An Efficient and Flexible Decentralized Multicast Key Distribution Scheme

Wen-Shenq Juang and Jyan-Cwan Wu
Department of Information Management
Shih Hsin University
Taipei, Taiwan, R.O.C.

Summary

Multicast communication is becoming the basis for a growing number of Internet-based applications. The secure distribution of services or messages by the server to all multicasting group members requires an efficient and scalable way for distributing a group key to eligible members. Most of the existing research are focusing on group key and group members management, and can be divided into four types: centralized, decentralized, distributed, and hybrid schemes. In this paper, we propose an efficient and flexible decentralized multicast key distribution scheme with less computational cost and more functionality. The merits include: (1) the scheme needs no shared keys table between the registration center and all members; (2) a group key is distributed by servers to eligible members; (3) the computation cost is very low; (4) members and servers can authenticate each other; (5) our scheme is nonce-based and dose not have a serious time-synchronization problem; (6) also, the shared secret key distribution between dynamic participants is addressed in our proposed scheme.

Key words: *Multicast, Multicast group key, Key distribution, Network security, Security service.*

1. Introduction

Multicasting is becoming the basis for a growing number of Internet-based applications [19], e.g. teleconferencing, pay-per-view, on-line TV and on-line games. The distribution of services or messages by the server to all multicasting group members requires a security framework with an efficient and scalable way of distributing a group key to the eligible members.

In a secure multicast communication system [3, 8], in order to preserve the secrecy of eligible members, the group key must be changed and redistributed to all the current members when some member leaves or join this group. Otherwise, it is possible for the new members to decrypt the past tapped encrypted messages or for the former members to decrypt the new encrypted messages. To prevent these problems, the following two security criteria are important for the group key distribution in secure multicast communication [16].

S1: Forward secrecy: If a person has left a group, the departed member cannot decrypt encrypted messages transmitted after the leaving.

S2: Backward secrecy: If a person joins a group, he cannot decrypt encrypted messages transmitted before the joining.

The process for achieving forward and backward secrecy requires redistributing the group key. This process is called group rekeying.

In [16], the multicasting group key distribution is divided into three main classes: centralized group key management protocols, decentralized architectures and distributed key management protocols. Most centralized models use Logical Key Hierarchical (LKH) methods [11, 15, 20]. However, these approaches introduce key storage risk and are inefficient when the group is large. Decentralized architectures use subgroup controllers to distribute the group key [2, 13]. They do not discuss how Key Encryption Keys (KEKs) between a subgroup controller and its members are distributed. Distributed key management protocols use the variant Diffie-Hellman key agreement [18]. All members submit some information to generate the corresponding group key. It is inefficient because it has higher computation and communication costs. Since the group is generated by all members, it is hard to control forward and backward secrecy.

In this paper, we propose an efficient and flexible decentralized multicast key distribution scheme with less computational cost and more functionality. Our proposed scheme satisfies the forward and backward secrecy requirement for members who join or leave a group. In addition, we propose a novel shared key distribution scheme between all members that is not addressed by other decentralized schemes.

The structure of this paper is organized as follows. In section 2, we give a brief review of related work. In section 3, we describe the decentralized multicast key

distribution scheme. Security analysis and efficiency considerations are discussed in sections 4 and 5. In section 6, we offer concluding remarks.

2. Related work

In [1, 16], secure approaches of multicasting group key distribution are divided into four main classes: centralized group key management protocols, decentralized architectures, distributed key management protocols, and hybrid models.

2.1 Centralized group key management

A typical centralized group key management protocol contains one important entity, the Key Distribution Center (KDC), which generates the group key and distributes it to all members. The easiest way to distributing group key is using the Group Key Management Protocol (GKMP) [6, 7]. When a rekeying is required, the old conference key is used to encrypt the new conference key. However, the approach is not a solution for the forward secrecy. Another important property of a centralized mode is the use of a Logical Key Hierarchical (LKH) [4, 5, 11, 15, 20]. LKH is a kind of a binary tree to solve the key distribution in a group. However, these approaches introduce key storage risk and are inefficient when the group is large.

2.2 Decentralized architectures

In decentralized models [2, 13], a KDC is used and the large group is split into small subgroups. Each subgroup has a subgroup controller to reduce the work of the KDC. In Iolus [13], there is a group security agent (GSA) to manage each subgroup. Those GSAs are also managed by a group security controller (GSC). The GSC uses independent keys for each subgroup. No general group key is available for all group members. Although membership changes in subgroups are local, the major drawback of Iolus is that when a subgroup member wants to transmit the messages to other subgroups, GSA must perform the translation since data are encrypted by each GSA's subgroup secret key. This approach can reduce the workload on the GSC, but the GSA can become a bottleneck. Furthermore, the GSC does not authenticate each subgroup member. The shared keys between GSC and its dynamic members are not addressed in those schemes [2, 13].

2.3 Distributed key management protocols

The distributed approaches [18] have no group controller. All members must contribute their own secrets to generate the group key. A typical way is using the variant Diffie-Hellman key distribution schemes to generate the group key. However, the computation and communication cost is very high due to many exponentiation operations.

2.4 Hybrid models

In [1], a hybrid secure multicast communication scheme was proposed by combining the LKH and the Iolus framework. The scheme has a group controller (GC) and the large group is also split into small subgroups. The GC uses Iolus to distribute the group key to the subgroup controller (SC). The SC uses LKH to forward the group key to group members. Although this approach can lower the GC load, the major drawbacks are the centralized problem in SC, and the GC could not authenticate each subgroup member. In addition, the shared key distribution between GC and his dynamic members is not addressed.

3. Our proposed scheme

In this section, we propose an efficient and flexible decentralized multicast key distribution scheme. There are three kinds of participants in our scheme: the registration center, subgroup controllers, and subgroup members. In our scheme, all members submit their identities to the registration center for registration. The registration center classifies members into different subgroups. In each subgroup, the registration center assigns a static member as the subgroup controller to reduce the cost of key distribution traffic. When a subgroup controller wants to send the group key to members, the KEK keys between the subgroup controller and its members can be inquired from the registration center. We assume the registration center is a trusted server that performs registration, shared key queries and multicast grouping. In the proposed approach, the registration center divides all members into subgroups based on their behaviors and distances. The subgroups are connected to the registration center and arranged in the form of a hierarchy.

Let RC be the registration center. Let $U_{i,j}$ denote the dynamic member j in the subgroup i and $ID_{i,j}$ denote the unique identification of $U_{i,j}$. Without loss of generality, let $U_{i,0}$ be the static member in the subgroup i. Let G_i be the subgroup i. Let $h()$ be a secure one-way hashing function [12]. Let $E_K(m)$ be the ciphertext of m encrypted using the secret key K of a secure symmetric cryptosystem [14]. Let $D_K(c)$ denote the plaintext of c decrypted using the secret key

K of the corresponding symmetric cryptosystem [14]. Then let $\|$ denote the conventional string concatenation operator. Let $X \rightarrow Y : \{Z\}$ denote a sender X sends a message Z to a receiver Y. Finally, let x be the master secret key of RC, $V_{i,j} = h(x, ID_{i,j})$ is the secret key computed by RC and shared by $U_{i,j}$ and RC after $U_{i,j}$ registering at RC. Therefore, $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$ is the secret key computed by RC and $U_{i,j}$ shared by $U_{i,0}$ and $U_{i,j}$. The proposed scheme consists of five phases: the registration phase, the grouping phase, the key distribution phase, the shared key inquiry phase, and the dynamic membership management phase.

3.1 The registration phase

RC performs grouping and key management required for the multicast key distribution. Each member submits his identity information to RC for registration. If RC accepts this request, then it will perform the following steps.

Step 1 : Compute $U_{i,j}$'s secret key $V_{i,j} = h(x, ID_{i,j})$

Step 2 : Send $V_{i,j}$ to $U_{i,j}$ via a secure channel or store $V_{i,j}$ in a smart card and give it to $U_{i,j}$.

3.2 The grouping phase

To reduce the cost of key distribution traffic, an efficient grouping mechanism follows. The registration center RC classifies members into different subgroups based on their behavior and geography. In each subgroup, RC assigns a static member as the subgroup head (controller). The general classification mechanism is based on the distance from the static members or the value of a threshold function. The static members perform the group key distribution for their subgroups. These approaches will reduce the burden on the RC.

3.3 The key distribution phase

After grouping, RC generates a group key K and sends it to all subgroup's static members $U_{i,0}$. Then all of the subgroup's static members distribute the group key K to all their members. The following protocol is the group key distribution for each subgroup i.

Step 1: $RC \rightarrow U_{i,0} : \{N_{i,1}, E_{V_{i,0}}(K, h(K \| N_{i,1}))\}$

Step 2: $U_{i,0} \rightarrow RC : \{E_{V_{i,0}}(N_{i,1} + 1, N_{i,2})\}$

Step 3: $RC \rightarrow U_{i,0} : \{E_{V_{i,0}}(N_{i,2} + 1)\}$

Step 4: $U_{i,0} \rightarrow U_{i,j \in G_i} : \{ID_{i,0}, N_{i,3}, E_{V_{i,0,j}}(K, h(K \| N_{i,3} \| ID_{i,0}))\}$

Step 5: $U_{i,j \in G_i} \rightarrow U_{i,0} : \{E_{V_{i,0,j}}(N_{i,3} + 1, N_{i,4})\}$

Step 6: $U_{i,0} \rightarrow U_{i,j \in G_i} : \{E_{V_{i,0,j}}(N_{i,4} + 1)\}$

In step 1, RC generates the group key K, a nonce $N_{i,1}$, and sends the message $\{N_{i,1}, E_{V_{i,0}}(K, h(K \| N_{i,1}))\}$ to $U_{i,0}$. The nonce $N_{i,1}$ is a fresh random number for freshness checking. The authentication tag $h(K \| N_{i,1})$ is used for verifying the identification of RC.

After receiving the message in step 1, $U_{i,0}$ decrypts the message using its secret key and derives the group K by computing $D_{V_{i,0}}(E_{V_{i,0}}(K, h(K \| N_{i,1})))$. Then it checks if the authentication tag $h(K \| N_{i,1})$ is valid. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0}}(N_{i,1} + 1, N_{i,2})$ back to RC. The nonce $N_{i,2}$ is for freshness checking.

Upon receiving the encrypted message in step 2, RC decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,1} + 1, N_{i,2}))$ and checks if the nonce $N_{i,1} + 1$ is in it for freshness checking. If yes, RC sends the encrypted message $E_{V_{i,0}}(N_{i,2} + 1)$ back to $U_{i,0}$ in step 3. Upon receiving the message $E_{V_{i,0}}(N_{i,2} + 1)$, $U_{i,0}$ decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,2} + 1))$ and checks if the nonce $N_{i,2} + 1$ is in it for freshness checking.

After step 3, RC has sent the group key K to all subgroup heads. When $U_{i,0}$ wants to send the group key K to his subgroup member $U_{i,j}$, the shared key $V_{i,0,j}$ can be inquired from RC using the shared key inquiry phase.

In step 4, $U_{i,0}$ generates a nonce $N_{i,4}$, and sends $ID_{i,0}$, $N_{i,3}$, the encrypted message $E_{V_{i,0,j}}(K, h(K \| N_{i,3} \| ID_{i,0}))$ to $U_{i,j}$. The nonce $N_{i,3}$ is for freshness checking.

After receiving $ID_{i,0}$, $N_{i,3}$ and the encrypted message in step 4, $U_{i,j}$ decrypts the message by first

computing the shared key $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$. Then, $U_{i,j}$ can derive the group key K by computing $D_{V_{i0,j}}(E_{V_{i0,j}}(K, h(K \| N_{i,3} \| ID_{i,0})))$ and checking if the authentication tag $h(K \| N_{i,3} \| ID_{i,0})$ is valid. If yes, $U_{i,j}$ sends the encrypted message $E_{V_{i0,j}}(N_{i,3}+1, N_{i,4})$ back to $U_{i,0}$ in step 5. The nonce $N_{i,4}$ is for freshness checking.

Upon receiving the encrypted message in step 5, $U_{i,0}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4}))$ and checking if the nonce $N_{i,3}+1$ is in it for freshness checking. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,4}+1)$ back to $U_{i,j}$. Upon receiving $E_{V_{i,0,j}}(N_{i,4}+1)$, $U_{i,j}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,4}+1))$ and checks if the nonce $N_{i,4}+1$ is in it for freshness checking. After finishing all steps, all group members can use the group key K for secure communication.

3.4 The shared key inquiry phase

When $U_{i,0}$ wants to send the group key K to his subgroup members $U_{i,j}$, the shared key $V_{i,0,j}$ can be inquired from RC. This approach can avoid the risk of secret key storage. The following protocols are the shared key inquiry phase.

Step 1: $U_{i,0} \rightarrow RC : \{N_{i,5}, ID_{i,0}, ID_{i,j}, E_{V_{i,0}}(h(ID_{i,0} \| ID_{i,j} \| N_{i,5}))\}$

Step 2: $RC \rightarrow U_{i,0} : \{E_{V_{i,0}}(V_{i,0,j}, h(V_{i,0,j} \| (N_{i,5}+1)))\}$

In step 1, $U_{i,0}$ sends the inquiry message to RC. The message includes the authentication tag $h(ID_{i,0} \| ID_{i,j} \| N_{i,5})$ and is encrypted by the secret key $V_{i,0}$ shared by $U_{i,0}$ and RC, which is for verifying the identification of $U_{i,0}$. After receiving the inquiry message, RC first decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(h(ID_{i,0} \| ID_{i,j} \| N_{i,5})))$, then checks if the message contains the authentication tag $h(ID_{i,0} \| ID_{i,j} \| N_{i,5})$ and if the nonce $N_{i,5}$ is fresh. RC can keep a simple table to record recently used nonces. RC rejects the inquiry if it is not valid. If it is valid and the nonce is fresh, RC computes the shared key $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$ shared by $U_{i,0}$ and his subgroup members $U_{i,j}$. Then RC sends the encrypted message $E_{V_{i,0}}(V_{i,0,j}, h(V_{i,0,j} \| (N_{i,5}+1)))$ back to $U_{i,j}$ in step 2.

Upon receiving the encrypted message in step 2, $U_{i,0}$ uses his secret key to decrypt it and derives the subgroup member's shared key $V_{i,0,j}$.

3.5 The dynamic membership management phase

To ensure the forward secrecy and backward secrecy, a secure multicast protocol must update the group key when the group members change. In our scheme, RC performs the dynamic membership management for scalability. The join and leave protocols are initialized by the dynamic members. We also consider the situation of the leaving of a subgroup's static member.

3.5.1 Member joining

We assume a new member $U_{i,k}$ wants to join the service, it must submit his identity information to RC for registering first. If RC accepts this request, then it will perform the following steps.

Step 1 : Compute $U_{i,k}$'s secret key $V_{i,k} = h(x, ID_{i,k})$

Step 2 : Send $V_{i,k}$ to $U_{i,k}$ via a secure channel or store $V_{i,k}$ in a smart card and give it to $U_{i,k}$.

After registering, RC updates the group key K' and sends it to all members to maintain forward secrecy. The following protocol is the group key distribution for each subgroup i, where $N, N_{i,1}, N_{i,2}, N_{i,3}, N_{i,4}$ are nonces, and JN is the joining-request message.

Step 1: $U_{i,k} \rightarrow RC : \{ID_{i,k}, JN, N, E_{V_{i,k}}(h(ID_{i,k} \| N \| JN))\}$

Step 2: $RC \rightarrow U_{i,0} : \{N_{i,1}, JN, ID_{i,k}, E_{V_{i,0}}(K', h(K' \| N_{i,1} \| ID_{i,k} \| JN))\}$

Step 3: $U_{i,0} \rightarrow RC : \{E_{V_{i,0}}(N_{i,1}+1, N_{i,2})\}$

Step 4: $RC \rightarrow U_{i,0} : \{E_{V_{i,0}}(N_{i,2}+1)\}$

Step 5: $U_{i,0} \rightarrow U_{i,j \in G_i} : \{ID_{i,0}, N_{i,3}, JN, E_{V_{i,0,j}}(K', h(K' \| ID_{i,0} \| N_{i,3} \| JN))\}$

Step 6: $U_{i,j \in G_i} \rightarrow U_{i,0} : \{E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})\}$

Step 7: $U_{i,0} \rightarrow U_{i,j \in G_i} : \{E_{V_{i,0,j}}(N_{i,4}+1)\}$

In step 1, $U_{i,k}$ sends his identity $ID_{i,k}$, a nonce N, the joining message JN and the message $E_{V_{i,k}}(h(ID_{i,k} \| N \| JN)$ to RC to join the service. If RC accepts the request, then it executes the grouping protocol and classifies the member $U_{i,k}$ into $U_{i,0}$'s subgroup. After grouping, RC updates the group key K' and sends the message $\{N_{i,1}, JN, ID_{i,k}, E_{V_{i,0}}(K', h(K' \| N_{i,1} \| ID_{i,k} \| JN))\}$ including the new member's joining message to $U_{i,0}$ in step 2. The nonce $N_{i,1}$ is a fresh random number for freshness checking. The authentication tag $h(K' \| N_{i,1} \| ID_{i,k} \| JN)$ is used for verifying the identification of RC.

Upon receiving the encrypted message in step 2, $U_{i,0}$ decrypts the message using his secret key and derives the new group key K' by computing $D_{V_{i,0}}(E_{V_{i,0}}(K', h(K' \| N_{i,1} \| ID_{i,k} \| JN)))$. Then checks if the authentication tag $h(K' \| N_{i,1} \| ID_{i,k} \| JN)$ is valid. If the authentication tag $h(K' \| N_{i,1} \| ID_{i,k} \| JN)$ is valid, $U_{i,0}$ sends the encrypted message $E_{V_{i,0}}(N_{i,1}+1, N_{i,2})$ back to RC in step 3. The nonce $N_{i,2}$ is for freshness checking.

After receiving the encrypted message in step 3, RC decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,1}+1, N_{i,2}))$ and checks if the nonce $N_{i,1}+1$ is in it for freshness checking. If yes, RC sends the encrypted message $E_{V_{i,0}}(N_{i,2}+1)$ back to $U_{i,0}$ in step 4. Upon receiving the message $E_{V_{i,0}}(N_{i,2}+1)$, $U_{i,0}$ decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,2}+1))$ and checks if the nonce $N_{i,2}+1$ is in it for freshness checking.

After step 4, RC has sent the new group key K' to all subgroup heads. In step 5, $U_{i,0}$ generates a nonce $N_{i,3}$, and sends his $ID_{i,0}$, $N_{i,3}$, the encrypted message $E_{V_{i,0,j}}(K', h(K' \| N_{i,3} \| ID_{i,0} \| JN))$ to $U_{i,j \in G_i}$. The nonce $N_{i,3}$ is for freshness checking. After receiving the $ID_{i,0}$, $N_{i,3}$ and the encrypted message in step 5, $U_{i,j \in G_i}$ decrypted the message by first computing the shared key $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$. Then, $U_{i,j}$ can derive the group key K' by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(K', h(K' \| N_{i,3})))$. $U_{i,j \in G_i}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})$ back to $U_{i,0}$ in step 6. The nonce $N_{i,4}$ is for freshness checking.

Upon receiving the encrypted message in step 6, $U_{i,0}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4}))$ and checks if the nonce $N_{i,3}+1$ is in it for freshness checking. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,4}+1)$ back to $U_{i,j \in G_i}$ in step 7. Upon receiving $E_{V_{i,0,j}}(N_{i,4}+1)$, $U_{i,j \in G_i}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,4}+1))$ and checks if the nonce $N_{i,4}+1$ is in it for freshness checking. After finishing the steps, all group members can use the group key K' for secure communication.

3.5.2 Member leaving

If $U_{i,k}$ wants to leave the service, the group key needs to be changed. After a member sends the leave-request to RC, RC updates the group key K'' to protect the forward secrecy. The following are the common key updating protocols for each subgroup i, where $N, N_{i,1}, N_{i,2}, N_{i,3}, N_{i,4}$ are nonces and LR is the leave-request message.

Step 1: $U_{i,k} \rightarrow RC: \{ID_{i,k}, N, LR, E_{V_{i,k}}(h(ID_{i,k} \| N \| LR))\}$

Step 2: $RC \rightarrow U_{i,0}: \{N_{i,1}, LR, ID_{i,k}, E_{V_{i,0}}(K'', h(K'' \| N_{i,1} \| ID_{i,k} \| LR))\}$

Step 3: $U_{i,0} \rightarrow RC: \{E_{V_{i,0}}(N_{i,1}+1, N_{i,2})\}$

Step 4: $RC \rightarrow U_{i,0}: \{E_{V_{i,0}}(N_{i,2}+1)\}$

Step 5: $U_{i,0} \rightarrow U_{i,j \neq k,\ j \in G_i}: \{ID_{i,0}, N_{i,3}, E_{V_{i,0,j}}(K'', h(K'' \| ID_{i,0} \| N_{i,3}))\}$

Step 6: $U_{i,j \neq k,\ j \in G_i} \rightarrow U_{i,0}: \{E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})\}$

Step 7: $U_{i,0} \rightarrow U_{i,j \neq k,\ j \in G_i}: \{E_{V_{i,0,j}}(N_{i,4}+1)\}$

In step 1, $U_{i,k}$ sends his identity $ID_{i,k}$, a nonce N, LR, and the message $E_{V_{i,k}}(h(ID_{i,k} \| N \| LR)$ to RC to leave the service. If RC accepts the request,

then RC updates the group key $K^{"}$ and redistributes to all group members. In step 2, RC sends a nonce $N_{i,1}$, $U_{i,k}$'s identify $ID_{i,k}$ and the leaving message *LR*, and the message $\{N_{i,1}, LR, ID_{i,k}, E_{V_{i,0}}(K^{"}, h(K^{"} \| N_{i,1} \| ID_{i,k} \| LR))\}$ to $U_{i,0}$. The nonce $N_{i,1}$ is a fresh random number for freshness checking. The authentication tag $h(K^{"} \| N_{i,1} \| ID_{i,k} \| LR)$ is used for verifying the identification of RC.

Upon receiving the encrypted message in step 2, $U_{i,0}$ decrypts the message using his secret key and derives the new group $K^{"}$ by computing $D_{V_{i,0}}(E_{V_{i,0}}(K^{"}, h(K^{"} \| N_{i,1} \| ID_{i,k} \| LR)))$. Then it checks if the authentication tag $h(K^{"} \| N_{i,1} \| ID_{i,k} \| LR)$ is valid. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0}}(N_{i,1}+1, N_{i,2})$ back to RC in step 3. The nonce $N_{i,2}$ is for freshness checking.

After receiving the encrypted message in step 3, RC decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,1}+1, N_{i,2}))$ and checks if the nonce $N_{i,1}+1$ is in it for freshness checking. If yes, RC sends the encrypted message $E_{V_{i,0}}(N_{i,2}+1)$ back to $U_{i,0}$ in step 4. Upon receiving the message $E_{V_{i,0}}(N_{i,2}+1)$, $U_{i,0}$ decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,2}+1))$ and checks if the nonce $N_{i,2}+1$ is in it for freshness checking.

After step 4, *RC* has sent the new group key $K^{"}$ to all subgroup heads. In step 5, $U_{i,0}$ generates a nonce $N_{i,3}$, and sends his $ID_{i,0}$, $N_{i,3}$, the encrypted message $E_{V_{i,0,j}}(K^{"}, h(K^{"} \| N_{i,3} \| ID_{i,0} \| JN))$ to $U_{i, j \neq k, j \in G_i}$. The nonce $N_{i,3}$ is for freshness checking. After receiving $ID_{i,0}$, $N_{i,3}$ and the encrypted message in step 5, $U_{i, j \neq k, j \in G_i}$ decrypts the message by first computing the shared key $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$. Then, $U_{i, j \neq k, j \in G_i}$ can derive the group key $K^{"}$ by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(K^{"}, h(K^{"} \| N_{i,3})))$ and check if the authentication tag $h(K^{"} \| N_{i,3})$ is valid. If yes, $U_{i, j \neq k, j \in G_i}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})$ back to $U_{i,0}$ in step 6. The nonce $N_{i,4}$ is for freshness checking.

Upon receiving the encrypted message in step 6, $U_{i,0}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4}))$ and checks if the nonce $N_{i,3}+1$ is in it for freshness checking. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,4}+1)$ back to $U_{i, j \neq k, j \in G_i}$ in step 7. Upon the receipt of $E_{V_{i,0,j}}(N_{i,4}+1)$, $U_{i, j \neq k, j \in G_i}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,4}+1))$ and checks if the nonce $N_{i,4}+1$ is in it for freshness checking. After finishing these steps, all group members can use the new group key $K^{"}$ for secure communication.

3.5.3 Reconfiguration

If the static member $U_{i,0}$ wants to leave the service, the group key needs to be updated. Following are the re-key operations required.

3.5.3.1 The grouping phase

After the static member $U_{i,0}$ sends the leave-request to *RC*, *RC* chooses another member $U_{i,k}$ as the new static member of this subgroup G_i. For simplicity, we exchange $ID_{i,k}$ with $ID_{i,0}$ and publish it.

3.5.3.2 The key distribution phase

RC updates the group key $K^{'''}$ to protect the forward secrecy, and sends it to other members except $U_{i,k}$. The following protocols are the group key distribution for each subgroup i, where $N, N_{i,1}, N_{i,2}, N_{i,3}$, $N_{i,4}$ are nonces and *SLR* is the static member leave-request message.

Step 1: $U_{i,k} \rightarrow RC : \{ID_{i,k}, N, SLR, E_{V_{i,k}}(h(ID_{i,k} \| N \| SLR))\}$

Step 2: $RC \rightarrow U_{i,0} : \{N_{i,1}, SLR, ID_{i,k}, E_{V_{i,0}}(K''', h(K''' \| N_{i,1} \| ID_{i,k} \| SLR))\}$

Step 3: $U_{i,0} \rightarrow RC : \{E_{V_{i,0}}(N_{i,1}+1, N_{i,2})\}$

Step 4: $RC \rightarrow U_{i,0} : \{E_{V_{i,0}}(N_{i,2}+1)\}$

Step 5: $U_{i,0} \rightarrow U_{i,j\neq k,\ j\in G_i} : \{ID_{i,0}, N_{i,3}, E_{V_{i,0,j}}(K''', h(K''' \| ID_{i,0} \| N_{i,3}))\}$

Step 6 $U_{i,j\neq k,\ j\in G_i} \rightarrow U_{i,0} : \{E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})\}$

Step 7 $U_{i,0} \rightarrow U_{i,j\neq k,\ j\in G_i} : \{E_{V_{i,0,j}}(N_{i,4}+1)\}$

In step 1, $U_{i,k}$ sends his identity $ID_{i,k}$, a nonce *N*, *SLR*, and the message $E_{V_{i,k}}(h(ID_{i,k} \| N \| SLR))$ to RC for leaving the service. If RC accepts the request, then RC updates the group key K''' and redistributes it to all group members. After grouping, RC sends a nonce $N_{i,1}$, $U_{i,k}$'s identify $ID_{i,k}$ and the leaving message SLR, and the message $\{N_{i,1}, SLR, ID_{i,k}, E_{V_{i,0}}(K''', h(K''' \| N_{i,1} \| ID_{i,k} \| SLR))\}$ to the new $U_{i,0}$ in step 2. The nonce $N_{i,1}$ is a fresh random number for freshness checking. The authentication tag $h(K''' \| N_{i,1} \| ID_{i,k} \| SLR)$ is used for verifying the identification of *RC*.

Upon receiving the message in step 2, $U_{i,0}$ decrypts the message using his secret key and derives the new group K''' by computing $D_{V_{i,0}}(E_{V_{i,0}}(K''', h(K''' \| N_{i,1} \| ID_{i,k} \| SLR)))$. Then it checks if the authentication tag $h(K''' \| N_{i,1} \| ID_{i,k} \| SLR)$ is valid. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0}}(N_{i,1}+1, N_{i,2})$ back to RC in step 3. The nonce $N_{i,2}$ is for freshness checking.

After receiving the encrypted message in step 3, RC decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,1}+1, N_{i,2}))$ and checks if the nonce $N_{i,1}+1$ is in it for freshness checking. If yes, RC sends the encrypted message $E_{V_{i,0}}(N_{i,2}+1)$ back to $U_{i,0}$ in step 4. Upon receiving the message $E_{V_{i,0}}(N_{i,2}+1)$, $U_{i,0}$ decrypts it by computing $D_{V_{i,0}}(E_{V_{i,0}}(N_{i,2}+1))$ and checks if the nonce $N_{i,2}+1$ is in it for freshness checking.

After step 4, *RC* has sent the new group key K''' to all subgroup heads. In step 5, $U_{i,0}$ generates a nonce $N_{i,3}$, and sends his $ID_{i,0}$, $N_{i,3}$, the encrypted message $E_{V_{i,0,j}}(K''', h(K''' \| N_{i,3} \| ID_{i,0} \| SLR))$ to $U_{i,j\neq k,\ j\in G_i}$. The nonce $N_{i,3}$ is for freshness checking. After receiving the $ID_{i,0}$, $N_{i,3}$ and the encrypted message in step 5, $U_{i,j\neq k,\ j\in G_i}$ decrypted the message by first computing the shared key $V_{i,0,j} = h(ID_{i,0}, V_{i,j})$. Then, $U_{i,j\neq k,\ j\in G_i}$ can derive the group key K''' by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(K''', h(K''' \| N_{i,3})))$ and check if the authentication tag $h(K''' \| N_{i,3})$ is valid. If yes, $U_{i,j\neq k,\ j\in G_i}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4})$ back to $U_{i,0}$ in step 6. The nonce $N_{i,4}$ is for freshness checking.

Upon receiving the encrypted message in step 6, $U_{i,0}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,3}+1, N_{i,4}))$ and checks if the nonce $N_{i,3}+1$ is in it for freshness checking. If yes, $U_{i,0}$ sends the encrypted message $E_{V_{i,0,j}}(N_{i,4}+1)$ back to $U_{i,j\neq k,\ j\in G_i}$ in step 7. Upon the receipt of $E_{V_{i,0,j}}(N_{i,4}+1)$, $U_{i,j\neq k,\ j\in G_i}$ decrypts it by computing $D_{V_{i,0,j}}(E_{V_{i,0,j}}(N_{i,4}+1))$ and checks if the nonce $N_{i,4}+1$ is in it for freshness checking. After finishing all steps, all group members can use the new group key K''' for secure communication.

4. Security analysis

In this section, we analyze the security of our proposed scheme.

(1) The secret key $V_{i,j} = h(x, ID_{i,j})$ of each member

computed by RC and the shared secret key $V_{i,0,j} = h(V_{i,j}, ID_{i,0})$ between $U_{i,0}$ and $U_{i,j}$ are both protected by the secure one-way hash function $h()$. It is infeasible to compute $V_{i,j}$ without knowing the secret information x of RC. It is also infeasible to compute $V_{i,0,j}$ without knowing the secret information x or $V_{i,j}$.

(2) The replay attack fails since the freshness of messages transmitted in key distribution phase is provide by the nonces $N, N_{i,1}, N_{i,2}, N_{i,3}, N_{i,4}$. Only valid participants can put the related nonces generated by the eligible partner in the encrypted message. The authentication tags are used to make sure that the received messages were correctly sent by the eligible partners.

(3) In the dynamic membership management phase, the group key is updated when members join, leave, or reconfigure. It provides forward and backward secrecy.

In our scheme, for improving the reparability mentioned in [9, 10], the shared keys $V_{i,j} = h(x, ID_{i,j})$ stored in $U_{i,j}$'s smart card can be replaced with the new value $V_{i,j} = h(x, ID_{i,j}, k)$, where *k* is the number of times that $U_{i,j}$ has revoked his used secret key $V_{i,j}$. This approach requires *RC* to record the number *k* in his database or $U_{i,j}$ to send the number *k* to his subgroup controller in key distribution.

5. Performance analysis

Table 1: Number of keys stored for our proposed scheme and related schemes

	Iolus [13]	Aslan's scheme [1]	Our Scheme
Group controller	N	n	1
Subgroup controller	N	2N-1	N
Each member	1	h	1

We analyze the number of stored keys for our proposed scheme and the related schemes [1, 13] in Table 1. We assume the total number of subgroups in [1, 13] and our scheme is n. The number of each subgroup members is N. Therefore, members of the whole group are $n*N$. The height of the protocol's tree in [1] is $h = (\log_d N) + 1$, where d is the degree of the tree. In our scheme, the shared key $V_{i,j} = h(x, ID_{i,j})$ and $V_{i,0,j} = h(V_{i,j}, ID_{i,0})$ can be computed by the master secret *x*, and RC only has to keep secretly the master key *x*. Each subgroup controller needs to store N-1 shared keys $V_{i,0,j\neq i} = h(V_{i,j}, ID_{i,0})$ inquired from the group controller with his subgroup member and one shared key $V_{i,0} = h(x, ID_{i,0})$ with the group controller. Each subgroup member $U_{i,j}$ only needs to store one secret key $V_{i,j} = h(x, ID_{i,j})$. In Iolus [13], the group controller needs to store n shared keys with n subgroup controllers. Each subgroup controller needs to store N-1 shared keys with his N-1 subgroup members and at least one shared key with his nearest subgroup controller. Each subgroup member only needs to store one shared secret key with his subgroup controller. In Aslan's protocol [1], the group controller needs to store n shared keys with n subgroup controllers. Each subgroup controller needs to store 2N-2 shared keys, including N-1 KEKs and N-1 LKH keys, with his N-1 subgroup members and at least one shared key with his nearest subgroup controller. The total number of keys is (N-1)+(N-1)+1=2N-1. Each subgroup member needs to store h shared secret keys with his subgroup controller.

Table 2: Computation cost of a multicast message for our proposed scheme and related schemes

	Iolus [13]	Aslan's scheme [1]	Our Scheme
Number of encryption/decryption operations for all subgroup controllers	3n-2	3n-2	0
Number of encryption/decryption operations for each subgroup member	1	1	1

The number of encryption/decryption operations for a multicast message in each subgroup controller and member about our scheme and related schemes is presented in Table 2. In our scheme, all subgroup controllers need not do any encryption or decryption for multicasting a message. Each subgroup member needs to do one encryption or decryption operation when a

multicast message is transmitted. In the schemes [1, 13], all subgroup controllers need to do message translation. This task needs at least 3n-2 encryption or decryption operations, including 1 decryption operation and n-1 encryption operation by the sender subgroup controller, and n-1 decryption operations and n-1 encryption operations for all other n-1 subgroup controllers.

If you would like to itemize some parts of your manuscript, please make use of the specified style "itemize" from the drop-down menu of style categories

In the case that you would like to paragraph your manuscript, please make use of the specified style "paragraph" from the drop-down menu of style categories

Table 3: Comparison between our proposed scheme and related schemes

	Iolus [13]	Aslan's scheme [1]	Our Scheme
No key table	No	No	Yes
Mutual authentication	No	No	Yes
Computation cost of multicasting messages	High	High	Very low
Secret key generation	No	No	Yes

We summarize the functionality of the related schemes [1, 13] and our scheme in Table 3. In the schemes [1, 13], the group controller must store the shared keys for all subgroup controllers. In our scheme, the registration center RC needs only to protect his master secret key x. The subgroup controller and subgroup members can authenticate each other in our scheme. The computation cost of multicasting messages in our scheme is very low compared to the schemes [1, 13]. We also address the shared keys generation between RC and subgroup controllers, RC and subgroup members, and subgroup controllers and subgroup members**.**

6. Concluding remarks

In this paper, we have proposed an efficient and flexible decentralized multicast key distribution scheme. Our scheme does not need a shared keys table between the registration center and all users to reduce the key storage risk, and the computation cost is very low. In our scheme, when a subgroup member wants to transmit messages to other subgroups, these messages can be encrypted by the group key. It does not have the bottleneck problem in subgroup heads. The subgroup heads can authenticate each subgroup members. The Key Encryption Keys distribution between the key center and all members are addressed in our proposed scheme but not considered in other related schemes.

Acknowledgments

This work was supported in part by the National Science Council of Republic of China under contract NSC 94-2213-E-128- 001 and NSC 95-2221-E-128-004-MY2.

References

[1] H. Aslan, "A Scalable and Distributed Multicast Security Protocol Using A Subgroup-Key Hierarchy," Computers and Security, Vol. 23, pp. 320-329, 2004.

[2] A. Ballardie, "Scalable Multicast Key Distribution," RFC 1949, 1996.

[3] M. Burmenster, Y. Desmedt, "A Secure and Efficient Conference Key Distribution System," In Advances in Cryptology — EUROCRYPT 94, A. D. Santis, Ed., Lecture Notes in Computer Science 950, Springer-Verlag, New York, pp. 275–286, 1994.

[4] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, "Multicast Security: A Taxonomy and Some Efficient Constructions," In Proceedings of the IEEE INFOCOM, Vol. 2, pp. 708–716, 1999.

[5] R. Canetti, T. Malkin, K. Nissim, "Efficient Communication-Storage Tradeoffs for Multicast Encryption," In Advances in Cryptology — EUROCRYPT '99, J. Stem, Ed. Lectures Notes in Computer Science 1599. Springer-Verlag, New York, pp. 459–474, 1999.

[6] H. Harney, C. Muckenhirn, "Group Key Management Protocol (GKMP) Specification," RFC 2093, 1997.

[7] H. Harney, C. Muckenhirn, "Group Key Management Protocol (GKMP) Architecture," RFC 2094, 1997.

[8] G. Horng, "Cryptanalysis of A Key Management Scheme for Secure Multicast Communications," IEICE Trans. Communications, Vol. E85-B, No. 5, pp. 1050-1051, 2002.

[9] T. Hwang and W. Ku, "Repairable Key Distribution Protocols for Internet Environments," IEEE Trans. on Communications, Vol. 43, No. 5, pp. 1947-1950, 1995.

[10] W. Ku and S. Chen, "Weaknesses and Improvements of an Efficient Password Based Remote User Authentication Scheme Using Smart

Cards," IEEE Trans on Consumer Electronics, Vol. 50, No. 1, pp. 204-207, 2004.

[11] M. Li, R. Poovendran, C. Berenstein, "Design of Secure Multicast Key Management Schemes with Communication Budget Constraint," IEEE Communications Letters, Vol. 6, No.3, pp.108-110, 2002.

[12] R. Merkle, "One Way Hash Functions and DES," In Brassard, G. (ed.), Advances in Cryptology-Crypt'89, LNCS 435, pp. 428-446, Springer, New York, 1989.

[13] S. Mittra, "Iolus: A Framework for Scalable Secure Multicasting," In Proceedings of ACM SIGCOMM, pp. 277-288, 1997.

[14] NIST FIPS PUB 197, "Announcing the Advanced Encryption Standard (AES)", National Institute of Standards and Technology, U. S. Department of Commerce, 2001.

[15] A. Perrig, D. Song, J. Tygar, "A New Protocol for Efficient Large-Group Key Distribution," In Proceedings of the IEEE Symposium on Security and Privacy, IEEE Computer Society Press, Los Alamitos, Calif., 2001.

[16] S. Rafaeli, D. Hutchison, "A Survey of Key Management for Secure Group Communication," ACM Computing Surveys, Vol. 35, No. 3, pp.309-329, 2003.

[17] O. Rodeh, K. Birman, D. Dolev, "Optimized group rekey for group communication systems," In Network and Distributed System Security, San Diego, Calif., Feb. 2000.

[18] M. Steiner, G. Tsudik, M. Waidner, "Diffie-Hellman Key Distribution Extended to Group Communication," In SIGSAC Proceedings of the 3rd ACM Conference on Computer and Communications Security, New York, pp. 31–37, 1996.

[19] R. Wittmann, M. Zitterbart, Multicast Communication Protocols and Applications, Morgan Kaufman Publishers, 2001.

[20] C. Wong, M. Gouda, L. Simon, "Secure Group Communication Using Key Graphs," IEEE/ACM Transactions on Networking, Vol. 8, No.1, pp.16-30, 2000.

Wen-Shenq Juang received his master's degree in Computer Information Science from National Chiao Tung University in 1993, and his Ph. D. degree in electrical engineering from National Taiwan University in 1998. He joined the Department of Information Management, Shih Hsin University, Taipei, Taiwan, in 2000 as an assistant professor. He is current an associate professor in the same department. Dr. Juang's current research interests include applied cryptography, information security, and network security.

Jyan-Cwan Wu is now working toward MS degree in Information Management from Shih Hsin University, Taiwan. His current interests include information security, electronic commerce, and network security.