A Novel Authenticated Group Key Distribution Scheme

Run-hua Shi, Hong Zhong and Shun Zhang
School of Computer Science and Technology, Anhui University, Hefei, 230601, China
[e-mail: shirh@ahu.edu.cn, zhongh@mail.ustc.edu.cn and shzhang27@163.com]
*Corresponding author: Run-hua Shi

Received August 17, 2015; revised November 1, 2015; revised December 11, 2015; accepted December 22, 2015; published February 29, 2016

Abstract

In this paper, we present a novel authenticated group key distribution scheme for large and dynamic multicast groups without employing traditional symmetric and asymmetric cryptographic operations. The security of our scheme is mainly based on the basic theories for solving linear equations. In our scheme, a large group is divided into many subgroups, where each subgroup is managed by a subgroup key manager (SGKM) and a group key generation center (GKGC) further manages all SGKMs. The group key is generated by the GKGC and then propagated to all group members through the SGKMs, such that only authorized group members can recover the group key but unauthorized users cannot. In addition, all authorized group members can verify the authenticity of group keys by a public one-way function. The analysis results show that our scheme is secure and efficient, and especially it is very appropriate for secure multicast communications in large and dynamic client-server networks.

Keywords: Group Key Distribution, Key Agreement, Authentication, Linear Equations

This work was supported by National Natural Science Foundation of China (Nos. 61173187, 61173188 and 11301002), the Natural Science Foundation of Anhui Province (Nos. 11040606M141 and 1408085QF107), and the 211 Project of Anhui University (Nos.33190187 and 17110099).

http://dx.doi.org/10.3837/tiis.2016.02.027 **ISSN : 1976-7277**

1. Introduction

Distributing a group key to all group members is a complicated task especially in a dynamic group where members may join and leave at any time. When any member joins/leaves the group, it needs to update and redistribute the group key to all group members, in order to ensure that a leaving member cannot learn about the new group keys after he leaves the group and a new member cannot learn about the previous group keys after he joins the group. In last decades, group key management had received much attention and was always a research focus. Therefore, lots of group key management schemes have been proposed. Generally speaking, these schemes can be roughly classified into three categories: a centralized key distribution scheme, a distributed key agreement scheme, and a hybrid group key management scheme.

In centralized key distribution schemes [1-9], all group members trust a centralized key management center, which generates and distributes the group keys and is also responsible to update the group key when group members join or leave the group. One of the most known centralized key distribution schemes was the logical key hierarchy (LKH) [1,2], which reduced the rekey messages and encryption operations from $O(N)$ to $O(\log N)$, where N was the number of group members. An improvement in the binary key tree was the one-way function tree (OFT) [3] in which an internal node key was generated from its children node keys. In comparison with the top-down LKH method, the bottom-up OFT algorithm approximately halved the number of bits that needed to be broadcast to members in order to rekey after a member was added or evicted. Furthermore, Perrig et al. [4] proposed an Efficient Large-group Key (ELK) distribution scheme. It was similar to OFT in the sense that intermediate keys were generated from its children, but pseudo-random functions (PRFs) were used rather than one-way functions, thus it further reduced the size of rekey messages for each member join from $\log N$ to 0. However, for a member join, the manager had to re-compute all auxiliary keys of the key tree.

In distributed key agreement schemes [10-21], all group members contribute to the generation of group keys and are equally responsible for the rekeying and distribution of group keys. In 1976, Diffie and Hellman [10] first described a method for two parties to agree upon a shared key in such a way the key would be unavailable to eavesdroppers. Thereafter, there were many distributed key agreement schemes, where most distributed key agreement schemes were the natural generalizations of the DH key agreement scheme. Well known schemes among these were perhaps the works of Ingemarsson *et al*. [11], Burmester and Desmedt [12], Steiner *et al*. [13], Joux *et al*. [14] and Kim *et al*. [15].

In hybrid group key management schemes [22-26], the authors make the best use of the individual advantages of both the centralized key distribution scheme and the distributed key agreement scheme. For example, Kwak et al. [25] presented a hybrid group key management scheme, which combined the LKH [1,2] and the tree-based group Diffie-Hellman (TGDH) schemes [15], and thus avoided the single point of failure problems of the LKH with much more enhanced performance than the TGDH.

In addition, there were also some novel group key management schemes for emerging networks. For example, in 2011, N.T.T. Huyen, *et al*. [27] presented two approaches for the polynomical pre-distribution scheme by exploiting the signal range and the deployment error, which are especially suitable for sensor networks. In 2013, to overcome the high frequency of group rekeying, D.H. Je *et al*. [28] proposed a novel group key management scheme, which is very suitable for vehicle networks. Above all, their proposed subscription-period-aware key management scheme can greatly reduce the communication, computation, and storage complexity in multicast group rekeying from $O(N)$ to $O(1)$, where N is the number of vehicles in a single group rekeying process.

In this paper, we mainly focus on the centralized key distribution schemes, which are more suitable for a large and dynamic multicast group due to low computation and communication costs. In early proposed group key schemes, the main secure goal is to protect the confidentiality of a broadcast key or re-key message, but lack of authentication. So these schemes simplify the security problem by assuming a passive adversary. To protect against active adversaries, most existed schemes can be transformed to the corresponding authenticated group key schemes using public key cryptographic techniques as the compiler of Katz and Yung [16]. However, the management of the public keys in a large and dynamic group is also a heavy burden. Thus, to seek an efficient authenticated group key distribution scheme without using public key cryptographic techniques becomes a very significant work in secure group communications.

In this paper, we present a novel authenticated group key distribution scheme for large and dynamic multicast groups without employing traditional symmetric and asymmetric cryptographic techniques. The security of our scheme is mainly based on the basic theories for solving linear equations. Compared with other centralized key distribution schemes of Key trees, or hierarchical key structures, our scheme has four highlighted advantages: 1) Our proposed scheme is not based on any difficult assumption; 2) When any group member joins/leaves the group, the number of auxiliary secrets (or keys) required to be updated is $O(1)$ instead of $O(\log N)$; 3) In order to obtain the group key, the computation cost of each group member is very low because it only needs to compute one inner product instead of other complex cryptographic operations; 4) It can provide group key authentication without using encryption and signature techniques.

2. Preliminaries

2.1 The secure goals of group key management

Referring to the literatures [4,9], we first review four secure goals of group key management: Group Key Confidentiality, Forward Secrecy, Backward Secrecy and Group Key Authentication.

1. **Group Key Confidentiality** is to protect the group key such that it can only be recovered by authorized group members; but not by any unauthorized user.

2. **Forward Secrecy** is to guarantee that a passive adversary who knows a contiguous subset of old group keys cannot discover subsequent group keys. This property ensures that a member cannot learn about the new group keys after he leaves the group.

3. **Backward Secrecy** is to guarantee that a passive adversary who knows a subset of group keys cannot discover preceding group keys. This property ensures that a new member cannot learn about the previous group keys after he joins the group.

4. **Group key authentication** is to provide assurance to authorized group members that the group key is distributed by GKGC; but not by an active attacker.

When any group member joins/leaves a group, obviously it needs to execute a rekeying (updating group key) procedure in order to maintain the forward secrecy and backward secrecy.

2.2 The basic theories for solving linear equations

There are lots of basic theories involved in solving linear equations. In the following section, we only introduce two theorems related to this paper.

Theorem 1. The necessary and sufficient condition for the solvability of linear equations $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$ is that the rank of the coefficient matrix $\mathbf{A}$ is equal to that of the augmented matrix $\overline{\mathbf{A}}$. That is, $r(\mathbf{A}) = r(\overline{\mathbf{A}})$. Where $\mathbf{A}$ is an $m \times n$ matrix, $\vec{\mathbf{x}}$ an n-dimensional vector and $\vec{\mathbf{y}}$ an m-dimensional vector.

Theorem 2. Furthermore, for the solvable linear equations $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$, if $r(\mathbf{A}) = n$, there exists a unique solution; if $r(\mathbf{A}) < n$, there exist infinitely many solutions. Furthermore, for the solvable linear equations $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$ over Z_p, if $r(\mathbf{A}) < n$, there exist at least p solutions, where p is a large prime integer and $Z_p = \{0,1,2,\ldots,p-1\}$.

3. Proposed Scheme

3.1 Model

Our model is a hierarchical tree structure, as shown in **Fig. 1**. In our model, a large group is divided into several subgroups which are independent of each other. Each subgroup is managed by a subgroup key manager (*SGKM*), and then all *SGKM*s are managed by a group key generator center (*GKGC*), which is responsible for generating, distributing and updating group keys for secure communications by all group members.

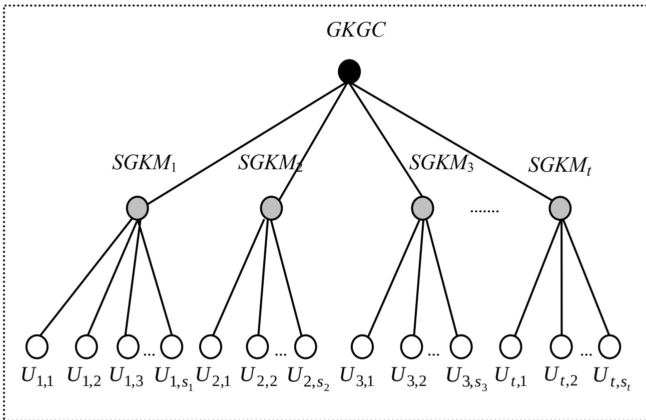


Fig. 1. The illustrations of a hierarchical tree structure

The hierarchical model described above is especially suitable for secure multicast communications in some "client-server" networks, such as e-commerce systems, where the *GKGC* is the electronic trade center or server center, each *SGKM* is a region agent, and all members are clients. In order to decrypt the encryption messages broadcasted by the center, all authorized clients (i.e., members) need to subscribe to a shared group key with their respective region agents in advance.

3.2 Group Initialization

In the following protocols, we assume that the *GKGC* manages *t* *SGKM*s and each *SGKM*$_i$ has s_i members ($1 \le i \le t$), as shown in **Fig. 1**. Group initialization consists of two main processes: the Initial Parameters Generation, and the *SGKM*s and Members Registration. The detailed description is as follows:

The Initial Parameters Generation. The *GKGC* first selects a large prime p and a secure one-way hash function $H(\cdot)$ and announces them to all group members publicly. Then, he privately generates an $m \times n$ matrix $\mathbf{A}$ ($1 < m \le n$ and $n > t$) and another m-dimensional column vector $\vec{\mathbf{y}}$ over Z_p, such that there exist at least p solutions of the linear equations $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$ over Z_p (i.e., it implies $r(\mathbf{A}) < n$). The detailed algorithm of generating the matrix $\mathbf{A}$ and vector $\vec{\mathbf{y}}$ is as follows.

Algorithm of generating $\mathbf{A}$ and $\vec{\mathbf{y}}$

1. Randomly generate an $m \times n$ matrix $\mathbf{A}$ over Z_p.
2. Verify that $r(\mathbf{A}) < n$ using Gaussian elimination method, or else goto 1 and restart.
3. Randomly generate an n-dimensional vector $\vec{\mathbf{x}}_0$ over Z_p.
4. Compute $\vec{\mathbf{y}} = \mathbf{A}\vec{\mathbf{x}}_0$. //It guarantees that $r(\mathbf{A}) = r(\overline{\mathbf{A}})$ in $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$.
5. Output ($\mathbf{A}$ and $\vec{\mathbf{y}}$).

// Therefore, there exist at least p solutions of the linear equations $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$ over Z_p.

Similarly, each *SGKM*$_i$ ($1 \le i \le t$) privately generates an $m_i \times n_i$ matrix $\mathbf{A}_i$ ($1 < m_i \le n_i$ and $n_i > s_i$) and another m_i-dimensional column vector $\vec{\mathbf{y}}_i$ over Z_p, such that there exist at least p solutions of the linear equations $\mathbf{A}_i\vec{\mathbf{x}} = \vec{\mathbf{y}}_i$ over Z_p.

The *SGKM*s and Members Registration. Furthermore, each *SGKM* is required to register at the *GKGC* as a legal region agent. During the *SGKM*s registration, the *GKGC* generates a unique n-dimensional column vector $\vec{\mathbf{x}}_i$ for each *SGKM*$_i$ ($1 \le i \le t$), such that $\vec{\mathbf{x}}_i$ satisfies the equation of $\mathbf{A}\vec{\mathbf{x}}_i = \vec{\mathbf{y}}$ (that is, $\vec{\mathbf{x}}_i$ is a solution of the linear equations, $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$), and then secretly sends $\vec{\mathbf{x}}_i$ to the *SGKM*$_i$ as his auxiliary secret. Similarly, each member is required to register at their respective *SGKM*$_i$ for subscribing the key distribution service. During the members registration, the *SGKM*$_i$ generates a unique n_i-dimensional column vector $\vec{\mathbf{x}}_{i,j}$ for each member $U_{i,j}$ ($1 \le j \le s_i$), such that $\vec{\mathbf{x}}_{i,j}$ satisfies the equation of $\mathbf{A}_i\vec{\mathbf{x}}_{i,j} = \vec{\mathbf{y}}_i$ (that is, $\vec{\mathbf{x}}_{i,j}$ is a solution of the linear equations, $\mathbf{A}_i\vec{\mathbf{x}} = \vec{\mathbf{y}}_i$), and then secretly sends $\vec{\mathbf{x}}_{i,j}$ to the member $U_{i,j}$ as his/her auxiliary secret.

3.3 The Group Key Generation and Distribution

The group key generation and distribution protocol (called *GKG&D* protocol hereafter) includes the following five steps:

Step 1. The *GKGC* randomly selects an m-dimensional column vector $\vec{\mathbf{k}}$ over Z_p and computes $\vec{\mathbf{r}} = \vec{\mathbf{k}}^T\mathbf{A}$. Then the *GKGC* broadcasts $\vec{\mathbf{r}}$ to all *SGKM*s.

Step 2. Furthermore, the *GKGC* computes $gk = \vec{\mathbf{k}} \cdot \vec{\mathbf{y}}$ and $H(gk)$. Here $\vec{\mathbf{k}} \cdot \vec{\mathbf{y}}$ denotes the inner product of the vectors $\vec{\mathbf{k}}$ and $\vec{\mathbf{y}}$, that is, $\vec{\mathbf{k}} \cdot \vec{\mathbf{y}} = \sum_{i=1}^{m} k_i y_i$, where $\vec{\mathbf{k}}^T = (k_1, k_2, ..., k_m)$ and

$\vec{\mathbf{y}}^T = (y_1, y_2, ..., y_m)$. Suppose that there is a public server at the *GKGC*, which is utilized to publish $H(gk)$ timely. In addition, we assume that all *SGKM*s and group members can only browse $H(gk)$ from the public server, but not modify it.

Step 3. Subsequently, each *SGKM* computes $gk_i = \vec{\mathbf{r}}^T \cdot \vec{\mathbf{x}}_i$ and verifies its authenticity by the equation of $H(gk_i) \overset{?}{=} H(gk)$. If the equation is true (i.e., $gk_i = gk$), the $SGKM_i$ randomly selects $m_i - 1$ integers, k_{i1}, k_{i2}, …, $k_{i(m_i-1)}$, over Z_p, and then computes and further gets k_{im_i} by the equation of $\vec{\mathbf{k}}_i \cdot \vec{\mathbf{y}}_i = gk_i$ (i.e., $k_{i1}y_{i1} + k_{i2}y_{i2} + ... + k_{i(m_i-1)}y_{i(m_i-1)} + k_{im_i}y_{im_i} = gk_i$), where $\vec{\mathbf{k}}_i^{\,T} = (k_{i1}, k_{i2}, ..., k_{i(m_i-1)}, k_{im_i})$ and $\vec{\mathbf{y}}_i^{\,T} = (y_{i1}, y_{i2}, ..., y_{i(m_i-1)}, y_{im_i})$; or else, this process ends up in failure.

Step 4. Each $SGKM_i$ computes $\vec{\mathbf{r}}_i = \vec{\mathbf{k}}_i^{\,T}\mathbf{A}_i$ and broadcasts $\vec{\mathbf{r}}_i$ to his/her all members.

Step 5. After receiving the broadcasted messages from the $SGKM_i$, each member $U_{i,j}$ $(1 \le j \le s_i)$ computes $uk_{i,j} = \vec{\mathbf{r}}_i^T \cdot \vec{\mathbf{x}}_{i,j}$ and verifies whether the equation of $H(uk_{i,j}) \overset{?}{=} H(gk)$ is correct. If it is correct, then he/she will believe that the shared group key is $uk_{i,j}$ indeed (i.e., $uk_{i,j} = gk$); or else, this process ends up in failure.

The correctness proofs:

Suppose that $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$ and $\vec{\mathbf{x}}_i^{\,T} = (x_{i1}, x_{i2}, ..., x_{in})$. Since $\vec{\mathbf{r}} = \vec{\mathbf{k}}^T\mathbf{A}$ and $gk_i = \vec{\mathbf{r}}^T \cdot \vec{\mathbf{x}}_i$, it gives

$$
\begin{aligned}
gk_i &= (\vec{\mathbf{k}}^{\mathbf{T}}\mathbf{A})^T \cdot \vec{\mathbf{x}}_i \\
&= [(k_1, k_2, ..., k_m)\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}]^T \cdot \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{pmatrix} \\
&= \begin{pmatrix} k_1a_{11} + k_2a_{21} + ... + k_ma_{m1} \\ k_1a_{12} + k_2a_{22} + ... + k_ma_{m2} \\ \vdots \\ k_1a_{1n} + k_2a_{2n} + ... + k_ma_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{in} \end{pmatrix} \\
&= (k_1a_{11} + k_2a_{21} + ... + k_ma_{m1})x_{i1} + (k_1a_{12} + k_2a_{22} + ... + k_ma_{m2})x_{i2} \\
&\quad + ... + (k_1a_{1n} + k_2a_{2n} + ... + k_ma_{mn})x_{in} \\
&= k_1(a_{11}x_{i1} + a_{12}x_{i2} + ... + a_{1n}x_{in}) + k_2(a_{21}x_{i1} + a_{22}x_{i2} + ... + a_{2n}x_{in}) \\
&\quad + ... + k_m(a_{m1}x_{i1} + a_{m2}x_{i2} + ... + a_{mn}x_{in}) \\
&= k_1y_1 + k_2y_2 + ... + k_my_m \ \text{(by } \mathbf{A}\vec{\mathbf{x}}_i = \vec{\mathbf{y}} \text{)}. \qquad (1)
\end{aligned}
$$

In addition,

$$gk = \vec{\mathbf{k}} \cdot \vec{\mathbf{y}} = \begin{pmatrix} k_1 \\ k_2 \\ \vdots \\ k_m \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = k_1 y_1 + k_2 y_2 + ... + k_m y_m . \tag{2}$$

By Eqs.1 and 2, it is obvious that $gk_i = gk$ for $i = 1$ *to* t, that is, $gk_1 = gk_2 = \cdots = gk_t = gk$.

Similarly, let $\mathbf{A}_i = \begin{bmatrix} a_{11}^i & a_{12}^i & \cdots & a_{1n_i}^i \\ a_{21}^i & a_{22}^i & \cdots & a_{2n_i}^i \\ \vdots & \vdots & \vdots & \vdots \\ a_{m_i 1}^i & a_{m_i 2}^i & \cdots & a_{m_i n_i}^i \end{bmatrix}$ and $\vec{\mathbf{x}}_{i,j}{}^T = (x_{i,j,1}, x_{i,j,2}, ..., x_{i,j,n_i})$. Since $\vec{\mathbf{r}}_i = \vec{\mathbf{k}}_i{}^T \mathbf{A}_i$ and $uk_{i,j} = \vec{\mathbf{r}}_i^T \cdot \vec{\mathbf{x}}_{i,j}$, then

$$\begin{aligned} uk_{i,j} &= (\vec{\mathbf{k}}_i{}^{\mathbf{T}} \mathbf{A}_i)^T \cdot \vec{\mathbf{x}}_{i,j} \\ &= [(k_{i1}, k_{i2}, ..., k_{im_i}) \begin{bmatrix} a_{11}^i & a_{12}^i & \cdots & a_{1n_i}^i \\ a_{21}^i & a_{22}^i & \cdots & a_{2n_i}^i \\ \vdots & \vdots & \vdots & \vdots \\ a_{m_i 1}^i & a_{m_i 2}^i & \cdots & a_{m_i n_i}^i \end{bmatrix}]^T \cdot \begin{pmatrix} x_{i,j,1} \\ x_{i,j,2} \\ \vdots \\ x_{i,j,n_i} \end{pmatrix} \\ &= \begin{pmatrix} k_{i1} a_{11}^i + k_{i2} a_{21}^i + ... + k_{im_i} a_{m_i 1}^i \\ k_{i1} a_{12}^i + k_{i2} a_{22}^i + ... + k_{im_i} a_{m_i 2}^i \\ \vdots \\ k_{i1} a_{1n_i}^i + k_{i2} a_{2n_i}^i + ... + k_{im_i} a_{m_i n_i}^i \end{pmatrix} \cdot \begin{pmatrix} x_{i,j,1} \\ x_{i,j,2} \\ \vdots \\ x_{i,j,n_i} \end{pmatrix} \\ &= (k_{i1} a_{11}^i + k_{i2} a_{21}^i + ... + k_{im_i} a_{m_i 1}^i) x_{i,j,1} + (k_{i1} a_{12}^i + k_{i2} a_{22}^i + ... + \\ &\quad k_{im_i} a_{m_i 2}^i) x_{i,j,2} + ... + (k_{i1} a_{1n_i}^i + k_{i2} a_{2n_i}^i + ... + k_{im_i} a_{m_i n_i}^i) x_{i,j,n_i} \\ &= k_{i1} (a_{11}^i x_{i,j,1} + a_{12}^i x_{i,j,2} + ... + a_{1n_i}^i x_{i,j,n_i}) + k_{i2} (a_{21}^i x_{i,j,1} + a_{22}^i x_{i,j,2} + \\ &\quad ... + a_{2n_i}^i x_{i,j,n_i}) + ... + k_{im_i} (a_{m_i 1}^i x_{i,j,1} + a_{m_i 2}^i x_{i,j,2} + ... + a_{m_i n_i}^i x_{i,j,n_i}) \\ &= k_{i1} y_{i1} + k_{i2} y_{i2} + ... + k_{im_i} y_{im_i} \text{ (by } \mathbf{A}_i \vec{\mathbf{x}}_{i,j} = \vec{\mathbf{y}}_i \text{)} , \end{aligned} \tag{3}$$

$$gk_i = \vec{\mathbf{k}}_i \cdot \vec{\mathbf{y}}_i = \begin{pmatrix} k_{i1} \\ k_{i2} \\ \vdots \\ k_{im_i} \end{pmatrix} \cdot \begin{pmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{im_i} \end{pmatrix} = k_{i1} y_{i1} + k_{i2} y_{i2} + ... + k_{im_i} y_{im_i} \tag{4}$$

Thus, there must exist $uk_{i,1} = uk_{i,2} = \cdots = uk_{i,s_i} = gk_i$ for $i = 1$ *to* t. Please note that the above computations are all over Z_p.

To sum up, $uk_{i,j} = gk$ for $j = 1$ *to* s_i and $i = 1$ *to* t. That is, all group members obtain a shared group key gk.

3.4 Rekeying

In order to maintain the forward secrecy and backward secrecy, a rekeying procedure must be executed when the *GKGC* withdraws/adds any *SGKM* or any member joins/leaves the group.

Withdrawing any *SGKM*s. Suppose that the *GKGC* wants to withdraw the *j*th *SGKM* (i.e., $SGKM_j$, $1 \le j \le t$). Then the group key must be updated. The rekeying procedure includes two steps as follows:

In the first step, the *GKGC* needs to renews his auxiliary secrets $\mathbf{A}$ and $\vec{\mathbf{y}}$. He first founds the linear equations as Eq.5 by all $\vec{\mathbf{x}}_i$ s of the remaining *SGKM*s, and then recomputes $\mathbf{A}$ and $\vec{\mathbf{y}}$ as new unknowns by using Gaussian elimination method in Eq.5. Please note that $\vec{\mathbf{x}}_1$, $\vec{\mathbf{x}}_2, \ldots, \vec{\mathbf{x}}_{j-1}, \vec{\mathbf{x}}_{j+1}, \ldots, \vec{\mathbf{x}}_t$ are $t-1$ known vectors in the following linear equations.

$$\begin{cases} \mathbf{A}\vec{\mathbf{x}}_1 = \vec{\mathbf{y}} \\ \mathbf{A}\vec{\mathbf{x}}_2 = \vec{\mathbf{y}} \\ \quad\vdots \\ \mathbf{A}\vec{\mathbf{x}}_{j-1} = \vec{\mathbf{y}} \\ \mathbf{A}\vec{\mathbf{x}}_{j+1} = \vec{\mathbf{y}} \\ \quad\vdots \\ \mathbf{A}\vec{\mathbf{x}}_t = \vec{\mathbf{y}} \end{cases} \tag{5}$$

Since the matrix $\mathbf{A}$ is $m \times n$ dimension and the vector $\vec{\mathbf{y}}$ is m dimension, obviously there exist $m \times n + m$ unknown variables while there are only $(t-1) \times m$ equations in Eq.5. Thus there must be infinitely many solutions of $\mathbf{A}$ and $\vec{\mathbf{y}}$ in Eq.5 because the number of unknown values is far more than that of equations. Furthermore, the *GKGC* computes a new particular solution of $\mathbf{A}$ and $\vec{\mathbf{y}}$ by Eq.5, which is different from the old solution, such that $\mathbf{A}\vec{\mathbf{x}}_i = \vec{\mathbf{y}}$ ($i \ne j$) but $\mathbf{A}\vec{\mathbf{x}}_j \ne \vec{\mathbf{y}}$.

In the second step, the *GKGC* regenerates the group key gk based on the new auxiliary secrets $\mathbf{A}$ and $\vec{\mathbf{y}}$ by executing the *GKG&D* protocol again. That is, the *GKGC* randomly selects a new vector $\vec{\mathbf{k}}$ over Z_p, recomputes $\vec{\mathbf{r}} = \vec{\mathbf{k}}^T\mathbf{A}$ and $gk = \vec{\mathbf{k}} \cdot \vec{\mathbf{y}}$, broadcasts $\vec{\mathbf{r}}$ to all remaining *SGKM*s, and renews $H(gk)$ in the public server. Furthermore, each remaining $SGKM_i$ computes and verifies their respective subgroup key gk_i by the new broadcasted message from the *GKGC*. At last, all members obtain the new group key gk by the same method as the initial group key generation and distribution in the *GKG&D* protocol.

Adding a new *SGKM*. Suppose that the new *SGKM* is marked as $SGKM_{t+1}$. Similarly, the group key must be updated. Since $t < n$, so $t+1 \le n$. After adding a new *SGKM*, it still satisfies the property that the column number of the matrix $\mathbf{A}$ is greater than or equal to that of the *SGMK*s. So, when a new $SGKM_{t+1}$ requests to join the group, the *GKGC* only needs to generate another unique $\vec{\mathbf{x}}_{t+1}$ ($\mathbf{A}\vec{\mathbf{x}}_{t+1} = \vec{\mathbf{y}}$) and secretly sends it to the $SGKM_{t+1}$ while other secret $\vec{\mathbf{x}}_i$ s are unaltered. Then the *GKGC* again executes the *GKG&D* protocol to update the group key.

Please note that it must satisfy $t \le n$ in this dynamic protocol in order to prevent the collusion attacks (see Theorem. 5). In case of $t = n$, if adding a new *SGMK*, it needs to regenerate $\mathbf{A}$ by using the similar method as withdrawing any *SGMK*s, such that the column number of the matrix $\mathbf{A}$ is greater than that of the *SGMK*s.

Removing any member: Suppose that a member $U_{i,j}$ of the *SGKM$_i$* requests to leave the group. After confirming his leaving, the *SGKM$_i$* needs to renew old secrets { $\mathbf{A}_i$, $\vec{\mathbf{y}}_i$ } in order to protect backward secrecy. Similarly, the *SGKM$_i$* first creates the similar linear equations like Eq.5 by the secrets ($\vec{\mathbf{x}}_{i,j}$s) of all remaining members and then recomputes { $\mathbf{A}_i$, $\vec{\mathbf{y}}_i$ } as unknowns. Furthermore, the *SGKM$_i$* requests the *GKGC* to update the group key due to his member leaving. At last, the *GKGC* again executes the *GKG&D* protocol to renew the group key.

Adding a new member: Suppose that a new member, U_{i,s_i+1} , requests to join the subgroup of the *SGKM$_i$*. After receiving a join request message of U_{i,s_i+1}, the *SGKM$_i$* first performs the register procedure to verify the identity of new member. If the *SGKM$_i$* agrees his join request, the *SGKM$_i$* generates another unique $\vec{\mathbf{x}}_{i,s_i+1}$ ($\mathbf{A}_i\vec{\mathbf{x}}_{i,s_i+1} = \vec{\mathbf{y}}_i$) and secretly sends it to U_{i,s_i+1} . Similarly, the *SGKM$_i$* again requests the *GKGC* to update the group key due to his new member joining. At last, the *GKGC* executes the *GKG&D* protocol to renew the group key.

In addition, in order to reduce the overhead of high frequent joins and leaves, we can consider rekeying in a batch as the method in the literature [5].

4. Analysis

4.1 Security Analysis

We have proved the correctness of the above proposed scheme. Furthermore, we focus on their security analysis, which sees Theorem 3, 4 and 5 in detail.

Theorem 3. The proposed scheme achieves four security goals of group key management: 1) Group Key Confidentiality, 2) Forward Secrecy, 3) Backward Secrecy, 4) Group key authentication.

Proof. 1) Group Key Confidentiality is guaranteed by the security of the public message $H(gk)$ and the broadcast message $\vec{\mathbf{r}}$. Here, we assume that $H(\cdot)$ is a secure one-way hash function. Therefore, any unauthorized users cannot obtain the group key gk only from the public message $H(gk)$. Similarly, for any unauthorized users, he/she cannot get the group key gk only from the broadcast message $\vec{\mathbf{r}}$, because $gk = \vec{\mathbf{r}}^T \cdot \vec{\mathbf{x}}_i$ or $gk = \vec{\mathbf{k}} \cdot \vec{\mathbf{y}}$, where $\vec{\mathbf{x}}_i$ and $\vec{\mathbf{y}}$ are unknown, and $\vec{\mathbf{k}}$ is randomly and secretly generated by the *GKGC*. 2) Forward Secrecy is guaranteed by the rekeying procedure. Whenever an *SGKM*, or a member, leaving the group, it needs to update not only the group key but also the auxiliary secrets { $\mathbf{A}$, $\vec{\mathbf{y}}$ }, or { $\mathbf{A}_i$, $\vec{\mathbf{y}}_i$ }. Thus, the leaving *SGKM* or member cannot learn about new group keys after he leaves the group since $\mathbf{A}\vec{\mathbf{x}}_j \neq \vec{\mathbf{y}}$ ($\mathbf{A}_i\vec{\mathbf{x}}_{i,j} \neq \vec{\mathbf{y}}_i$). 3) Backward Secrecy is guaranteed by the fact that the group key is always updated whenever new *SGKM* or member joining the group. 4) Key Authentication is provided through the value of $H(gk)$ generated by the *GKGC* who owns the secrets { $\mathbf{A}$, $\vec{\mathbf{y}}$ }. For any active attacker, it is impossible to forge a broadcast vector $\vec{\mathbf{r}}^*$ without the secrets $\mathbf{A}$ and $\vec{\mathbf{y}}$, such that $H(\vec{\mathbf{r}}^{*T} \cdot \vec{\mathbf{x}}_i) = H(gk)$ (i.e., $\vec{\mathbf{r}}^{*T} \cdot \vec{\mathbf{x}}_i = gk$) for all i . Please note that $H(gk)$ is published at the public server of the *GKGC*, and can only be modified by the *GKGC*.

Theorem 4 (Outsider attack). Our scheme is secure against outsider attack.

Proof. 1) Firstly, we assume that an outsider active attacker who impersonates the *GKGC* to broadcast a forged key or rekeying message in order to share a group key with all group members. As the above analysis in Theorem 3, it is impossible for the attacker to successfully pass the authenticity verification (i.e., for a forged $\vec{\mathbf{r}}^*$, obviously it must be $H(\vec{\mathbf{r}}^{*T}\cdot\vec{\mathbf{x}}_i)\neq H(gk)$) because he can't modified $H(gk)$ at the public server of the *GKGC*. Secondly, we assume that an outsider active attacker who impersonates a group member for requesting a group key service. In our scheme, each member needs to beforehand subscribe the group key service, and then obtains the respective secret $\vec{\mathbf{x}}_{i,j}$ which is shared with the manager, *SGKM*$_i$. Thus, legal group key service requests from group members can be authenticated by their respective secret $\vec{\mathbf{x}}_{i,j}$. At last, the attacker cannot obtain any secret information of the group key direct from the broadcasted key messages due to the confidentiality of group keys analyzed above in Theorem 3. 2) In addition, the value of $\vec{\mathbf{r}}$ are obviously different for every rekeying process because $\vec{\mathbf{k}}$ is randomly generated, and thus our scheme is secure against the replay attack. Therefore, our scheme is secure against outsider attack.

Theorem 5 (Insider attack). Our scheme is secure against insider attack.

Proof. For each *SGMK*$_i$, he only knows his secret vector $\vec{\mathbf{x}}_\mathbf{i}$. By his secret vector $\vec{\mathbf{x}}_\mathbf{i}$ and the broadcasted vector $\vec{\mathbf{r}}$, obviously, *SGMK*$_i$ can obtain the value of gk_i by computing $gk_i=\vec{\mathbf{r}}^T\cdot\vec{\mathbf{x}}_i$, which is the shared group key (i.e., $gk_i=gk$). However, he cannot get any secret information about the matrix $\mathbf{A}$ and the vector $\vec{\mathbf{y}}$, because $\vec{\mathbf{k}}^T=(k_1,k_2,...,k_m)$ is randomly and secretly selected by the *GKGC*. Similarly, each member cannot obtain any secret information about the matrix $\mathbf{A}_i$ and the vector $\vec{\mathbf{y}}_i$. Furthermore, our scheme is secure against the collusion attacks of the insider *SGKM*s or members. Especially, we assume that all t *SGKM*s try to get the secrets of the *GKGC* (i.e., $\mathbf{A}$ and $\vec{\mathbf{y}}$) with colluding each other. In order to achieve this aim, they collude to found the following equations by their respective secret $\vec{\mathbf{x}}_i$ s:

$$\begin{cases}\mathbf{A}\vec{\mathbf{x}}_1=\vec{\mathbf{y}}\\ \mathbf{A}\vec{\mathbf{x}}_2=\vec{\mathbf{y}}\\ \quad\vdots\\ \mathbf{A}\vec{\mathbf{x}}_t=\vec{\mathbf{y}}\end{cases}\tag{6}$$

However, for these colluding *SGKM*s, there are $m\times n+m$ unknown variables while there are only $t\times m$ equations ($t\leq n$) in Eq.6. Thus, they do not obtain any secret information of $\mathbf{A}$ and $\vec{\mathbf{y}}$ only from Eq.6 based on the basic theories for solving linear equations. Similarly, all subgroup members cannot get any secret information about $\mathbf{A}_i$ and $\vec{\mathbf{y}}_i$ yet. In fact, even if all *SGKM*s and all group members collude to perform this attack they cannot obtain the secrets of the *GKGC*, and furthermore they cannot impersonate the *GKGC* to authorize a new *SGKM* or update group keys.

4.2 Performance Analysis

By Theorem 5, there are at most n *SGKM*s in our proposed scheme in order to resist the collusion attacks of all *SGKM*s. In the following section, we assume that there are just n *SGKM*s in a group and each *SGKM* also has n group members. Thus, there are total n^2 group members in a group.

In our proposed scheme, whenever a member joins the group, the *SGKM*$_i$ only needs to generate new member's secret $\vec{\mathbf{x}}_{i,j}$ based on his secret linear equations, and further requests the *GKGC* to update the group key. Whenever a member leaves the group, the $SGKM_i$ only needs to renew his auxiliary secret ($\mathbf{A}_i$, $\bar{\mathbf{y}}_i$) based on the known linear equations, and further requests the *GKGC* to update the group key. In the updating the group key (i.e., *GKG&D*) procedure, it only takes one multiplication of the vector and the matrix for the *GKGC*, two inner products for the $SGKM_i$, and one inner product for each member, respectively, instead of other complex cryptographic operations. **Table 1** shows the main computation costs of LKH[1,2], OFT[3], ELK[4], HL[9] and our proposed scheme, where ***E***, ***D***, ***R***, ***H***, ***F***, ***P***, ***M***, ***S*** denote the computation costs of encryption, decryption, random key generation, hashing, pseudo-random function, the *N*-degree interpolating polynomial, scalar multiplication and a particular solution of the linear equations, respectively. From **Table 1**, the most complex computation of our scheme is to solve a particular solution of the linear equations. Please note that it is not to compute all general solutions. As we know, it is easier to compute a particular solution than the general solution. Compared with other group key management schemes, obviously the computation costs of our scheme are lower, especially for group members.

Table 1. The main computation costs of LKH, OFT, ELK, HL and our proposed scheme

		LKH	OFT	ELK	HL	Ours
Join	*GKGC*	(2***E***+***R***)log*N*	(2***E***+2***H***+***F***)log*N*	***E***log*N*+(2*N*-1)***F***+***R***	*N****P***+1***R***+1***H***	(*mn*+*n*)***M***+1***H***
	SGKM	-	-	-	-	1***S***+2*n****M***+1***H***
	Old member	***D*** log*N*	***D*** log*N*	***D*** log*N*	1***P***+1***H***	*n****M***+1***H***
	New member	***D*** log*N*	(***D***+***H***) log*N*	2***F***log*N*	1***P***+1***H***	*n****M***+1***H***
Leave	*GKGC*	(2***E***+***R***)log*N*	(***E***+***H***+***F***)log*N*	(2***E***+7***F***)log*N*	*N****P***+1***R***+1***H***	(*mn*+*n*)***M***+1***H***
	SGKM	-	-	-	-	1***S***+2*n****M***+1***H***
	Member	***D*** log*N*	(***D***+***F***) log*N*	(***D***+4***F***)log*N*	1***P***+1***H***	*n****M***+1***H***

Table 2. The communication and storage costs of LKH, OFT, ELK, HL and our proposed scheme

		LKH	OFT	ELK	HL	Ours
Communication (broadcast)	Join	*2logN****L***	*logN****L***	0	*(2N+1)****L***	*n****L***
	Leave	*2logN****L***	*logN****L***	*logN****L***	*(2N+1)****L***	*n****L***
Storage	*GKGC*	*(2N-1)****L***	*(2N-1)****L***	*(2N-1)****L***	*2N****L***	(*m*+*mn*+*nn*)***L***
	SGKM	-	-	-	-	(*m*+*n*+*mn*+*nn*)***L***
	Member	*logN****L***	*(logN+1)****L***	*logN****L***	*2****L***	*n****L***

Furthermore, in our proposed scheme, for *GKGC*, the communication cost is mainly used to broadcast key or rekeying. The size of the broadcasted key or rekeying message (mainly including an n -dimensional vector $\bar{\mathbf{r}}$) is *n**L***, where the constant ***L*** is the size of the group key. In addition, each *SGKM* needs to broadcast an *n**L***-size message to his group members in order to transfer the group key. For the storage cost, it involves three kinds of participants: *GKGC*, *SGKM* and group member in our scheme. For *GKGC*, *SGKM* and group member, it needs to store $\{\mathbf{A}, \bar{\mathbf{y}}$ and all $\bar{\mathbf{x}}_i$ s}, $\{\bar{\mathbf{x}}_i$, $\mathbf{A}_i$, $\bar{\mathbf{y}}_i$ and all $\bar{\mathbf{x}}_{i,j}$ s}, and $\{\bar{\mathbf{x}}_{i,j}\}$, respectively. The detailed comparisons are listed in **Table 2**. In addition, please note that we can let $m = 2$ $(1 < m \le n)$ to reduce the storage cost.

In addition, the proposed scheme can be easily and naturally extended into the single-level and multi-level architecture, as shown in **Fig. 2** and **Fig. 3**, respectively. In **Fig. 2**, the *GKGC* directly utilize a linear equation ($\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$) to distribute the group keys to all group members. As the literature [9], the single-level architecture is only suitable for a group with a small group size. Assume that there are N group members in **Fig. 2**. Then the *GKGC* needs to broadcast a message containing N elements to all group members, which is lower than the HL scheme [9] (including N points). Especially, our scheme has better computation complexity than their scheme because each member only needs to compute N scalar multiplications (i.e., one inner product) instead of an N-degree interpolating polynomial. In **Fig. 3**, each internal node uses a secret linear equation ($\mathbf{A}_i\vec{\mathbf{x}} = \vec{\mathbf{y}}_i$) to transfer the group key message from his parent node to his all child nodes, where the internal nodes can also be group members (i.e., group members play the part of the internal nodes). When any group member or internal node joins/leaves the group, the number of auxiliary secrets required to be updated in the secret tree is $O(1)$ instead of $O(\log N)$, which is just required in the key tree methods, such as LKH, OFT and ELK.

Furthermore, in order to extend more group members, we may first build multiple groups managed by different *GKGC*s using the method proposed above, respectively, and further combine these groups into a larger group using a well-known distribution key agreement scheme (see **Fig. 4**), such as BD [12], TDH [14] and TGDH [15]. Thus it becomes a hybrid group key management scheme. In this hybrid scheme, all *GKGC*s first agree a group key using a distribution key agreement scheme and then propagate it to their respective group members using the proposed distribution method above.

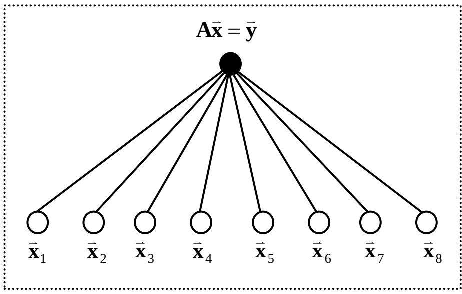


Fig. 2. The single-level auxiliary secret tree

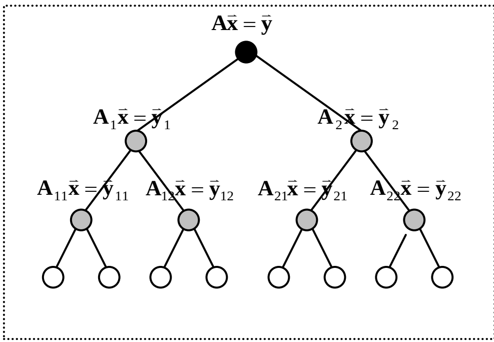


Fig. 3. The multi-level auxiliary secret tree

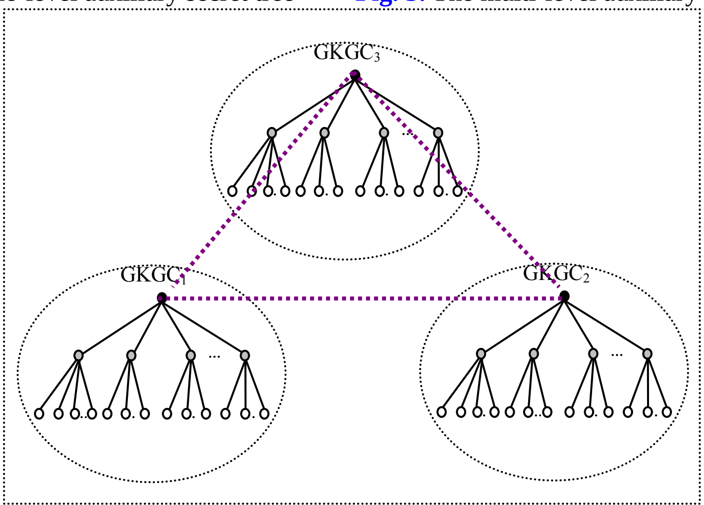


Fig. 4. A hybrid group key management scheme

5. Conclusion

We have presented an authenticated group key management scheme, which is divided into two or multiple levels to achieve it based on the basic theories for solving linear equations. This scheme is suitable for secure multicast communications in some client-server networks due to its higher efficiency, flexibility and adaptability. Especially, our scheme has some good advantages as follows.

1) For the *GKGC*, when adding/removing any group member, he does not need to do anything else except updating the group key. Furthermore, when updating the group key it only needs to broadcast new group key information to all *SGKM*s. In addition, even if adding/removing any *SGKM* it is also easy to implement it because the most complex computation is to solve a particular solution of the linear equations, $\mathbf{A}\vec{\mathbf{x}} = \vec{\mathbf{y}}$.
2) For the *SGKM*s, it is very easy to recover the group keys, and further it only needs to broadcast different subgroup key information to their respective members when transferring the group key. In addition, when adding/removing any *SGKM* it does not need to update the remaining *SGKM*s' auxiliary secrets (i.e., $\vec{\mathbf{x}}_i$ s).
3) For group members, it takes very low computation cost to recover the group key because it only needs to compute an inner product instead of other complex cryptographic operations, and when adding/removing any member it does not need to update the remaining members' auxiliary secrets (i.e., $\vec{\mathbf{x}}_{i,j}$ s).
4) Our scheme provides authenticated information used for the authentication of the group key without employing symmetric and asymmetric cryptographic operations.

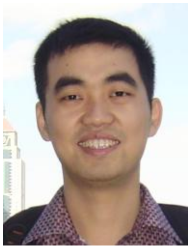
References

[1] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: Issues and architecture," *IETF RFC* 2627, June 1999. Article (CrossRef Link)
[2] C.K. Wong, M.G. Gouda, and S.S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol.8, no.1, pp.16–30, 2000. Article (CrossRef Link)
[3] M. Sheng, J. Zhu, and G. Cui, "A hybrid group key management scheme for two-layered Ad Hoc networks," in *Proc. of 9th International Conferences on Information Technology* (*ICIT'06*), IEEE Computer Society, India, pp.83-84, 2000. Article (CrossRef Link)
[4] A. Perrig, D.X. Song, and J.D. Tygar, "Elk, a new protocol for efficient large-group key distribution," in *Proc. of IEEE Symposium on Security and Privacy*, pp. 247–262, 2001. Article (CrossRef Link)
[5] X.S.Li, Y.R. Yang, M.G. Gouda, and S.S. Lam, "Batch rekeying for secure group communications," in *Proc. of International World Wide Web Conference* (*WWW*). pp. 525–534, 2001. Article (CrossRef Link)
[6] Y.R. Chen, J.D. Tygar and W.G. Tzeng, "Secure Group Key Management Using Uni-Directional Proxy Re-Encryption Schemes," in *Proc. of IEEE INFOCOM 2011*, pp. 1952-1960, 2011. Article (CrossRef Link)
[7] J.A.M. Naranjo, N. Antequera, L.G. Casado, J.A. Lopez-Ramos, "A suite of algorithmes for key distribution and authentication in centralized secure multicast environments," *Journal of Computational and Applied Mathematics*, vol. 236, no.12, pp.3042-3051, 2012. Article (CrossRef Link)
[8] P. Vijayakumar, S. Bose, A. Kannan, "Centralized key distribution protocol using the greatest common divisor method," *Computer & Mathematics with Applications*, vol.65, no.9, pp.1360-1368, 2013. Article (CrossRef Link)
[9] L. Harn and C. L. Lin, "Authenticated Group Key Transfer Protocol Based on Secret Sharing," *IEEE Transactions on Computers*, vol.59, no.6, pp.842-846, 2010. Article (CrossRef Link)

[10] W. Diffie, M. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol.22, no.6, pp.644–654, 1976. Article (CrossRef Link)
[11] I. Ingemarsson, D.T. Tang, C.K. Wong, "A conference key distribution system," *IEEE Transactions on Information Theory*, vol.28, no.5, pp.714–719, 1982. Article (CrossRef Link)
[12] M. Burmester, Y. Desmedt, "A secure and efficient conference key distribution system," *Advances in Cryptology – EUROCRYPT 1994*, Lecture Notes in Computer Science, vol. 950, pp. 275–286, 1994. Article (CrossRef Link)
[13] M. Steiner, G. Tsudik, M. Waidner, "Key agreement in dynamic peer groups," *IEEE Transactions on Parallel and Distributed Systems*, vol.11, no.8, pp.769–780, 2000. Article (CrossRef Link)
[14] A. Joux, "A One Round Protocol for Tripartite Diffie-Hellman," *Journal of Cryptology*, vol.17, no.4, pp.263-276, 2004. Article (CrossRef Link)
[15] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM T. Inf. and System Security* (*TISSEC*), vol.7, no.1, pp.60-96, 2004. Article (CrossRef Link)
[16] J. Katz, M. Yung, "Scalable protocols for authenticated group key exchange," *Advances in cryptology—CRYPTO 2003*. Lecture notes in computer science, vol.2729. Springer-Verlag, pp.110–25, 2003. Article (CrossRef Link)
[17] Q. Wu, Y. Mu, W. Susilo, B. Qin, J. Domingo-Ferrer, "Asymmetric Group Key Agreement," in *Proc. of EUROCRYPT 2009*, LNCS, vol.5479, Springer-Verlag, pp.153–170, 2009. Article (CrossRef Link)
[18] L. Zhang, Q.H. Wu, B. Qin, *et al.*, "Provably secure one-round identity-based authenticated asymmetric group key agreement protocol," *Information Sciences*, vol.181, no.19, pp.4318-4329, 2011. Article (CrossRef Link)
[19] L. Zhang, Q.H. Wu, B. Qin, *et al*., "Asymmetric group key agreement protocol for open networks and its application to broadcast encryption," *Computer Networks*, vol.55, no.15, pp.3246-3255, 2011. Article (CrossRef Link)
[20] S. Jarecki, J. Kim and G. Tsudik, "Flexible Robust Group Key Agreement," *IEEE Transactions on Parallel and Distributed Systems*, vol.22, no.5, pp.879-886, 2011. Article (CrossRef Link)
[21] X.X. Lv, H. Li and B.C. Wang, "Group key agreement for secure group communication in dynamic peer systems," *Journal of Parallel and Distributed Computing*, vol.72, no.10, pp.1195-1200, 2012. Article (CrossRef Link)
[22] A. Ballardie, "Scalable Multicast Key Distribution," *RFC 1949*, 1996. Article (CrossRef Link)
[23] S. Mittra, "Iolus: A framework for scalable secure multicasting," in *Proc. of the ACM SIGCOMM*, vol. 27, 4 (New York, Sept.) ACM, New York, pp.277–288, 1997. Article (CrossRef Link)
[24] L.R. Dondeti, S. Mukherjee, and A. Samal, "Scalable secure one-to-many group communication using dual encryption," *Computer Communications*, vol.23, no.17, pp.1681–1701, 2000. Article (CrossRef Link)
[25] 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. Article (CrossRef Link)
[26] D.W. Kwak, J.W. Kim, "A Decentralized Group Key Management Scheme for the Decentralized P2P Environment," *IEEE Communications Letters*, vol.11, no.6, pp.555-557, 2007. Article (CrossRef Link)
[27] N.T.T. Huyen, M. Jo, T.D. Nguyen and E.N. Huh, "A beneficial analysis of deployment knowledge for key distribution in wireless sensor networks," *Security and Communication Networks*, vol.5, no.5, pp.485-495, 2012. Article (CrossRef Link)
[28] D.H. Je, Y.H. Choi, S.W. Seo, "Subscription-Period-Aware Key Management for Secure Vehicular Multicast Communications," *IEEE Transaction on Vehicular Technology*, vol.62, no.9, pp.4213-4227, 2013. Article (CrossRef Link)

Run-hua Shi received the Ph.D. degree from University of Science and Technology of China in 2011. He is currently a Professor with Anhui University, and a visiting fellow at the School of Computing and Information Technology, University of Wollongong. His current research interest includes classical and quantum cryptography, in particular, privacy-preserving multiparty computation.

Hong Zhong received the Ph.D. degree from University of Science and Technology of China in 2005. She is currently a Professor with Anhui University. Her main research works focus on wireless network and network security.



Shun Zhang received the Ph.D. degree from Beijing Normal University in 2012. He is currently an associate professor with Anhui University. His current research interest includes computational complexity and quantum computation.