ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2012, Vol.41, No.2

Towards Scalable Key Management for Secure Multicast Communication

Yuh–Min Tseng[*,1], Chen–Hung Yu[1], Tsu–Yang Wu[2]

[1] *Department of Mathematics, National Changhua University of Education, Jin–De Campus, Chang–Hua City 500, Taiwan, R.O.C.*
e–mail: ymtseng@cc.ncue.edu.tw

[2] *School of Computer Science and Technology, Shenzhen Graduate School, Harbin Institute of Technology, Shenzhen 518055, P.R. China*

crossref *http://dx.doi.org/10.5755/j01.itc.41.2.846*

Abstract. Secure multicast communication allows a sender to deliver encrypted messages to a group of authorized receivers. A practical approach is that the sender uses a common key shared by the authorized receivers to encrypt the transmitted messages. The common key must be renewed to ensure forward/backward secrecy when group members leave/join the group, called the rekeying process. Thus, the rekeying problem is a critical issue for secure multicast communication. Many key management schemes have been proposed to improve the performance of the rekeying process. In 2010, Lin *et al*. proposed two key management schemes without the rekeying process. However, the transmission size required in their schemes increases linearly with the number of group members. In this article, we use the time-bound concept to propose two new key management schemes without the rekeying process. The point is that the required transmission size is constant. Performance analysis is given to demonstrate that our schemes have better performance as compared with the recently proposed key management schemes in terms of transmission size and computational cost. Under several security assumptions, we prove that the proposed schemes satisfy the requirements of secure multicast communication.

Keywords: multicast; key management; group communication; rekeying problem.

1. Introduction

With the rapid growth of the Internet and digital technologies, group communication has widely been used to many concrete applications such as distance education, multi-media streaming and pay-TV [1]. To achieve efficient group communication, multicast technique allows a sender to deliver messages to a group of authorized receivers. It can efficiently reduce the required network bandwidth. Since the Internet or wireless communications are operated on a public channel, the multicast messages must be encrypted to resist eavesdropping or unauthorized users obtaining the transmitted messages. A practical approach is to use a common key shared by all authorized users to encrypt the transmitted messages. For preventing the joining/left users from obtaining the previous/later messages, the common key must be renewed when the group membership is changed. Two requisite security requirements for secure multicast communication are defined as follows [2].

– *Forward secrecy*: When a member leaves the group, he/she should not be able to access the future multicast messages.
– *Backward secrecy*: When a member joins the group, he/she should not be able to access the past multicast messages.

In order to achieve forward/backward secrecy, the common key must be refreshed, called the rekeying process. However, it also incurs the 1–affect–*n* problem [3]. That is, when a member leaves or joins the group, all group members will be affected because the common key is held by each group member and must be updated. For a dynamic group, the highly joining/leaving frequency would cause highly computational burden for updating the common key.

For solving the 1–affect–*n* problem, many key management schemes for secure multicast communication have been proposed [3-11]. Mittra [3] proposed the notion of secure distribution tree to solve the scalability problem. In the proposed framework, the group controller (GC) arranged all group members to several smaller hierarchical subgroups, while the management authority of each subgroup is assigned to a subgroup controller (SGC). However, Mittra's scheme has a disadvantage that before the encrypted message reaches a group member, it must be

* Corresponding author

decrypted and encrypted repeatedly by the SGCs which locate on the transmission path. In such a case, it will cause the transmission delay. Since each SGC (or called intermediate node) can access the encrypted messages, it will incur a trusted problem whether the SGCs are trusted or not. In order to solve the trusted problem, Molva and Pannetrat [4] proposed a new key management scheme based on the distributed tree computation. In Molva and Pannetrat's scheme, all routers, network nodes or application proxies form a tree structure and they can be viewed as intermediate nodes of the distributed tree. All group members are arranged to several subgroups according to locations or functionalities, and each subgroup is linked to a leaf node in the tree structure. Each intermediate node is also responsible for a security function, but not a decrypting/encrypting function. Thus, the trusted problem of intermediate nodes will be resolved partially. However, Molva and Pannetrat's scheme did not solve the transmission delay and the trusted problem of intermediate nodes completely.

In 1999 and 2000, Wallner *et al*. [5] and Wong *et al*. [6] proposed a logical key hierarchy (LKH) tree approach, respectively. In their schemes, there is a centralized GC without any subgroup controllers. Thus, their schemes can avert from the transmission delay and the trusted problem of intermediate nodes. The root of the logical key hierarchy tree is viewed as a traffic encryption key (TEK) which is used to encrypt messages for multicast communications and shared by all group members. The group members are arranged to the leaf nodes of the LKH tree. Each leaf node is given a key encryption key (KEK) which is shared only for the group member and the GC. Each internal node of the LKH tree is also given a KEK in order to be used to encrypt the broadcast messages and smooth the way of the rekeying process. Each group member has to store all KEKs of the path from the root to its seat while memorizing the key path. In addition, the GC must keep all KEKs corresponding to each node of the LKH tree. When a member wants to join or leave the group, the keys of all nodes on the path from this leaving/joining member to the root have to be renewed to satisfy forward/backward secrecy. As a result, they reduced the transmission cost of the rekeying process from O(n) to O(log$n$), where n is the number of the group members.

Afterwards, many LKH–like schemes [7-11] were proposed to improve the performance of the rekeying process and the key storage requirement. In order to solve the inefficiency problem of the rekeying process, Li *et al*. [7] proposed a periodic batch rekeying method to reduce the overhead of the rekeying process. Sherman and McGrew [8] also proposed an optimization of the hierarchical binary tree. The Sherman and McGrew's contribution is to reduce the rekeying transmission cost from $2\log_2 n$ to $\log_2 n$. In 2003, Tseng [9] proposed a scalable key management scheme to reduce the key storage of the GC to a constant size. In addition, for the tree balancing problem, Goshi and Ladner [10] proposed a height-balanced 2-3 tree and presented that it has the best performance for the tree balancing strategies. However, to balance the 2-3 tree after member joining will involve the node-splitting problem. It requires 5h worst-case rekeying cost, where h is the height of the LKH tree. In 2005, Lu [11] proposed an NSBHO (Non–Split Balancing High–Order) tree, in which the NSBHO tree approach resolves the node-splitting problem. Nevertheless, these LKH–like schemes still require the rekeying process when group members join/leave the group.

In 2010, Lin *et al*. [12] proposed two key management schemes without the rekeying process. They described a star–based construction for multicast key management. However, the transmission size required in their schemes increases linearly with the number of group members. Thus, it inspires us to solve this transmission size problem. In this article, we propose two key management schemes without the rekeying process while the transmission size is constant. In the proposed schemes, we use the time–bound concept to solve the rekeying problem. The first scheme is suitable for group members with continuous time intervals. In the second scheme, it is suitable for discrete time intervals (or called non-continuous time intervals). Under several security assumptions, we shall prove that the proposed schemes satisfy the requirements of secure multicast communication. Performance analysis is given to demonstrate that our schemes have the better performance as compared with the recently proposed key management schemes in terms of transmission size and computational cost.

The remainder of this paper is organized as follows. Preliminaries are given in Section 2. In Section 3, we propose our key management schemes for secure multicast communication. Security analysis of the proposed schemes is presented in Section 4. In Section 5, we demonstrate performance analysis and comparisons with the previously proposed schemes. Conclusions are drawn in Section 6.

2. Preliminaries

In this section, we briefly review the concepts of bilinear pairings, the RSA cryptosystem, the Lucas function, and some security problems as well as assumptions.

2.1. Bilinear pairings and its security assumptions

Let G_1 and G_2 be two multiplicative cyclic groups of large prime order q, and let g be a generator of G_1. We say that the map $\hat{e}$: $G_1 \times G_1 \rightarrow G_2$ is an admissible bilinear map if it satisfies the following properties:

1. Bilinear: For all g_1, $g_2 \in G_1$ and x, $y \in Z_q^*$, $\hat{e}(g_1^x, g_2^y) = \hat{e}(g_1, g_2)^{xy}$.
2. Non–degenerate: There exist $g_1, g_2 \in G_1$ such that $\hat{e}(g_1, g_2) \neq 1$.

3. Computable: For all $g_1, g_2 \in G_1$, there is an efficient algorithm to compute $\hat{e}(g_1, g_2)$.

To prove the security of the proposed schemes, we define several security problems and assumptions for bilinear pairings defined on elliptic curves as follows.

- **Computational Diffie–Hellman (CDH) problem:** Given $g, g^a, g^b \in G_1$ for unknown $a, b \in Z_q^*$, the CDH problem is to compute $g^{ab} \in G_1$.
- **CDH assumption:** No probabilistic polynomial time (PPT) algorithm can solve the CDH problem with a non–negligible advantage.
- **Decision bilinear Diffie–Hellman (DBDH) problem:** Given $g, g^a, g^b, g^c, g^d \in G_1$, for some $a, b, c, d \in Z_q^*$, the DBDH problem is to distinguish $(g, g^a, g^b, g^c, g^d, \hat{e}(g, g)^{abc})$ from $(g, g^a, g^b, g^c, g^d, \hat{e}(g, g)^d)$.
- **DBDH assumption:** No PPT algorithm can solve the DBDH problem with a non–negligible advantage.

The detailed descriptions and security assumptions for bilinear pairings can be referred to [13-16].

2.2. Other security assumptions

[Integer factorization and RSA cryptosystem]

Given two large prime numbers p and q, it is easy to compute $n = p \cdot q$. Given n, however, no probabilistic polynomial time algorithm can find its factors p and q. The detailed characterizations for integer factorization can be referred to [17]. The security of the RSA cryptosystem is based on the difficulty of integer factoring problem. In the RSA cryptosystem, there are the public values n and d, as well as the secret values p, q and e such that $n = p \cdot q$ and $d \cdot e \equiv 1 \mod \phi(n)$, where $\phi(n) = (p-1)(q-1)$. Without knowing p and q, given n and d, an attacker cannot compute the secret key e. The detailed descriptions for the RSA cryptosystem can be referred to [18, 19].

[Lucas function]

The detailed definitions and properties of the Lucas function are referred to [20-22]. Here, we briefly present them. Let $n = p \cdot q$, where p and q are two large primes. The Lucas function is defined as

$$V_i(x) = \begin{cases} x \cdot V_{i-1}(x) - V_{i-2}(x) \mod n & i \geq 2 \\ 2 & i = 0 \\ x & i = 1 \end{cases},$$

where x is an integer and the sequence $\langle V_i(x) \rangle_{i=0}^{\infty}$ is called the Lucas sequence over x.

Lucas function has the following properties:

1. For all a, b, $x \in N$, we have $V_a(V_b(x)) = V_b(V_a(x)) = V_{ab}(x)$ and $V_{a+b}(x) = V_a(x)V_b(x) - V_{a-b}(x) \mod n$. Thus, we can obtain the following equations
   $$V_{2a}(x) = V_a^2(x) - 2 \mod n,$$
   and
   $$V_{2a+1}(x) = V_{a+1}(x)V_a(x) - x \mod n.$$
2. For all $e \geq 3$ and $d \in N$ satisfying two equations $\gcd(e, (p^2-1)(q^2-1)) = 1$ and $ed \equiv 1 \mod (p^2-1)(q^2-1)$, we have $V_e(V_d(x)) = 1$.
3. Given the values a, n and $V_a(x)$ above, to compute x is intractable.

[Hash Function]

A secure one–way hash function H [23] operates on an arbitrary length input m and outputs the fixed length $y = H(m)$ such that

- Function feasibility: Given m, it is easy to compute $y = H(m)$.
- Pre-image resistance: Given y, it is computationally infeasible to derive m such that $y = H(m)$.
- Second pre-image resistance: Given m, it is computationally infeasible to find m' ($m' \neq m$) such that $H(m') = H(m)$.
- Collision resistance: It is computationally infeasible to find m and m' such that $H(m) = H(m')$.

2.3. Notations

The following notations are used throughout the whole paper:

- $\hat{e}$: an admissible bilinear map $G_1 \times G_1 \to G_2$, where G_1 and G_2 are multiplicative cyclic groups of a large prime order q.
- $H()$: a secure one–way hash function $H: \{0, 1\}^* \to Z_q^*$.
- ID_i: the user U_i's identity.
- z: the maximum time of the system life cycle.
- (I_c, a_i, K_i): the secret key tuple of U_i, where I_c is the time–bound information for a continuous time interval from t_1 to t_2, where $0 < t_1 < t_2 \leq z$.
- (I_d, a_i, K_i): the secret key tuple of U_i, where I_d is the time–bound information for a discrete time interval set $T_i \subseteq T$, where $T = \{1,2,\ldots z\}$ is the set of all discrete intervals of the system life cycle.
- $E()$: a symmetric encryption function.

- $D()$: the corresponding symmetric decryption function of $E()$.

3. Scalable key management schemes

In this section, we present two scalable key management schemes (Scheme 1 and Scheme 2) for secure multicast communication without the rekeying process. Scheme 1 is suitable for users with continuous time intervals and Scheme 2 is suitable for users with discrete time intervals. In Scheme 1, we adopt the Lucas function and the ID-based broadcast scheme [24] to construct it. On the other hand, Scheme 2 uses the RSA-based key assignment scheme and the ID-based broadcast scheme. Note that we adopt the functionality of the ID-based broadcast encryption technique in [24] into the proposed schemes, but no traitor tracing functionality is used.

3.1. Scheme 1

Assume that there is a group controller (GC) and n members U_i with ID_i, where $1 \le i \le n$. Here, the GC is also the key generation center to generate all needed keys and public parameters. Now, the GC would like to broadcast a message to members at time t, where $0 < t \le z$ and z denotes the maximum time of the system life cycle. If t is located within members' time–bound intervals, these members can decrypt the broadcasting encrypted message. The proposed scheme consists of four phases: the system setup, the key assignment, the encryption and the decryption phases. We describe four phases in details as follows:

[System setup]

The GC first selects two large prime numbers p_1, q_1, and computes $n_1 = p_1 \cdot q_1$. The GC defines an admissible bilinear map $\hat{e} : G_1 \times G_1 \to G_2$, where G_1 and G_2 are multiplicative cyclic groups of the same order q. Then the GC randomly selects five integers a, f_1, $f_2 \in Z$, x_1, $x_2 \in Z_q^*$, where $1<a<n_1$. Note that three values a, x_1 and x_2 are kept as secret. Finally, the GC selects a secure one–way hash function $H:\{0, 1\}^* \to Z_q^*$. Then, the GC publishes the public parameters $\{n_1, f_1, f_2, \hat{e}, H()\}$.

[Key assignment]

When a user U_i wants to join the group and obtain a time–bound information I_c for the continuous time from t_1 to t_2, U_i submits her/his identity ID_i to the GC. Upon receiving the request, the GC randomly selects an integer $a_i \in Z_q^*$ and computes $K_i = g_1^{1/H(ID_i)+x_1+a_i x_2}$ and $I_c=V_{f_1^{z-t_2} f_2^{t_1}}(a)$, where g_1 is a secret value of G_1 and z denotes the maximum time of the system life cycle. Finally the GC sends (I_c, a_i, K_i) to U_i as her/his private key via a secure channel.

[Encryption]

Let M be a message to be broadcasted by the GC at time t, where $0 < t \le z$. Then, the GC performs the following steps:

Step 1: The GC randomly selects a secret value $g_2 \in G_1$, $s \in G_2$ and $r \in Z_q^*$ to compute $X_1 = (g_2^{x_1})^r$, $X_2 = (g_2^{x_2})^r$, $Y_1 = g_2^r$ and $Y_2 = s \cdot \hat{e}(g_1, g_2)^r$.

Step 2: The GC computes $w_t = V_{f_1^{z-t} f_2^t}(a) \bmod n_1$ and $k_c=H(s||w_t)$. The GC uses the encryption key k_c to encrypt the message M as $C=E(k_c, M)$.

Step 3: Finally, the GC broadcasts (t, X_1, X_2, Y_1, Y_2, C) to all users.

[Decryption]

Suppose that a user U_i owns the private key (I_c, a_i, K_i) for the continuous time from t_1 to t_2. If $t_1 \le t \le t_2$, the user U_i can perform the following steps to recover the message M.

Step 1: Upon receiving (t, X_1, X_2, Y_1, Y_2, C), the user U_i can use her/his secret key tuple (I_c, a_i, K_i) and public parameters to compute $s = Y_2 / \hat{e}(K_i, X_1 X_2^{a_i} Y_1^{H(ID_i)})$ and $w_t = V_{f_1^{t_2-t} f_2^{t-t_1}}(I_c)$.

Step 2: Then the user U_i can obtain the decryption key $k_c=H(s||w_t)$.

Step 3: The user U_i then uses the key k_c to decrypt the message $M= D(k_c, C)$.

Here, we present the correctness of s and w_t in Step 1 of the decryption phase as follows:

$$
\begin{aligned}
& Y_2 / \hat{e}(K_i, X_1 X_2^{a_i} Y_1^{H(ID_i)}) \\
&= s \cdot \hat{e}(g_1, g_2)^r / \hat{e}(g_1^{1/H(ID_i)+x_1+a_i x_2}, X_1 X_2^{a_i} (g_2^r)^{H(ID_i)}) \\
&= s \cdot \hat{e}(g_1, g_2)^r / \hat{e}(g_1^{1/H(ID_i)+x_1+a_i x_2}, (g_2^{x_1} g_2^{a_i x_2} g_2^{H(ID_i)})^r) \\
&= s \cdot \hat{e}(g_1, g_2)^r / \hat{e}(g_1^{1/H(ID_i)+x_1+a_i x_2}, g_2^{(H(ID_i)+x_1+a_i x_2) r}) \\
&= s \cdot \hat{e}(g_1, g_2)^r / \hat{e}(g_1, g_2)^r \\
&= s
\end{aligned}
$$

and

$$
\begin{aligned}
& V_{f_1^{t_2-t} f_2^{t-t_1}}(I_c) \\
&= V_{f_1^{t_2-t} f_2^{t-t_1}}(V_{f_1^{z-t_2} f_2^{t_1}}(a)) \\
&= V_{f_1^{t_2-t} f_2^{t-t_1} f_1^{z-t_2} f_2^{t_1}}(a) \\
&= V_{f_1^{z-t} f_2^{t}}(a) \\
&= w_t .
\end{aligned}
$$

3.2. Scheme 2

In Scheme 2, assume that a user U_i wants to join the group and she/he is allowed to obtain the time–bound information with a discrete time interval set $T_i \subseteq T$. Now, the GC would like to broadcast a message to group members at time t. If $t \in T_i$, the member U_i can decrypt the broadcasting encrypted message while the other members U_j cannot obtain the message if $t \notin T_j$. The proposed scheme also consists of four phases: the system setup, the key assignment, the encryption and the decryption phases. The detailed descriptions of four phases are presented as follows.

[System setup]

The GC first selects two prime numbers p_2 and q_2 to compute $n_2 = p_2 \cdot q_2$ and $\phi(n_2) = (p_2 - 1)(q_2 - 1)$. The GC defines an admissible bilinear map $\hat{e}$: $G_1 \times G_1 \to G_2$, where G_1 and G_2 are multiplicative cyclic groups of the same order q. Then, the GC chooses three secret values $b \in Z$, x_1, $x_2 \in Z_q^*$, where $1<b<n_2$. Then the GC selects a secure one–way hash function $H:\{0, 1\}^* \to Z_q^*$. Finally, the GC determines the RSA key pairs (d_y, e_y) such that $d_y \cdot e_y = 1 \bmod \phi(n_2)$ for each discrete time interval $y = 1,2,...,z$. Then the GC publishes the public parameters $\{ n_2, \hat{e}, H(), z, d_1, d_2,\ldots, d_z\}$.

[Key assignment]

Assume that a user U_i wants to join the group and obtain a time–bound information I_d of a discrete time interval set T_i. The user U_i submits her/his identity ID_i to the GC. Upon receiving the request, the GC randomly selects an integer $a_i \in Z_q^*$ and computes $K_i = g_1^{1/H(ID_i)+x_1+a_i x_2}$ and $I_d = b^{\prod_{y \in T_i} e_y} \bmod n_2$, where g_1 is a secret value of G_1. Finally the GC sends the key tuple (I_d, a_i, K_i) to U_i via a secure channel.

[Encryption]

Assume that the GC wants to broadcast a message M at time t, where $t \in T$ and $T = \{1,2,\ldots z\}$ is the set of all discrete intervals of the system life cycle. Then, the GC performs the following steps:

Step 1: The GC randomly selects a secret value $g_2 \in G_1$, $s \in G_2$ and $r \in Z_q^*$ to compute $X_1 = (g_2^{x_1})^r$, $X_2 = (g_2^{x_2})^r$, $Y_1 = g_2^r$, and $Y_2 = s \cdot \hat{e}(g_1, g_2)^r$.

Step 2: The GC computes $k_t = b^{e_t} \bmod n$ and $k_d = H(s||k_t)$. The GC uses the encryption key k_d to encrypt the message M as $C = E(k_d, M)$ using the symmetric encryption function $E()$.

Step 3: Finally, the group controller broadcasts $(t, X_1, X_2, Y_1, Y_2, C)$ to all users.

[Decryption]

Suppose that a user U_i owns the private key (I_d, a_i, K_i) for the discrete time interval set T_i. If $t \in T_i$, the user U_i can perform the following steps to recover the message M.

Step 1: Upon receiving $(t, X_1, X_2, Y_1, Y_2, C)$, the user U_i can use her/his secret key tuple (I_d, a_i, K_i) and the public parameters to compute $s = Y_2 / \hat{e}(K_i, X_1 X_2^{a_i} Y_1^{H(ID_i)})$ and $k_t = I_d^{\prod_{y \in T_i, y \neq t} d_y} = b^{e_t}$.

Step 2: Then the user U_i can obtain the decryption key $k_d = H(s||k_t)$.

Step 3: The user U_i uses the key k_d to decrypt the message $M = D(k_d, C)$.

Note that the correctness of the key s is the same with Scheme 1. Here, we only present the correctness of k_t in Step 1 as follows:

$$k_t = I_d^{\prod_{y \in T_i, y \neq t} d_y} = (b^{\prod_{y \in T_i} e_y})^{\prod_{y \in T_i, y \neq t} d_y} = b^{\prod_{y \in T_i} e_y \prod_{y \in T_i, y \neq t} d_y} = b^{e_t}.$$

4. Security analysis

In this section, we demonstrate the security analysis of the proposed schemes. The security of Scheme 1 is based on the computation of both s and w_t. Meanwhile, the security of Scheme 2 is based on the computation of both s and k_t. Therefore, we must prove the security of s, w_t and k_t, respectively. Since $k_c = H_1(s||w_t)$ and $k_d = H_1(s||k_t)$, we obtain the security of the encryption keys k_c and k_d if s, w_t and k_t are hard to compute. Finally, we prove that Schemes 1 and 2 are secure key management schemes for multicast communication.

[Security of the value *s*]

Lemma 1. In the proposed schemes, any illegal user cannot obtain the value s from the broadcast messages and public parameters.

▼**Proof.** If an illegal user tries to compute the value s, he must compute $\hat{e}(g_1, g_2)^r$. However, given the broadcast messages, only $Y_1 = g_2^r$ is known by the illegal user. Since g_1 is a secret value of G_1, an illegal user cannot get g_1 because the broadcast messages don't include any information about g_1. So it is impossible to compute $\hat{e}(g_1^r, g_2) = \hat{e}(g_1, g_2)^r$. Thus, he cannot obtain the secret value s. ▲

[Security of w_t]

Lemma 2. Under the Lucas function assumption, a legal user without the valid time–bound information or an outside adversary cannot compute $w_{t'}$ in the proposed Scheme 1.

▼**Proof:** Assume that the GC would like to broadcast a message to members at time t'. We will prove that unauthorized users or outside adversaries cannot compute the key $w_{t'}$. Note that legal users with the invalid time–bound information have more powerful information than outside adversaries. Here, we consider the attack of legal users with invalid time–bound information. Therefore, we must prove the following two situations.

(1) Assume a user Bob was given a time–bound information $I_c = V_{f_1^{z-t_2} f_2^{t_1}}(a)$, the user cannot compute $w_{t'}$ if $t' \in [0,z] \setminus [t_1,t_2]$. If Bob wants to compute $w_{t'}$, Bob must compute $f_1^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{(f_1^{-1})^{t'-t_2} f_2^{t'-t_1}}(I_c)$ for $t' < t_1$ or compute $f_2^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{f_1^{t_2-t'}(f_2^{-1})^{t_1-t'}}(I_c)$ for $t' > t_2$. According to the Lucas function assumption, without knowing p_1 and q_1, to compute f_1^{-1} or f_2^{-1} are computationally infeasible.

(2) Assume that there are two time–bound information I_{c_1} and I_{c_2}, where I_{c_1} denotes the time interval from t_1 to t_2, and I_{c_2} denotes the time interval from s_1 to s_2. The user cannot compute $w_{t'}$ if $t' \in [0,z] \setminus \{[t_1,t_2] \cup [s_1,s_2]\}$. It means that the user cannot compute $w_{t'}$ from $I_{c_1} = V_{f_1^{z-t_2} f_2^{t_1}}(a)$ and $I_{c_2} = V_{f_1^{z-s_2} f_2^{s_1}}(a)$. There are two kinds of attack scenarios. One is that Bob is given two time–bound information $I_{c_1} = V_{f_1^{z-t_2} f_2^{t_1}}(a)$ and $I_{c_2} = V_{f_1^{z-s_2} f_2^{s_1}}(a)$, and he wants to compute $w_{t'}$, where $t' \in [0,z] \setminus \{[t_1,t_2] \cup [s_1,s_2]\}$. The other one is that Bob is given a time–bound information $I_{c_1} = V_{f_1^{z-t_2} f_2^{t_1}}(a)$ and Alice is given the other time–bound information $I_{c_2} = V_{f_1^{z-s_2} f_2^{s_1}}(a)$, then they collude to compute $w_{t'}$, where $t' \in [0,z] \setminus \{[t_1,t_2] \cup [s_1,s_2]\}$. Without loss of generality, we assume that $0 < t_1 < t_2 < s_1 < s_2 < z$. Thus, three cases $t' \in [0, t_1)$, $t' \in (t_2, s_1)$ and $t' \in (s_2, z]$ are required to be addressed. Here we use the attacker to denote both Bob and Alice.

Case 1. For $t' \in [0, t_1)$, if the attacker wants to compute $w_{t'}$, he must compute $f_2^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{f_1^{s_2-t'}(f_2^{-1})^{s_1-t'}}(I_{c_2})$ or $w_{t'} = V_{f_1^{t_2-t'}(f_2^{-1})^{t_1-t'}}(I_{c_1})$. By the Lucas function assumption, without knowing p_1 and q_1, to compute f_2^{-1} is computationally infeasible.

Case 2. For $t' \in (t_2, s_2)$, if the attacker wants to compute $w_{t'}$, he must compute $f_1^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{(f_1^{-1})^{t'-t_2} f_2^{t'-t_1}}(I_{c_1})$ or $f_2^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{f_1^{s_2-t'}(f_2^{-1})^{s_1-t'}}(I_{c_2})$. By the similar reason, to compute f_1^{-1} or f_2^{-1} are computationally infeasible.

Case 3. For $t' \in (s_2, z]$, if the attacker wants to compute $w_{t'}$, he must compute $f_1^{-1} \bmod (p_1^2-1)(q_1^2-1)$ to obtain $w_{t'} = V_{(f_1^{-1})^{t'-t_2} f_2^{t'-t_1}}(I_{c_1})$ or $w_{t'} = V_{(f_1^{-1})^{t'-s_2} f_2^{t'-s_1}}(I_{c_2})$. By the Lucas function assumption, without knowing p_1 and q_1, to compute f_1^{-1} is computationally infeasible. ▲

[Security of k_t]

Lemma 3. Under the RSA cryptosystem assumption, a legal user without a valid time–bound information or an outside adversary cannot compute $k_{t'}$ in the proposed Scheme 2.

▼**Proof:** Assume that the GC would like to broadcast a message to members at the time t'. We will prove that unauthorized users or outside adversaries cannot compute the key $k_{t'}$. Here, we prove two cases.

(1) Given a time–bound key I_d, the attacker U_i can compute $k_t = I_d^{\prod_{y \in T_i, y \neq t} d_y} = b^{e_t}$ for $t \in T_i$, but the user cannot compute $k_{t'} = b^{e_{t'}}$ for $t' \notin T_i$. Of course, the attacker can compute the value b, but it is no help to obtain the data encryption key $k_{t'}$. Since $t' \notin T_i$, the time–bound key I_d didn't include $e_{t'}$. Therefore, the attacker must compute $e_{t'}$ from the public value $d_{t'}$ such that $d_{t'} \cdot e_{t'} \equiv 1 \bmod \phi(n_2)$. By the RSA cryptosystem assumption, it is hard to compute $e_{t'}$ because the

attacker cannot compute p_2 and q_2 from $n_2 = p_2 \cdot q_2$.

(2) Given several time–bound keys $I_{d\_t_1}, I_{d\_t_2}, ..., I_{d\_t_\alpha}$, where $t_1, t_2, ..., t_\alpha \in T_i$. The user U_i can compute $k_{t_1}, k_{t_2}, ..., k_{t_\alpha}$, but the user cannot compute $k_{t'}$ for $t' \notin T_i$. By the similar reason, the security is under the RSA cryptosystem assumption. ▲

Based on the lemmas above, the following theorems demonstrate that Schemes 1 and 2 are secure key management schemes for multicast communication.

Theorem 4. *In the proposed Scheme 1, any illegal user cannot compute the encryption key k_c, where $k_c=H_1(s||w_t)$.*

▼**Proof.** By Lemmas 1 and 2, we have proven that s and w_t are secure against attackers, so that any illegal users cannot obtain s and w_t. If the illegal user can obtain a value $v \neq s||w_t$ such that $k_c = H(v)$, there is a contradiction with the "collision resistance" property of the one-way hash function. Therefore, for any illegal users, it is impossible to obtain the encryption key $K_c = H(s||w_t)$. ▲

Theorem 5. *In the proposed Scheme 2, any illegal user cannot compute the encryption key k_d, where $k_d=H_1(s||k_t)$.*

▼**Proof.** By Lemmas 1 and 3, we have proved that s and k_t are secure against adversaries, thus any illegal users cannot obtain s and k_t. If the illegal user can obtain a value $v \neq s||k_t$ such that $k_d = H(v)$, there is also a contradiction with the security property "collision resistance" of the one-way hash function. Therefore, for any illegal users in Scheme 2, it is impossible to obtain the encryption key $K_d = H(s||k_t)$. ▲

5. Discussions and comparisons

In this section, we compare our proposed schemes with the previously proposed schemes. Here, we briefly review and analyze several previously proposed schemes that include the LKH scheme [5] and Lin *et al.*'s schemes [12]. Then, performance comparisons are given to demonstrate that our proposed schemes have better performance as compared with the recently proposed key management schemes in terms of transmission size and computational cost.

5.1. Review of several key management schemes

[LKH scheme]

In 1999, Wallner *et al.* [5] proposed the logical key hierarchy (LKH) tree management scheme. In their scheme, the multicast group includes the GC and n group members u_0, u_1,…,u_{n-1}, where the GC is responsible to generate all keys of the LKH tree. Each user is assigned to one leaf of the LKH tree. For example, eight members form the LKH tree as depicted in Fig 1. The GC gives a key k_v to every node v in the LKH tree, and sends each user (via a secure channel) the keys along the path from the member to the root. In Fig 1, the user u_0 will obtain the keys k_{000}, k_{00}, k_0 and k. The GC can use k to encrypt messages for multicast communications.

Suppose u_7 is the new joining user. The GC gives u_7 a key k_{111} via a secure channel. For the nodes v_0, v_1 and v_2 along the path from the root to u_7, the GC generates the new keys k', k'_1 and k'_{11}, respectively. Then k', k'_1, and k'_{11} are encrypted with the keys k, k_1, and k_{11}, respectively. All encryptions are broadcasted to all old users. Then, every old user in the group can decrypt them by their own keys. Meanwhile, $E_{k_{111}}(k')$, $E_{k_{111}}(k'_1)$, and $E_{k_{111}}(k'_{11})$ are sent to the user u_7, where $E()$ is a symmetric encryption function. Therefore, when a user u joins the group, both the computation cost and the transmission size of the rekeying process are O(h) for the GC, where h is the height of the LKH tree. At the same reason, in order to delete the user u_0 from the LKH tree in Fig 1, the following encryptions $E_{k_{001}}(k'_{00})$, $E_{k'_{00}}(k'_0)$, $E_{k_{01}}(k'_0)$, $E_{k'_0}(k')$, and $E_{k_1}(k')$ are transmitted. Therefore, when a user u leaves the group, both the computation cost and the transmission size of the rekeying process are O(h).

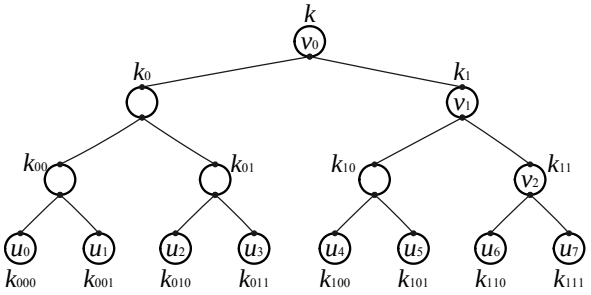

Figure 1. The logical key hierarchy tree

[Lin *et al.*'s schemes]

In 2010, Lin *et al.* [12] proposed two multicast key management schemes without the rekeying process. Here, we briefly review their first scheme. Without loss of generality, assume that there are n group members $U_1, U_2, \ldots, U_n$ and a group controller (GC). The GC assigns secret keys to group members as follows:

(1) The GC chooses two secret primes p_i and q_i, and then computes the public product $N_i = p_i \times q_i$ for each group member U_i, where $i = 1, 2, \ldots, n$.

(2) The GC computes the least common multiple

$L_0 = LCM(\phi(N_1), \phi(N_2), ..., \phi(N_n))$,

where $\phi(N_i) = (p_i - 1) \times (q_i - 1)$ and $1 \le I \le n$. The GC then chooses a prime $e_0 < \min\{\phi(N_1), \phi(N_2), ..., \phi(N_n)\}$ such that gcd (e_0,L_0)=1. Also, the GC computes $d_0 > \min\{\phi(N_1), \phi(N_2), ..., \phi(N_n)\}$ such that $e_0 \times d_0 = 1 \bmod L_0$.

(3) The GC, respectively, sends the secret key $d_i = d_0 \bmod \phi(N_i)$ to each group member U_i via a secure channel, where i = 1, 2,…, n.

Suppose that the GC wants to send a message M to some group members. The GC first defines a set DU of dedicated users whom the GC wants to send to. Then the GC encrypts the message M by using $C = M^{e_0} \bmod \prod_{U_i \in DU} N_i$, where $M \in [0, \min\{N_1, N_2, \ldots, N_n\} - 1]$. Upon receiving the encrypted message C, each user $U_i \in DU$ can decrypt the message $M = (C \bmod N_i)^{d_i} \bmod N_i$ by using his/her secret key d_i and the corresponding public parameter N_i. Thus, the transmission size is $|C| = |M^{e_0} \bmod \prod_{U_i \in DU} N_i|$. Obviously, the transmission size is dependent on the number of dedicated users in the set DU.

5.2. Performance analysis and Comparisons

For convenience, the following notations are used to analyze the computational cost and the transmission size.

- n: the number of members in the group.
- d: the degree of the tree in the tree–based schemes.
- h: the height of the balanced LKH tree.
- T_H: the time of executing a one–way hash function $H(\ )$.
- T_E: the time of executing a symmetric encryption or decryption.
- T_L: the time of executing the Lucas function operation.
- T_{Ge}: the time of executing bilinear pairing operation $\hat{e}$.
- T_{inv}: the time of executing modular inverse operation in Z_q^*.
- T_{pow}: the time of executing modular exponential operation in Z_q^*.
- T_{ex}: the time of executing modular exponential operation in G_1.
- T_m: the time of executing modular multiplication operation in Z_q^*.
- $|m|$: the bit size of a message m.

As reviewed in Subsection 5.1, every user in the LKH tree is assigned $\log_d n$ keys from his/her location to the root. The LKH-like schemes [5-11] may apply the d–ary key trees to arrange members so that the height is $h = \log_d n$. In the following, we analyze the performance of two cases (worst case and best case) in the LKH-like schemes. For the worst case, the GC must encrypt the same message using at most $n/2$ different keys. Thus, the transmission size in the LKH approach is O(n). For example, in Fig. 1, the GC wants to send the message M to u_0, u_2, u_4, and u_6. Then, M is encrypted using k_{000}, k_{010}, k_{100}, and k_{110}, respectively. Considering the transmission size in the best case of multicast communication in Fig. 1, the GC encrypts the message using the root key k in the LKH tree. Obviously, it requires O(1) transmission size. According to the number of encryptions, the computational costs of the GC are O(1) and O(n) for the best and worst cases, respectively. Since each member performs one decryption to obtain the message for two cases, the computational cost is O(1).

As reviewed in Subsection 5.1, Lin *et al*. [12] use $C = M^{e_0} \bmod \prod_{U_i \in DU} N_i$ to send the message to the set DU of dedicated users. According to the mentioned best and worst cases above, the set DU includes n members and $n/2$ members, respectively. Since the transmission size is dependent on $|\prod_{U_i \in DU} N_i|$, two cases require O(n) transmission size. Considering the computational cost, the GC and each member require O(n) and O(1) for two cases, respectively.

In the following, let us discuss the transmission size of our proposed schemes. In our Schemes 1 and 2, the GC broadcasts (t, X_1, X_2, Y_1, Y_2, C) to all members. It is obvious that the transmission size is constant since these values X_1, X_2, $Y_1 \in G_1$, $Y_2 \in G_2$, and C=E(k_c, M). Thus, the transmission size required for both Schemes 1 and 2 is O(1). Considering the required computational costs for the GC and each member, the GC and each member, respectively, perform the encryption and decryption procedures described in Section 3. In Scheme 1, the GC and each member require $3T_{ex}+T_{pow}+T_{Ge}+T_L+T_H+T_E$ and $2T_{ex}+T_{Ge}+$ $+T_{inv}+T_H+T_E$ time, respectively. In Scheme 2, the GC and each member, respectively, require $3T_{ex}+2T_{pow}+$ $+T_{Ge}+T_H+T_E$ and $2T_{ex}+T_{Ge}+T_{inv}+T_{pow}+T_H+T_E+lT_m$ time, where l denotes the size of the discrete time interval set T_i. Nevertheless, four computational costs are independent on the number of group members. The computational costs of both the GC and each member are O(1).

Table 1 summarizes the comparisons between the LKH-like schemes [5-11], Lin *et al*.'s scheme [12] and our proposed schemes (Scheme 1 and Scheme 2) in terms of the rekeying process, the computational cost and the transmission size in the worst case and best case. According to Table 1, it is obvious that our schemes have the better performance than the

previously proposed schemes in terms of transmission size and computational cost.

Table 1. Comparisons of our schemes and the previously proposed schemes

	Rekeying process	Transmission size		Computational cost in best/worst cases	
		Best case	Worst case	GC	Member
LKH-like schemes [5-11]	O(h)	O(1)	O(n)	O(1)/O(n)	O(1)
Lin's scheme [12]	Not required	O(n)	O(n)	O(n)	O(1)
Our Scheme 1	Not required	O(1)	O(1)	O(1)	O(1)
Our Scheme 2	Not required	O(1)	O(1)	O(1)	O(1)

6. Conclusions

In this paper, we have proposed two scalable key management schemes without the rekeying process. The point is that the transmission size is independent on the number of group members. Meanwhile, the required computational costs of each member and the group controller are constant. We have proved that the proposed schemes are secure key management schemes without the rekeying process. As compared with the previously proposed schemes, performance analysis has been made to demonstrate that our schemes have the better performance and are suitable for a dynamic multicast group.

Acknowledgements

The authors would like to thank the anonymous referees for their valuable comments and constructive suggestions. This research was partially supported by National Science Council, Taiwan, R.O.C., under contract no. NSC97-2221-E-018-010-MY3.

References

[1] **T. Liao.** Webcanal: a multicast web application. In: *Computer Networks and ISDN Systems*, 1998, 29 (8-13), 1091–1102.

[2] **S. Rafaeli, D. Hutchison.** A survey of key management for secure group communication. In: *ACM Computing Surveys*, 2003, 35 (3), 309–329. http://dx.doi.org/10.1145/937503.937506.

[3] **S. Mittra.** Iolus: A Framework for Scalable Secure Multicasting. In: *Proc. ACM SIGCOMM'97*, 1997, 277–288.

[4] **R. Molva, A. Pannetrat.** Scalable multicast security with dynamic recipient groups. In: *ACM Transactions on Information and System Security*, 2000, 3(3), 136–160. http://dx.doi.org/10.1145/357830.357834.

[5] **D. Wallner, E. Harder, R. Agee.** Key Management for Multicast: Issues and Architectures. In: *Technical Report RFC 2727, Internet Engineering Task Force*, 1999.

[6] **C. K. Wong, M. Gouda, S. S. Lam.** Secure Group Communications Using Key Graphs. In: *IEEE/ACM Transactions on Networking*, 2000, 8(1), 16–30. http://dx.doi.org/10.1109/90.836475.

[7] **X. S. Li, Y. R. Yang, M. G. Gouda, S. S. Lam.** Batch Rekeying for Secure Group Communications. In: *Proc. ACM SIGCOMM'01*, 2001, 525–534.

[8] **A. T. Sherman, D. A. McGrew.** Key establishment in large dynamic groups using one–way function trees. In: *IEEE Trans. Softw. Eng.*, 2003, 29(5), 444–458. http://dx.doi.org/10.1109/TSE.2003.1199073.

[9] **Y. M. Tseng.** A Scalable Key Management Scheme with Minimizing Key Storage for Secure Group Communications. In: *International Journal of Network Management*, 2003, 13(6), 419–425. http://dx.doi.org/10.1002/nem.503.

[10] **J. Goshi, R. E. Ladner.** Algorithms for Dynamic Multicast Key Distribution Trees. In: *Proc. Twenty–second Annual Symp. Principles of Distributed Computing*, 2003, 243–251. http://dx.doi.org/10.1145/872035.872071.

[11] **H. Lu.** A novel high–order tree for secure multicast key management. In: *IEEE Trans. Computers*, 2005, 54(2), 214–224. http://dx.doi.org/10.1109/TC.2005.15.

[12] **I. C. Lin, S. S. Tang, C. M. Wang.** Multicast Key Management without Rekeying Processes. In: *The Computer Journal*, 2010, 53 (7), 939-950. http://dx.doi.org/10.1093/comjnl/bxp060.

[13] **T. Y. Wu, Y. M. Tseng.** An efficient user authentication and key exchange protocol for mobile client-server environment. In: *Computer Networks*, 2010, 54 (9), 1520-1530. http://dx.doi.org/10.1016/j.comnet.2009.12.008.

[14] **Y. M. Tseng, T. Y. Wu, J. D. Wu.** An efficient and provably secure ID-based signature scheme with batch verifications. In: *International Journal of Innovative Computing, Information and Control*, 2009, 5 (11), 3911-3922.

[15] **L. Chen, Z. Cheng, N. P. Smart.** Identity–based key agreement protocols from pairings. In: *International Journal of Information Security*, 2007, 6 (4), 213–241. http://dx.doi.org/10.1007/s10207-006-0011-9.

[16] **D. Boneh, M. Franklin.** Identity–based encryption from the Weil pairing. In: *Proc. CRYPTO 2001*, LNCS, 2139, 2001, 213–229.

[17] **A. K. Lenstra.** Integer Factoring. In: *Designs, Codes and Cryptography*, 2000, 19 (2-3), 101–128. http://dx.doi.org/10.1023/A:1008397921377.

[18] **J. H. Yeh.** A secure time-bound hierarchical key assignment scheme based on RSA public key cryptosystem. In: *Information Processing Letters*, 2008, 105 (4), 117–120. http://dx.doi.org/10.1016/j.ipl.2007.08.017.

[19] **A. J. Menezes, P. C. van Oorschot, S. A. Vanstone.** Handbook of Applied Cryptography, CRC press, 1996. http://dx.doi.org/10.1201/9781439821916.

[20] **W. G. Tzeng.** A Time–Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy. In: *IEEE Trans. Knowledge and Data Eng.*, 2002, 14 (1), 182–188. http://dx.doi.org/10.1109/69.979981.

[21] **S. M. Yen, C. S. Laih.** Fast Algorithms for LUC Digital Signature Computation. *IEE Proceedings on*

Computers and Digital Techniques, 1995, 142 (2), 165–169. http://dx.doi.org/10.1049/ip-cdt:19951788.

[22] S. Chiou, C. S. Laih. An efficient algorithm for computing the LUC chain. *IEE Proceedings on Computers and Digital Techniques*, 2000, 147 (4), 263–265. http://dx.doi.org/10.1049/ip-cdt:20000534.

[23] NIST/NSA FIPS 180–2, SHS. Gaithersburg, MD, USA, 2005.

[24] C. Yang, W. Ma, X. Wang. New traitor tracing scheme against anonymous attack. *In: Proc. 1st International Conference on Innovative Computing, Information and Control*, *IEEE*, 2006, 389–392.

Received September 2010.