; and randomly selects different integers $j_3^{(i)}, j_4^{(i)}, ..., j_m^{(i)}$ satisfying $1 \leq j_y^{(i)} \leq n$ and $j_y^{(i)} \neq j_z^{(i)}$ if $y \neq z$, where $1 \leq y, z \leq m$.

Denote the set of $j_1^{(i)}, j_2^{(i)}, ..., j_m^{(i)}$ by $J^{(i)} = \{j_1^{(i)}, j_2^{(i)}, ..., j_m^{(i)}\}$. $J^{(i)}$ is called u_i's non-zero position set, or position set for short, which is used to store the non-zero positions of vector X_i in the following steps.

For $i = 1, 2, ..., n$,

for $y = 1, 2, ..., m$, the GC computes

$$x_{i,j_y^{(i)}} = g(g(v_{i,y}) \oplus r),$$

where $g(v_{i,1}), g(v_{i,2}), ..., g(v_{i,m})$ can be pre-computed and stored for later use. For each $j \notin \{j_1^{(i)}, j_2^{(i)}, ..., j_m^{(i)}\}$, the GC lets

$$x_{i,j} = 0.$$

Then the GC constructs new vectors

$$X_1 = (x_{1,1}, x_{1,2}, \cdots, x_{1,n}),$$
$$......,$$
$$X_n = (x_{n,1}, x_{n,2}, \cdots, x_{n,n})$$

over the finite field $\mathbb{F}$.

Step 4. The GC judges whether $X_1, ..., X_n$ are linearly independent or not. If they are linearly independent, then proceed to Step 5; otherwise, repeat Steps 2 - 4.

Note that since r and vectors V_i are all randomly selected, it is very easy to get a set of linearly independent vectors $X_1, ..., X_n$.

Step 5. The GC delivers each member's private vector V_i and the position set $J^{(i)}$ to the member u_i individually via a secure channel, where $i = 1, 2, ..., n$.

Note that the private vector V_i and the position set $J^{(i)}$, including the ID i, are the member u_i's private information and should be kept secret.

Step 6. The GC chooses a random number $k \in F$ as the group key. Suppose $a_1, a_2, ..., a_n$ are unknown parameters. The GC attempts to solve $a_1, a_2, ..., a_n$ from the following system of linear equations.

$$\begin{cases} x_{1,1}a_1 + x_{1,2}a_2 + ... + x_{1,n}a_n = k \\ x_{2,1}a_1 + x_{2,2}a_2 + ... + x_{2,n}a_n = k \\ ...... \\ x_{n,1}a_1 + x_{n,2}a_2 + ... + x_{n,n}a_n = k \end{cases} \tag{1}$$

Let vector $K = (k, k, ..., k)$, $A = (a_1, a_2, ..., a_n)$, and let matrix

$$X = \begin{bmatrix} X_1 \\ X_2 \\ ... \\ X_n \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & ... & x_{1,n} \\ x_{2,1} & x_{2,2} & ... & x_{2,n} \\ ... & ... & ... & ... \\ x_{n,1} & x_{n,2} & ... & x_{n,n} \end{bmatrix},$$

then the system of linear equations in (1) can be represented as vector expression:

$$X \times A^T = K^T \tag{2}$$

Since $X_1, X_2, ..., X_n$ are linearly independent vectors, the determinant of coefficient matrix $|X| \neq 0$. Therefore, the set of equations in (2) or (1) has unique solutions $A = (a_1, a_2, ..., a_n)$.

Step 7. The GC broadcasts vector $A = (a_1, a_2, ..., a_n)$ and random number r to all group members via open channel.

Step 8. After receiving $A = (a_1, a_2, ..., a_n)$ and r, each member u_i will firstly map its m-dimensional private vector V_i to a n-dimensional vector X_i.

By retrieving the first element $j_1^{(i)}$ from the set $J^{(i)}$, the member u_i can know its ID i.

For $y = 1, 2, ..., m$, the member u_i computes

$$x_{i,j_y^{(i)}} = g(g(v_{i,y}) \oplus r),$$

where $g(v_{i,1}), g(v_{i,2}), ..., g(v_{i,m})$ can be pre-computed and stored for later use.

For each $j \notin \{j_1^{(i)}, j_2^{(i)}, ..., j_m^{(i)}\}$, the member u_i lets

$$x_{i,j} = 0.$$

Then each member u_i can construct the vector

$$X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,n}),$$

and calculate the group key k via the following equation, whose value is the inner product of two vectors X_i and A:

$$\begin{aligned} k &= X_i \times A^T = x_{i,1}a_1 + x_{i,2}a_2 + ... + x_{i,n}a_n \\ &= x_{i,j_1^{(i)}} a_{j_1^{(i)}} + x_{i,j_2^{(i)}} a_{j_2^{(i)}} + ... + x_{i,j_m^{(i)}} a_{j_m^{(i)}} \end{aligned} \tag{3}$$

Obviously, we learn from (1) that each u_i can yield the same value k via (3), where $i = 1, ..., n$.

3.3 Membership Changing and Re-keying

If there are members need to join or leave the group, then the old group key should be discarded and a new group key should be generated to guarantee the backward and forward secrecy.

When adding new members, the steps are similar to the ones in the previous sub-section. In brief, the GC assigns new IDs and selects new private vectors V_is for new members in Step 1, and re-chooses a random number r in Step 2. Then the GC maps new private vectors V_i to vectors X_i and determines if $X_1, \ldots, X_n$ are linearly independent or not in Step 3 and 4. After that, the GC delivers new private vectors and new position sets to each new member individually via a secure channel in Step 5. The GC re-selects a new group key k and re-calculates the public central vector A in Step 6, then broadcasts central vector A and random number r to all group members via open channel in Step 7. Finally, each legal member can compute the common group key k by using its position set $J^{(i)}$ and private vector V_i in Step 8.

When removing members, the GC deletes the leaving members' private vectors and the position sets in Step 1. Other steps are similar to the ones in the previous sub-section except that each member has already owned its private information V_i and $J^{(i)}$, then it is not necessary to re-send the private information by the GC again.

Massive adding and removing, where a lot of members join the group and other members leave the group at the same time in batch mode, can be regarded as the combination of adding operation and removing operation. There is only one re-keying operation for massive adding and removing regardless the number of members changed.

4 Security, Performance Analysis and Experiments

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 (3), which is the inner product of the group member u_i's vector X_i and the group's public vector A. The vector X_i is derived by invoking u_i's private vector V_i , the position set $J^{(i)}$, and the random number r. Any illegitimate user, who is not the group member, cannot calculate X_i without the knowledge of legitimate vector V_i and the set $J^{(i)}$, therefore cannot derive the group key k.

2) Forward and backward secrecy is provided. Backward secrecy is used to prevent a new member from decoding messages exchanged before it joined the group [10, 11, 1]. In this paper, a new group key k is randomly generated when a new member joins, therefore it is not able to decrypt previous messages by the new members even if earlier messages enciphered with the old group key have been stored.

Forward secrecy is used to prevent a leaving or expelled group member to continue accessing the group's communication (if it keeps receiving the messages) [10, 11, 1]. In this paper, the group key is changed each time when a member leaves, then the leaving or expelled member will not be able to decrypt group messages enciphered with the new group key.

3) It is extremely difficult for an attacker to derive member u_i's private vector V_i. The attacker might be a legitimate member or an illegitimate user. If the attacker isn't a group member, since any illegitimate user will know less information than group member, it will be more difficult to break others' private information. Then we can assume that the attacker is the legitimate member u_j, and tried to derive other member u_i's private vector V_i . The security is discussed from the following aspects.

The attacker is unable to get other member u_i's private information V_i and $J^{(i)}$ through network communication. Each group member u_i's private vector V_i is randomly generated by the GC, and u_i's private position set $J^{(i)}$ is assigned by the GC. The GC sends the member u_i's private information V_i and $J^{(i)}$ via secure channel, therefore other members or illegitimate users cannot get V_i and $J^{(i)}$ through network communication.

The attacker is unable to solve member u_i's secret vector $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,n})$ from (3), then unable to derive u_i's private vector V_i via X_i. The attacker u_j can compute the group key k, and might try to derive member u_i's information $x_{i,1}, x_{i,2}, ..., x_{i,n}$ from (3). Even though there are $(n - m)$ zero elements among $x_{i,1}, x_{i,2}, ..., x_{i,n}$, where $m \geq 2$, but no one can uniquely solve m unknowns $x_{i,j_1^{(i)}}, x_{i,j_2^{(i)}}, ..., x_{i,j_m^{(i)}}$ from one linear equation.

Therefore, the attacker cannot solve u_i's secret vector X_i, then cannot compute u_i's private vector V_i via the X_i.

The attacker cannot derive u_i's private vector V_i by recording a series of public parameters A and r at different time period. The attacker u_j might record a series of public central vector A and random number r. Suppose that the central vector and random number recorded by u_j at time t_1 is denoted by $A^{(1)}$ and $r^{(1)}$, where $A^{(1)} = (a_1^{(1)}, a_2^{(1)}, ..., a_n^{(1)})$, and the group key at time t_1 is $k^{(1)}$; the central vector and random number recorded by u_j at time t_2 is denoted by $A^{(2)}$ and $r^{(2)}$, where $A^{(2)} = (a_1^{(2)}, a_2^{(2)}, ..., a_n^{(2)})$, and the group key at time t_2 is $k^{(2)}$; and so on. Then the attacker u_j can have a series of equations:

$$\begin{aligned} k^{(1)} &= x^{(1)}_{i,j_1^{(i)}} a^{(1)}_{j_1^{(i)}} + x^{(1)}_{i,j_2^{(i)}} a^{(1)}_{j_2^{(i)}} + ... + x^{(1)}_{i,j_m^{(i)}} a^{(1)}_{j_m^{(i)}} \\ &= g(g(v_{i,1}) \oplus r^{(1)}) a^{(1)}_{j_1^{(i)}} + ... + g(g(v_{i,m}) \oplus r^{(1)}) a^{(1)}_{j_m^{(i)}}, \end{aligned}$$

$$\begin{aligned} k^{(2)} &= x^{(2)}_{i,j_1^{(i)}} a^{(2)}_{j_1^{(i)}} + x^{(2)}_{i,j_2^{(i)}} a^{(2)}_{j_2^{(i)}} + ... + x^{(2)}_{i,j_m^{(i)}} a^{(2)}_{j_m^{(i)}} \\ &= g(g(v_{i,1}) \oplus r^{(2)}) a^{(2)}_{j_1^{(i)}} + ... + g(g(v_{i,m}) \oplus r^{(2)}) a^{(2)}_{j_m^{(i)}}, \end{aligned}$$

$$...... \tag{4}$$

In order to solve the unknowns $v_{i,1}, v_{i,2}, ..., v_{i,m}$ from the above series of equations, the attacker u_j has to solve two difficult problems at the same time: 1) can u_j derive x from $y = g(x)$ by knowing y ? where g is a secure hash function without any algebra features, like SHA-1; 2) can u_j know the values of $j_1^{(i)}, ..., j_m^{(i)}$? where the position set $J^{(i)} = \{j_1^{(i)}, ..., j_m^{(i)}\}$ is u_i's private information and is unknown to u_j .

For the first problem, the properties of secure hash function guarantee that it is computationally infeasible to derive x from $y = g(x)$ by knowing y.

For the second problem, the attacker u_j might try to guess the value of each element of the set $J^{(i)} = \{j_1^{(i)}, ..., j_m^{(i)}\}$, the probability to have a successful guess is

$$P = \frac{1}{n} \times \frac{1}{n/2} \times \frac{1}{C_{n-2}^{m-2}} = \frac{2(m-2)!(n-m)!}{n^2(n-2)!} \tag{5}$$

By observing (5), we know that the greater n is, the smaller P is; and for $2 \le m \le \frac{n}{2} - 1$, the greater m is, the smaller P is. The value of P in (5) reflects the difficulty of guess, i.e., the smaller P is, the more difficult for the attackers to have a successful guess.

Therefore, even with the knowledge of a series of equations about $v_{i,1}, v_{i,2}$, ..., $v_{i,m}$ in (4), when facing the two difficult problems at the same time, the attacker u_j is still extremely difficult to solve member u_i's private information $v_{i,1}, v_{i,2}, ..., v_{i,m}$.

The member u_i's private vector V_i cannot be broken even a lot of attackers collude. Even though a lot of attackers collude, they cannot know more information then the attacker u_j to solve member u_i's private information $v_{i,1}, v_{i,2}, ..., v_{i,m}$. Therefore, group member u_i's private vector V_i is extremely difficult to be derived by the attackers.

4) Brute force attack to explore the group key is extremely difficult. The group key k is randomly selected from the finite field $\mathbb{F}$. 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.

5) The trade-off between security and performance. As analyzed above, our scheme is secure for all m that satisfies $2 \leq m \leq n$. From performance perspective, the greater m will increase the storage and computation overhead, therefore, we can have a better performance for a small m, especially $m = 2$ will become an optimized and secure model. From security perspective, for $2 \leq m \leq \frac{n}{2} - 1$, the greater m is, the more difficult for the attacker to break our scheme.

4.2 Performance Analysis

The performance discussed hereafter is aimed at the optimized model where $m = 2$. Suppose it requires L bits to represent an element in the finite field $\mathbb{F}$.

Storage for secret information. Each member needs only to store its private two-dimensional vector V_i and the set $J^{(i)}$ with two elements. The GC should store all the group members' private two-dimensional vectors $V_1, V_2, ..., V_n$, and the position sets $J^{(1)}, J^{(2)}, ..., J^{(n)}$. Then the storage for each member is $4L$ bits, and the one for the GC is $4nL$ bits.

Computation. The computation by each member is to calculate vector X_i in Step 8, and to compute the group key via (3). The values of $g(v_{i,1})$ and $g(v_{i,2})$ can be pre-computed, then the computation by each member includes two XOR operations, two g hash functions, two multiplications and one addition over finite field.

The computation for the GC is to solve a system of linear equations in (1). Since the coefficient matrix in (1) is sparse and like a triangular matrix when $m = 2$, the computation overhead of solving $(a_0, a_1, \ldots, a_n)$ from (1) includes $2n$ times of g hash function, $(3n+8)$ multiplications, $(n+1)$ multiplicative reverses, and $(n+3)$ additions over the finite field $\mathbb{F}$.

Number and size of re-keying message. The total number of re-keying messages is two, one is the vector A, another is the random number r. The size of re-keying messages is $(n+1)L$ bits. The re-keying messages are broadcasted or multicasted via open channel.

Batch processing for massive membership change. If there are a lot of users joining and leaving simultaneously, only one batch processing is needed.

A summary of the performance requirements for both GC and members is presented in Table 1.

Table 1. Performance requirements by the GC and members

		GC	Member
Storage (bits)		$4nL$	$4L$
Computation overhead		$2nG + (3n + 8)M + (n + 3)A + (n + 1)R$	$2G + 2M + A$
Re-keying messages	Number	2	0
	Size (bits)	$(n + 1)L$	0

Notation for Table 1:

n	number of members in the group
L	the max length in bits to represent an element in $\mathbb{F}$
G	average time required by a g hash function
M	average time required by a multiplication over $\mathbb{F}$
A	average time required by an addition over $\mathbb{F}$
R	average time required by a multiplication reverse over $\mathbb{F}$

4.3 Experiments

Our experiments are conducted on a 2.33GHz Intel Xeon quad-core dual-processor PC server with 4GB memory for GC, and a HP XW4600 Workstation with 2.33GHz Intel dual-processor with 2GB memory for a member of the group. The programs are written in C/C++. We select $m = 128$, and $\mathbb{F}$ to be of 128 bits, i.e., $L = 128$. Each experiment is repeated 20 times. The results presented are the average, and the difference between the same experiments is less than 2%. For the GC computation, OpenMP is used to speed up the computation by utilizing 8 parallel threads in dual quad-core processors. The summary of the experimental results are presented in Table 2 and 3.

Table 2. Storage and computation required by the members

Size of group	Storage (bytes)	Computation (ms)
10	64	0.0002
100	64	0.0002
1,000	64	0.0002
10,000	64	0.0002
100,000	64	0.0002
1,000,000	64	0.0002
10,000,000	64	0.0002

Table 2 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.

Table 3. Storage and computation required by the GC

Size of group	Storage (bytes)	Computation (ms)
10	640	0.1512
100	6,400	0.4096
1,000	64,000	2.4139
10,000	640,000	16.5994
100,000	6,400,000	79.0096
1,000,000	64,000,000	555.8474
10,000,000	640,000,000	4,011.9175

In Table 3, the first column represents the size of the group; the second, the storage for the computation, and the third, the computation time with using OpenMP. The GC needs only 4,011 ms or 4.011 seconds to process each re-keying for a group of size $n = 10^7$. Therefore, the GC can manage a large group efficiently.

The experimental results confirm that our scheme is very scalable and very efficient for large group.

Table 4. Feature and computation complexity comparison among schemes

	GKMP	LKH	Secure Lock	This Paper
Major principle adopted	Encryption	Tree structure	Chinese Remainder Theorem	Linear geometry
Secrecy	No	Yes	Yes	Yes
Strong encryption needed	Yes	Yes	Yes	No
Efficient for very large group	No	Yes	No	Yes
Scalable to massive membership change	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	$\mathcal{O}(1)$ linear operations
GC computation complexity	$\mathcal{O}(n)$ encryptions	$\mathcal{O}(2\log_2 n)$ encryptions	$\mathcal{O}(n)$ encryptions and modular operations	$\mathcal{O}(n)$ linear operations

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 sym-

Table 5. GC's Computation comparison between Secure Lock and our scheme

	Secure Lock	This Paper
Computation overhead	$nE + 2nM + nA + 2nR$	$2nG + (3n + 8)M$ $+(n + 3)A + (n + 1)R$
Computation complexity	$E \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(2n)$ $+A \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(2n)$	$2G \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(3n)$ $+A \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$
Difference between schemes	$E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$	$2G \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(n)$

metric 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 a-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 encryption functions; 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: linear geometry is adopted in our scheme, but tree structure is adopted in LKH; 2) The computation complexity by the GC in our scheme is $\mathcal{O}(n)$ linear operations, but the one in LKH is $\mathcal{O}(2 \times \log_2 n)$ encryptions. In average conditions, the computation of linear operations can be much faster than encryptions.

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

Table 5 compares the computation by the GC in SL and our scheme. The first row shows the computation overhead. The one in SL is based on an optimized CRT [4]. The second row presents the computation complexity. And the third row shows the difference of computation complexity of two schemes by omitting the identical items in the second row. The complexity differences are: $E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$ in SL, and $2G \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(n)$ in our scheme, where n is the number of members in the group, E, R, G and M are average time required by encryption, modular multiplication reverse, g mapping, and modular multiplication, respectively. The mapping g is a simple mapping and can be computed extremely fast, so $E \gg 2G$. Modular reverse operation over finite field is a time-consuming computation, thus $R > M$, and then

$$E \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n) \gg 2G \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(n),$$

or

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

$$\gg 2G \cdot \mathcal{O}(n) + M \cdot \mathcal{O}(3n) + A \cdot \mathcal{O}(n) + R \cdot \mathcal{O}(n)$$

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

The advantages of our scheme overwhelming the SL include: 1) our scheme requires no encryption functions; 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, namely a very fundamental mathematical point of view. A family of new and simple secure group key management schemes based on pure linear geometry is constructed, where each member in the group corresponds to a vector in the vector space (which we call the ID vector) and the group controller (GC) computes a public central vector, whose inner product with every member's private ID vector are identical. The central vector is published and each member can compute a common group key via the inner product.

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 correct inner product without the legitimate vector, therefore cannot derive the group key.

The advantages of our scheme include: 1) it is not necessary to invoke strong encryption algorithm, the re-keying messages can be broadcast or multicast via open channel, and the secure channel is required only for the initialization when members register to form the group or new members join in; 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 vector 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, and parallel programming can be used to speed up the GC's computation.

The performance estimates are confirmed by our experiments. For example, in the case of a group of size $n = 10^7$, the storage for each member's private information is 64 bytes, and the time for each member to compute the group key is 0.0002 ms or 2×10^{-7} seconds, and the time for the GC to process membership change is 4011 ms or 4.011 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, 2003.
2. H. Harney, C. Muckenhirn, and T. Rivers, "Group key management protocol (GKMP) specification," *RFC 2093*, July 1997.
3. H. Harney, C. Muckenhirn, and T. Rivers, "Group key management protocol (GKMP) architecture," *RFC 2094*, July 1997.
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, 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, 2003.
7. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient constructions," *IEEE INFOCOM*, vol. 2, pp. 708-716, 1999.
8. R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," *Lecture Notes in Computer Science*, pp. 459-474, 1999
9. A. Perrig, D. Song, and J. Tygar, "ELK, a new protocol for efficient large-group key distribution," In *IEEE Symposium on Security and Privacy*, pp. 247-262, 2001.
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, no. 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.