Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 601625, 10 pages
http://dx.doi.org/10.1155/2014/601625

Research Article

Secure Collaborative Key Management for Dynamic Groups in Mobile Networks

Sukin Kang, Cheongmin Ji, and Manpyo Hong

Department of Computer Engineering, Ajou University, Suwon 443-749, Republic of Korea

Correspondence should be addressed to Manpyo Hong; mphong@ajou.ac.kr

Received 29 March 2014; Accepted 31 July 2014; Published 21 August 2014

Academic Editor: Young-Sik Jeong

Mobile networks are composed of heterogeneous mobile devices with peer-to-peer wireless communication. Their dynamic and self-organizing natures pose security challenge. We consider secure group key management for peer dynamic groups in mobile wireless networks. Many group based applications have achieved remarkable growth along with increasing use of multicast based services. The key sharing among the group members is an important issue for secure group communication because the communication for many participants implies that the likelihood of illegal overhearing increases. We propose a group key sharing scheme and efficient rekeying methods for frequent membership changes from network dynamics. The proposed method enables the group members to simply establish a group key and provide high flexibility for dynamic group changes such as member join or leave and group merging or partition. We conduct mathematical evaluation with other group key management protocols and finally prove its security by demonstrating group key secrecy, backward and forward secrecy, key independence, and implicit key authentication under the decisional Diffie-Hellman (DDH) assumption.

1. Introduction

Advances in wireless communications and mobile devices have made various types of mobile networks such as mobile ad hoc networks (MANETs), wireless mobile sensor networks (WMSNs), and Internet of things (IoT). In mobile networks, heterogeneous devices such as smartphones, laptops, and smart sensors perform peer-to-peer (machine-to-machine) communications without depending on any fixed infrastructure. Mobile networks have features distinct from conventional networks. First, network topology changes dynamically due to the mobility of nodes, which causes frequent switching of network connection state. Additionally, many applications in mobile networks support one-to-many (multicast) communication, where common data are transferred to multiple destinations from a source, for instance, military communication (battlefield), health care system, industrial monitoring, on-line conferencing, collaborative workspace, and disaster management. They build a collaborative group of entities, called group members, which participate in multicast group communications as a group member and manage group membership changed by node mobility.

Group communication over wireless networks is susceptible to illegal overhearing such as packet sniffing. When a group deals with sensitive information, secure group communication must be achieved by sharing a common secret key—*group key* for confidentiality of group messages with data encryption. In other words, it is essential to decide how to share a key among group members and how to update the group key for group membership change [1–3]. A typical approach is based on centralized key distribution with a trusted third party (TTP) [4–8]. It provides scalable group key management for large groups using symmetric encryption such as advanced encryption standard (AES) and hierarchical logical key tree. However, it fairly depends on a constantly accessible TTP. This requirement is not suitable for mobile networks with peer-to-peer communication. To apply a symmetric key based approach without a TTP, a node

should establish secure connection for sharing a pairwise key with all other mobile nodes in a group. It requires much communication and depends on another key sharing scheme [9]. Diffie-Hellman (DH) key exchange [10] is a protocol to establish a common key based on asymmetric keys without any TTP. It allows two parties to share a key using their secrets over an insecure channel. To extend DH into group setting, group key agreement (GKA) protocols have been developed [11–16]. In the protocols, also known as contributory key agreement, all group members contribute to generation of a common key. While providing dynamic group key management, they require considerable messages or operations to establish and update group keys. An approach for reducing computation cost deploys tree structure to handle key management. Tree-based group key protocols [15–18] need to support management of tree structure and require ordered message delivery for calculation from leaves to the root of the tree.

In this paper, we investigate secure group key distribution and management for collaborative groups with high group flexibility. We propose a DH-based group key management protocol and show security proof of the proposed scheme and mathematical evaluation with other GKA protocols.

The remainder of the paper is organized as follows. In Section 2, we address related works. Section 3 explains our group key management scheme with group membership events and security requirements. Section 4 describes performance analysis and Section 5 shows security proof for the proposed key management. We conclude the paper in Section 6.

2. Related Work

Over the past few decades, a considerable number of studies have been conducted on group key establishment and management. A typical approach is centralized key distribution based on constantly accessible TTP and pairwise keys [4–8]. These studies showed apparent efficiency for large groups such as wireless sensor network (WSN). Since, however, a mobile network is comprised of peer-to-peer communications with dynamic mobility and without a TTP, it is difficult to provide scalable group key management on arbitrary group setting [15].

We focus on DH based group key management, known as group key agreement (GKA), in which a common key is generated by all group members' equal contributions. DH protocol allows two parties to share a key using their secrets over an insecure channel [10]. The key computation of DH uses the multiplicative group of integer modulo p, where p is a large prime number. Each party chooses a random number x_i in $\mathbb{Z}_p$ and computes $g^{x_i} \bmod p$, where g is a primitive root (generator) mod$p$. They exchange the computed values, $g^{x_1} \bmod p$ and $g^{x_2} \bmod p$, and agree on the common key:

$$K = \left(g^{x_1}\right)^{x_2} \bmod p = \left(g^{x_2}\right)^{x_1} \bmod p. \quad (1)$$

For extending it to group setting, Burmester and Desmedt (BD) proposed a conference key exchange system [11] depending on a broadcast manner. When the number of group members is n, the group key (GK) of BD becomes

$$\mathrm{GK} = g^{x_1x_2 + x_2x_3 + \cdots + x_{n-1}x_n} \bmod p. \quad (2)$$

As BD system requires large communication messages, Steiner et al. proposed group key agreement protocols called group Diffie-Hellman (GDH) [12, 13]. In GDH,

$$\mathrm{GK} = g^{x_1x_2 \cdots x_{n-1}x_n} \bmod p. \quad (3)$$

They showed not only that DH can be extended efficiently to group setting, but also that their protocol can deal efficiently with group membership change. They presented three distinct group key agreements GDH.1, GDH.2, and GDH.3, which later was advanced as a protocol suite known as CLIQUES [13]. In GDH.x, group members can individually or massively join and leave; CLIQUES also considers group integration and group division. A variant of GDH protocol is a centralized key distribution (CKD) scheme. In CKD, a controller distributes the group key to every member using pairwise temporal keys between the controller and each of the members, which is computed using DH fashion.

As group dynamics have become an important issue, some studies have adopted tree-based approach [15–18]. Skinny tree (STR) protocol [16] has good performance for member addition. In STR,

$$\mathrm{GK} = g^{x_n g^{x_{n-1} g^{\cdots g^{x_3 g^{x_1x_2}}}}} \bmod p. \quad (4)$$

While STR uses unbalanced key tree for group key computation, tree-based group Diffie-Hellman (TGDH) leverages balanced tree structure. Given eight group members in TGDH, the group key is computed as follows:

$$\mathrm{GK} = g^{g^{g^{x_1x_2}\, g^{x_3x_4}}\, g^{g^{x_5x_6}\, g^{x_7x_8}}} \bmod p. \quad (5)$$

STR and TGDH require a sponsor node which distributes intermediate computing keys in the tree during membership event changes. As tree-based protocols apparently help to reduce communication cost and operation cost, there have been several variants of TGDH [17, 18]. However, they need to support management for tree balance and require message delivery order due to hierarchical tree structure. In mobile networks, much communication would be required to make sure that the group members can keep the synchronized tree structure.

In summary, DH-based group key protocol is generally known as GKA protocol. Although our protocol is based on DH, we do not classify it as a GKA protocol because of key distribution feature from a controller. Our proposed scheme provides the advantage of dynamics and collaborative contribution in computing group keys with a modified key agreement method.

3. Secure Group Key Management for Mobile Networks

3.1. Membership and Security Requirements. Group membership events occur with either insertion of a new node or

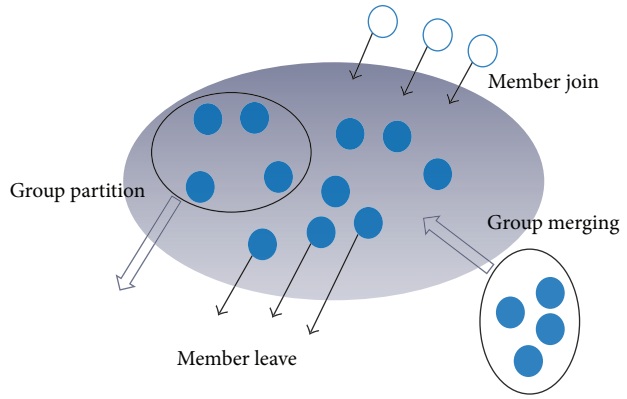

Figure 1: Four kinds of membership events; (1) member join (single join or mass join), (2) member leave (a single leave or mass leave), (3) group merging (group join), and (4) group partition (group leave). A small circle represents a node while a big circle represents a group of nodes.

deletion of an existing member. We define the insertion event as *member join* and the deletion event as *member leave*. When there is only one event node specifically, we call each *single join* and *single leave*, and when there are two or more event nodes we call each *mass join* and *mass leave*. Furthermore, we consider a group insertion into a group and a group partition into two distinct groups. We define them as *group merging* and *group partition*, respectively. Figure 1 shows summary of defined membership events.

Group membership change is closely related to security of group communication. Outgoing members should have no access to group communication after it leaves the group, and ingoing nodes should be prevented from accessing previous group communication before it joins the group. We define cryptographic properties in which a secure group, depending on a group key, should meet (1) *group key secrecy* that guarantees an adversary who knows that messages sent to group members cannot discover any group key in polynomial time, (2) *backward secrecy* that guarantees a new member or an adversary who knows that the current group key cannot discover any previous group key in polynomial time, (3) *forward secrecy* that guarantees a former group member or an adversary who knows that previous group keys cannot discover any subsequent group key in polynomial time, (4) *key independence* that guarantees an adversary who knows that a proper subset of group keys cannot discover any other group keys in polynomial time, and (5) (*implicit*) *key authentication* that guarantees that no one apart from a group member recovers the group key.

3.2. Group Key Establishment. We present a new group key protocol, collaborative Diffie-Hellman (CODH). CODH has centralized topology and key distribution property from a leader node. But, unlike conventional centralized scheme with TTP, in CODH, a group leader computes and distributes a group key by using public keys of group members. We formalize the group key protocol and prove its security.

CODH has one leader called *master*. The leader is also one of group members. It consumes more energy than normal nodes for communication and operation in managing group keys. There will be a policy for choosing a leader. In mobile networks, signal strength, degree to neighbors, identity, and resources (CPU, memory, battery, and bandwidth) would be criteria for leader election [19–21]. When a group is created, the first master is elected among group members and performs group key initialization. Afterwards, group members select a new master when receiving master notification for leader change. Once a new group master is selected for group management, the previous master forwards information about group members to the new master; that is, a delegation process is run (refer to Sections 3.3 and 3.4). On the other hand, connection failure may occur by network isolation or denial of service attacks. (We assume that group participants are honest and not compromised. However, they can be threatened by network adversaries who can perform all of network-based attacks.) We consider the connection failure as a kind of member leave whether the left node is a member or the master.

Notation section represents notations used to illustrate our group key protocol. The index "s" stands for the master node in a group that is distinct from i or j which indicates a general member node. Therefore, M_i or M_j means an identity for general member, while M_s denotes the master. *Lock-secret* is defined as a secret value of a member. It locks the group key so that M_s can securely transfer the group key to the members. General members use their *unlock-secret* to extract the group key from M_s's broadcast message of a locked group key.

We adopt inverse exponentiation for obtaining the group key. Let C_n be a group of size n; that is, $C_n = \{M_1, M_2, \ldots, M_n\}$ and $M_s \in C_n$. To share the initial group key, the group C_n runs steps in Box 1 for the initial phase.

The initial phase consists of two rounds. In the first round, all members except the group master send their locker g^{x_i} to the master via unicast and the master produces the locker list, XL_C, from receiving messages. In the second round, the master M_s selects a random secret k and computes and broadcasts the locked group key $(X_i)^k = (g^{x_i})^k$ using XL_C. Then, each member can compute the group key GK using their own unlock-secret, y_i, as follows:

$$\mathrm{GK} \equiv \left(X_i^k\right)^{y_i} \bmod p \equiv \left(g^{x_i y_i}\right)^k \bmod p \equiv g^k \bmod p. \quad (6)$$

The group key is equal to the locker of the group master when k is the master's secret. Therefore, operations for computing X_i^k and group messages never include X_s.

3.3. Group Rekeying for Member Join and Leave. The master-secret should be renewed when membership changes, since it is used for the new group key GK$'$. In Box 2 (member join process), k' means a new master-secret that M_s selects. Let M_{n+1} be the first new member and let M_{n+m} be the last new member, when m new members join the group C_n (if a single member joins, the new member is only one node, M_{n+1}). A new member M_j $(n+1 \le j \le n+m)$ sends its locker X_j to the

Assume that the group of n members establish a group key.
Step 1. Each member selects random $x_i \in \mathbb{Z}_q$ and computes $X_i = g^{x_i} \bmod p$.
$M_i \rightarrow M_s$: X_i $(i \in [1, n], i \neq s)$
Step 2. M_s selects random k in $\mathbb{Z}_q$ for group key sharing and computes key-locks.
$M_s \Rightarrow C_n$: $\{(X_i)^k \mid i \in [1, n], i \neq s\}$

Box 1: Group key initialization.

Assume that m members are added to the group C_n.
Step 1. Each new member M_j $(n+1 \leq j \leq n+m)$ selects random $x_j \in \mathbb{Z}_q$ and computes $X_j = g^{x_j} \bmod p$.
$M_j \rightarrow M_s$: X_j $(j \in [n+1, n+m])$
Step 2. M_s selects random k' in $\mathbb{Z}_q$ for new group key and computes key-locks.
$M_s \Rightarrow C_n$: $\{(X_i)^{k'} \mid i \in [1, n+m], i \neq s\}$

Box 2: Group rekeying for member join.

Assume that a subset L_m of current group C_n is composed of m leaving members in the group and does not include the group master M_s.
Step 1. M_s selects random k' in $\mathbb{Z}_q$ for new group key and computes key-locks with updated locker list.
$M_s \Rightarrow C_n \setminus L_m$: $\{(X_i)^{k'} \mid i \in [1, n] \wedge M_i \notin L_m, i \neq s\}$

Box 3: Group rekeying for member leave.

master, and then M_s broadcasts locked new group key $\mathrm{GK}' = g^{k'}$ to all the group members in the same manner as second round of initial phase, as in Box 2. All members, including new members, can extract the new group key GK' in the same way as (6).

Unlike the join event, member leave process does not require the first round for sending lockers to the master. Let a subset of C_n for leaving members be $L_m \subset C_n$ $(M_s \notin L_m)$. Group members conduct rekeying operations for the new group key GK' as in Box 3.

The leaving nodes cannot learn the new group key because the broadcast message from M_s does not contain any locker X_i for leaving members. Note that the set L_m for the leaving node does not include the master. Leaving of the master requires 'delegation' during which the master forwards locker list XL_C for group C_n to new group master($M_{s'}$) as follows:

$$M_s \longrightarrow M_{s'} : XL_C = \left\{X_j \mid M_j \in C_n, j \neq s\right\}. \quad (7)$$

The delegation can be used for another case where the master wishes to finish its master's role for a reason such as network topology change or resource exhaustion; that is, the master turns to a group member not leaving the group. In this case, the delegation message includes the former master's locker generated with new selected secret x_s as follows:

$$M_s \longrightarrow M_{s'} : XL_C = \left\{X_j \mid M_j \in C_n, x_s \neq k, x_s \neq k'\right\}. \quad (8)$$

When group members detect unexpected disconnection from the master, they restart group key initialization with new master selection. At the worst case, members can suffer from frequent connection failure with the master. In this case, the first protocol should be slightly modified to make all of group members have the locker list and any member be the group master to proceed Box 3. For instance, a general member at the first step of Box 1 broadcasts its locker to the group as follows:

$$M_i \Longrightarrow C_n : X_i \, (i \in [1, n], i \neq s). \quad (9)$$

The group members continue secure communication with a fresh group key obtained through group rekeying. We provide formal security proofs in Section 5.

3.4. Group Rekeying for Group Merging and Partition. There are two ways to integrate two groups into one group completely: *individual join* and *group join*. The former is that members of a group join another group individually. It is similar to the mass joining process, saving that the joining master should generate his lock-secret, x_s, and locker, g^{x_s}. The latter way is that a group is absorbed into the other group via delegation process between both group masters.

Let two groups be merged $C_n = \{M_1, M_2, \ldots, M_n\}$ and $R_m = \{M_1, M_2, \ldots, M_m\}$ $(n \geq m)$. The master M_s of C_n survives after group merging, while the master $M_{s'}$ of R_m becomes a member of the merged group. Smaller group members $(\in R_m)$ become a member of C_{n+m}; that is, $C_{n+m} = \{M_1, M_2, \ldots, M_n, M_{n+1}, \ldots, M_{n+m}\}$ and $M_s \in C_n$ after group merging. Group merging process runs with delegation (in the first round) as in Box 4. Figure 2 represents an instance for a merging process for a current group C_4 and a merged group

Assume that a group R_m is merged into a group C_n where $n \geq m$, and the merged group $C_{n+m} = C_n \cup R_m$. $M_{s'}$ is the master of R_m and M_s is the master of C_n.
Step 1. $M_{s'}$ selects a random number $x_{s'}$ in $\mathbb{Z}_q$, computes $X_{s'} = g^{x_{s'}} \bmod p$, and updates the locker list into $XL_R = \{X_1, X_2, \ldots, X_m\} \cup X_{s'}$.
(*delegation*) $M_{s'} \rightarrow M_s$: XL_R
Step 2. M_s selects random k' in $\mathbb{Z}_q$ for new group key and computes key-locks with updated locker list.
$M_s \Rightarrow C_{n+m}$: $\{(X_i)^{k'} \mid i \in [1, n+m], i \neq s\}$

Box 4: Group merging.

Assume that a current group C_n is partitioned into two groups, P_m ($\subset C_n$) and C_{n-m} ($=C_n \setminus P_m$).
The master of R_m is $M_{s'}$ and the master of C_n is M_s ($\notin P_m$)
Step 1. M_s generate $XL_P = \{X_j \mid M_j \in P_m, j \neq s\}$ from XL_C.
(*delegation*) $M_s \rightarrow M_{s'}$: XL_P
Step 2. M_s and $M_{s'}$ select random k', k'' in $\mathbb{Z}_q$ respectively and compute key-locks with their locker list.
$M_s \Rightarrow C_{n-m}$: $\{(X_i)^{k'} \mid i \in [1, n] \wedge M_i \notin P_m, i \neq s\}$
$M_{s'} \Rightarrow P_m$: $\{(X_j)^{k''} \mid M_j \in P_m, j \neq s\}$

Box 5: Group partition.

R_3. In Figure 2, the number in a circle indicates members' index (such as by a joined order). Before they are merged, the number of the current group C is four including the group master (i.e., $n = 4$, C_4) and the number of members of joining group is three (i.e., $m = 3$, R_3). To be merged, the master of R_3 sends the master of C_4 the locker list XL_R for M_1 and M_2. Note that the master M_s of R_m must forward its locker after changing its own secret because it was used as the former group key. The master of C_4 becomes the master for the merged group. It updates XL_C and generates key-locks XL_C^k with a new selected random k.

As shown in Figure 3, the current group will be divided into two groups. When the number of left members is m, the current group will have $(n - m)$ members after the partition process. Group partition requires one more master $M_{s'}$ for a separated subgroup $P_m \subset C_n$ ($M_s \notin P_m$). Group partition process can be easily conducted through delegation, from the master M_s of group C_n to the fresh master $M_{s'}$ of subgroup P_m. The divided groups perform a group key initial phase after the delegation process, as in Box 5.

3.5. Implicit Key Authentication. For the secure key authentication, the messages sent from all members should be signed with a signature key. Hash-based signature such as message authentication code (MAC) is fairly efficient in terms of computation cost. However, it is too costly to share one-to-one pairwise keys between all of group members in advance.

We assume that a member holds long-term private and public keys certified by a trusted certificate authority (CA). (Each member can use a different signature algorithm such as RSA-based signature algorithm, digital signature algorithm (DSA), and elliptic curve digital signature algorithm (ECDSA). Note that some of them do not provide message encryption; that is, it is used for message signing and verifying. We consider that DSA is better for our scheme since its public key includes $g^x \bmod p$.) The group members send to the master the signed messages with their own private key; for example, in the first step of Box 1, a member, M_i, sends to the master $\{X_i, Sig_{M_i}(X_i)\}$ which M_i signs for X_i with its private key. Note that this process runs one-time at initial phase or it can be precomputed with X_i.

Members can obtain the group key securely by verifying the messages of the master with signature signed with the master's private key. All of messages from the master come with a master-signed signature for the origin and integrity of a group key. For example, in the second step of Box 1, the master broadcasts $\{X_1^k, X_2^k, \ldots, X_n^k, Sig_{M_s}(\mathrm{GK})\}$. The master produces a locked set for the group key using verified members' locker. It implies that outsiders cannot recover the group key from the master's messages.

4. Evaluation

We measure performance of the proposed scheme through communication and computation cost spent for all group members to complete group rekeying by membership change. Table 1 shows summary of comparison with other DH-based key management protocols: CKD, GDH, BD, STR, and TGDH. In Table 1, n, m, and p denote the number of current group members, joining or merged-group members, and leaving or partitioned-group members, respectively. Therefore, $m = 1$ or $p = 1$ indicates the single-member event. For TGDH, the height of the key tree is denoted as h, and, for STR, s is denoted as the index of the sponsor, which helps other members to calculate group keys. Group merging is a case where a group of m members is merged into a group of n members ($n \geq m$), and group partition is a case where a group of n members is divided into separate subgroups: (1) a group

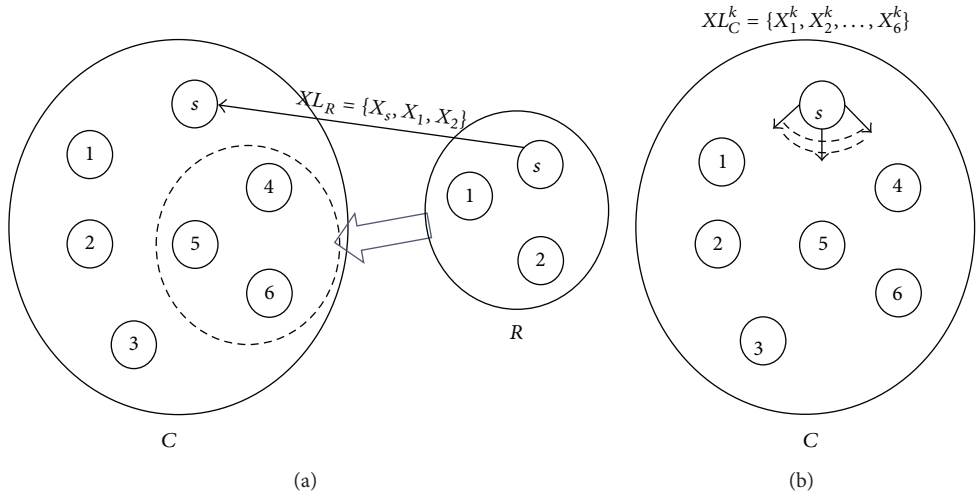

Figure 2: Group merging process: (a) when groups C_4 and R_3 are merged, R's master sends the locker list of R to C's master and (b) after groups are merged, C's master becomes the master for merged group and broadcasts the key-locks for new group key to all of the members.

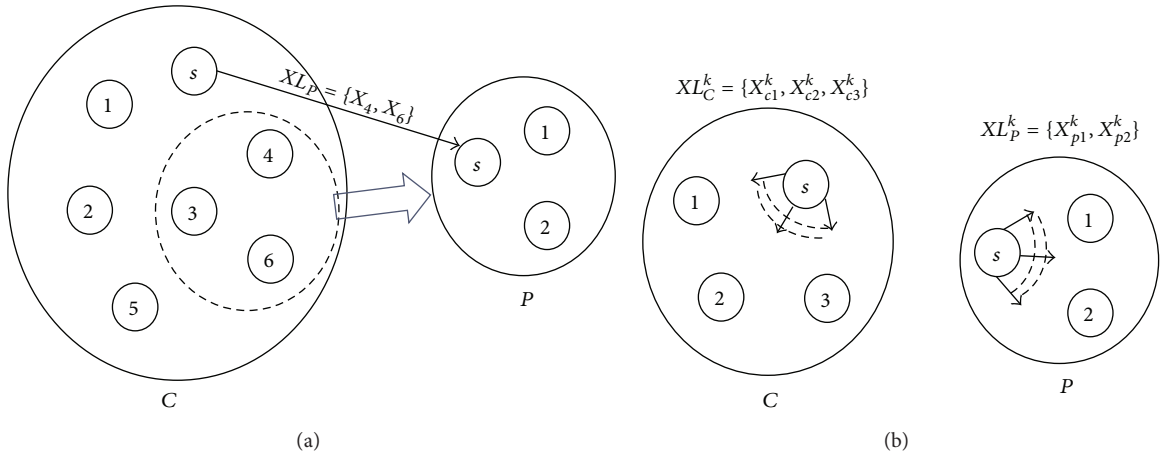

Figure 3: Group partition process: (a) when group C_7 is partitioned into two groups (new group P_3), the master of the original group sends P's master the locker list of the new subgroup and (b) after the group is split, each group master broadcasts the key-locks for each new group key.

of p members and (2) a group of $(n - p)$ members, where $(n-p) \geq p$. The costs for the group partition event include the costs for updating two subgroup keys. In computation costs, we consider concurrent execution in distributed nodes if it is possible. In CODH, we assume the master is selected by group-join order; the first master is M_1, and when M_1 leaves the group, M_2 becomes the next master.

CKD distributes the group key in a similar way with our protocol. Its communication and computation costs are also similar to our protocol. However, the worst case of CKD is when the master leaves. It requires large costs for rekeying. On the other hand, in CODH, the rekeying cost for a leaving master is analogous to that for a leaving member due to efficient delegation or sharing of public locker list. GDH is operated through communication chain from the first node to the last node, and the last node becomes the master of the group. Steiner et al. presented three GDH protocols: GDH.1, 2, and 3. GDH.2 is the most efficient in communication whereas GDH.3 is the most efficient in computation cost among GDH.x. We select GDH.3 for comparison. As shown in Table 1, GDH has weaknesses in group merging and mass joining. BD employs a completely distributed way using broadcast messages. Without sponsors or controllers, all of members broadcast messages for updating the group key. Although it seems to be fairly efficient in computation cost, there are hidden costs for multiplications. In addition, it requires a large communication cost compared to other protocols. STR and TGDH are tree-based key agreement protocols. They use different tree structures for key management. STR, especially, uses the extremely unbalanced tree structure.

Table 1: Communication and computation costs.

		Rounds	Messages	Exponentiations	Signatures	Verifications
GDH	Join, merge	$m+3$	$n+2m+1$	$n+2m+1$	$m+3$	$n+2m+1$
	Leave	1	1	$n-p$	1	1
	Partition	2	$p+1$	$n-p$	2	p
	Master leave	2	$n-1$	$n-1$	2	$n-1$
STR	Join	3	$m+2$	$3m$	2	$m+2$
	Leave	1	1	$3(n-p)-2s-1$	1	1
	Merge	3	3	$3m-1$	2	3
	Partition	2	3	$3(n-p)-2s-1$	2	3
TGDH	Join, merge	2	3	$3h-3$	2	3
	Leave, partition	h	$2h$	$3h$	h	h
BD	Join, merge	2	$2n+2m$	3	2	$2n+2m-2$
	Leave	2	$2n-2p$	3	2	$2n-2p-2$
	Partition	2	$2n$	3	2	$2n-2p-2$
CKD	Join, merge	3	$m+2$	$n+2m$	3	$m+2$
	Leave	1	1	$n-p$	1	1
	Partition	3	$p+2$	$\max(n-p, 2p-1)$	3	4
	Master leave	3	n	$2n-3$	3	n
CODH	Join	2	$m+1$	$n+m+1$	2	$m+1$
	Leave	1	1	$n-p$	1	1
	Merge	2	2	$n+m$	2	2
	Partition	2	3	$n-p$	2	2
	Master leave	2	2	$n-1$	2	2

Table 2: Communication and computation costs for CODH member and master.

		Send	Receive	Exponentiations	Signatures	Verifications
General member	Join, merge	0	1	1	0	1
	Leave, partition	0	1	1	0	1
Group master	Join	1	m	$n+m$	1	1
	Leave, partition	1	0	$n-p$	1	0
	Merge	1	1	$n+m$	1	1

Accordingly, the performance of STR depends on the location of the sponsors. In TGDH, the costs depend on the height of the resulting key tree and locations of joining or leaving members in the tree. We provide the worst case cost for TGDH.

Most of the cost in CODH comes from the master node. A general node consumes only one communication, modular exponentiation, signature, and verification in all of group rekeying process. We summarize the costs for a general member and the group master in Table 2. Although the exponentiation cost looks heavy in the master, its cost is insignificant. We conducted an experiment to measure computation delays for modular exponentiations. Table 3 shows the average delay of 10 experimental results for each. The first device has less CPU power than the second device. When modular prime p is 1024 bits long and $n \leq 50$, the computation delay is less than 1 s. The average delay of one exponentiation is less than 8 ms in the second device. Moreover, reducing communication cost is important for mobile devices because data communication consumes more energy than any other process. Therefore, our group key protocols can be efficiently applied in dynamic mobile networks.

5. Security

Let p be a large prime number of the form $2q+1$ for a prime q in $\mathbb{Z}_p$. Let G be a cyclic group of prime order q and let g be a generator of G; that is, $G = \langle g \rangle$. The decisional Diffie-Hellman problem (DDH) is as follows: given (g, g^x, g^y, g^z), where $x, y, z \in \mathbb{Z}_q$, decide whether $z = xy$ or a randomly chosen number. In particular, the security of our protocol is based on the divisible decisional Diffie-Hellman problem

Table 3: Computation delays on mobile devices (ms).

	p = 1024-bit			p = 2048-bit		
	$n = 1$	$n = 25$	$n = 50$	$n = 1$	$n = 25$	$n = 50$
Exponentiation (1 GHz CPU, 512 MB RAM)	37	452.6	907.8	168.9	3385.4	6762.5
Exponentiation (2.26 GHz CPU, 2 GB RAM)	12.9	192.8	385.3	75.4	1458.8	2917.4

(DDDH), which is stronger assumption than the divisible computational Diffie-Hellman problem (DCDH).

Definition 1. The DCDH problem is as follows: given (g, g^x, g^y), where $x, y \in \mathbb{Z}_q$, compute $g^{y/x}$.

Definition 2. The DDDH problem is as follows: given (g, g^x, g^y, g^z), where $x, y, z \in \mathbb{Z}_q$, decide whether $z = y/x$ or a randomly chosen number.

The DDDH problem is weaker than DCDH, since if an adversary could solve the DCDH problem, he could solve the DDDH problem by computing g^x to decide $g^z = g^{y/x}$; thus the DDDH assumption is stronger than the DCDH assumption. Similarly, the DDH problem is weaker than the computational Diffie-Hellman problem (CDH), which is weaker than discrete logarithm problem (DL) [22]. We want to prove the security of our protocol under the DDH and DDDH assumptions.

Theorem 3. *The DDDH problem is equivalent to the DDH problem.*

Proof. Given the DDDH input (g, g^x, g^y, g^z), where $z = y/x$, one submits (g, g^x, g^z, g^y) to DDH to decide whether $y = xz$ or a randomly chosen number. Similarly, given the DDH input (g, g^x, g^y, g^z), where $z = xy$, one submits (g, g^x, g^z, g^y) to DDDH to decide if $y = z/x$ or a randomly chosen number. □

Therefore, we know that if there is no polynomial time algorithm to solve the DDH problem, it is hard to solve the DDDH problem.

Theorem 4. *If the DDH problem is hard, it is hard to find a polynomial time algorithm to recover the group key from the proposed protocol; in other words, it provides group key secrecy against passive adversaries under the DDH assumption.*

Proof. Let $view(n, k)$ be public information for a group of n members to establish a group key g^k; thus it is a view of passive attackers,

$$view\,(n,k) := \left(g^{x_1}, g^{x_1 k}, g^{x_2}, g^{x_2 k}, \ldots, g^{x_n}, g^{x_n k}\right). \quad (10)$$

Suppose we had an algorithm **F** that with significant probability succeeds to distinguish between $(view(n, k), g^y)$, where y is a random number $y \in \mathbb{Z}_q$, and $(view(n, k), g^k)$ where g^k is the group key; that is, $\mathbf{F}(view(n, k), g^y) = \mathbf{F}(g^{x_1}, g^{x_1 k}, g^{x_2}, g^{x_2 k}, \ldots, g^{x_n}, g^{x_n k}, g^y) = 1$, where $y = k$, otherwise, returns 0. Then we can query to **F** with input $view(n-1, k) = (g^{x_1}, g^{x_1 k}, g^{x_2}, g^{x_2 k}, \ldots, g^{x_{n-1}}, g^{x_{n-1} k})$ for $n-1$ members' information and additional input $((g^{x_i})^r, (g^{x_i k})^r)$ for a random number $r \in {}_R\mathbb{Z}_q$, where $0 < i < n$, that is, $\mathbf{F}(view(n-1, k), g^{x_i r}, g^{x_i k r}, g^y)$. It follows that $(view(1, k), g^{x r_1}, g^{x k r_1}, g^{x r_2}, g^{x k r_2}, \ldots, g^{x r_{n-1}}, g^{x k r_{n-1}}, g^y) = \mathbf{F}(g^x, g^{xk}, g^{x r_1}, g^{x k r_1}, g^{x r_2}, g^{x k r_2}, \ldots, g^{x r_{n-1}}, g^{x k r_{n-1}}, g^y)$, where $r_i \in {}_R\mathbb{Z}_q$ for $0 < i < n$. Then F can solve the DDDH problem since it can decide whether $y = xk/x$ or a random number, given $(view(1, k), g^y)) = (g^x, g^{xk}, g^y)$. It means that F can also solve the DDH problem by Theorem 3. □

Theorem 5. *The proposed scheme provides backward secrecy, forward secrecy, and key independence provided the DDH problem is intractable.*

Proof. Whenever membership is changed or the group key is updated, the group controller alters its own secret k to k', where k' is an independently random number to $k \in \mathbb{Z}_q$; it implies that it is impossible to find an algorithm F such that $F(g^k) \rightarrow g^{k'}$ without knowledge of k and k'. We assume that the secret values are uniformly distributed by a pseudorandom generator. Therefore, when the group key has been changed, an adversary must use new public information, $view(n, k') = (g^{x_1}, g^{x_1 k'}, g^{x_2}, g^{x_2 k'}, \ldots, g^{x_n}, g^{x_n k'})$, to recover the group key updated into $g^{k'}$ and it depends on a solution to solve the DDH problem by Theorem 4. It follows that past members, future members, or adversaries who know a subset of previous group keys cannot learn the current group key, since the broadcast message from the master does not contain their locker X_i in $view()$. □

Theorem 6. *The proposed scheme provides implicit key authentication under the security of certified public key.*

Proof. A locker which the master obtains from group members is what a group member signs with its public key certified by a CA. Concretely, a locker X_i is hashed by a one-way function such as SHA-2, and hash (X_i) is signed with M_i's private key using a digital signature algorithm such as RSA, DSA, and ECDSA. Then, the locker is verified with the public key bound to M_i and certified by CA. If there is a locker of nonmember in the locker list of a group, it must be along with a forged signature. It means that the problem occurs in a hash collision attack or a rogue CA certificate [23]. Once all verified lockers are transferred to the master, any other nodes which are not a group member cannot recover the group key under the DDH assumption (Theorems 4 and 5). □

6. Conclusion

In this paper, we propose a secure group key management protocol based on DH key agreement. The proposed key management requires only one data communication and one modular exponentiation at each member for any membership event. It shows prominent efficiency in renewing the group keys against dynamic group membership change, member join/leave and group merging/partition. We proved group key secrecy, backward/forward secrecy, key independence, and key authentication. No outsiders can learn the group key under the DDH assumption. We conclude that CODH can be adapted efficiently for multicast security in mobile networks.

Notations

n:	Number of protocol participants
M_i:	ith group member, $i \in [1, n]$
M_s:	Master node (controller), $s \in [1, n]$
p:	Prime of the form $2q + 1$ for a prime q
g:	Generator in $\mathbb{Z}_p^*$
x_i:	Lock-secret; random number picked by M_i such that $1 < x_i < p - 1$ and $\gcd(x_i, p - 1) = 1$
y_i:	Unlock-secret for M_i such that $x_i * y_i \equiv 1 \bmod (p - 1)$
k:	Master-secret randomly selected in $\mathbb{Z}_q^*$, by M_s
X_i:	Locker; $g^{x_i} \bmod p$
C_n:	Current group of n members; $\#(C) = n$
XL_C:	Locker list of group C; $XL_C = \{X_1, X_2, \ldots, X_n\} \setminus X_s$
XL_C^k:	Key-locks for group C; $XL_C^k = \{X_1^k, X_2^k, \ldots, X_n^k\} \setminus X_s^k$
$M_i \rightarrow M_j$: m:	Unicast message (m) from M_i to M_j
$M_i \Rightarrow C_n$: m:	Broadcast message (m) from M_i to n members of C.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors appreciate anonymous reviewers for their helpful comments. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0011289).

References

[1] R. Canetti, T. Malkin, and K. Nissim, "Efficient communication-storage tradeoffs for multicast encryption," in *Proceedings of Advances in Cryptology (Eurocrypt '99)*, vol. 1592 of *Lecture Notes in Computer Science*, pp. 459–474, Prague, Czech Republic, May 1999.

[2] S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: a scalable group re-keying approach for secure multicast," in *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 215–228, Berkeley, Calif, USA, May 2000.

[3] M. K. Reiter, "A secure group membership protocol," *IEEE Transactions on Software Engineering*, vol. 22, no. 1, pp. 31–42, 1996.

[4] D. Wallner, E. Harder, and R. Agee, "Key management for multicast: issues and architectures," RFC 2627 Informational, 1999.

[5] C. K. Wong, M. Gouda, and S. S. Lam, "Secure group communications using key graphs," *IEEE/ACM Transactions on Networking*, vol. 8, no. 1, pp. 16–30, 2000.

[6] S. Mittra, "Iolus: a framework for scalable secure multicasting," in *Proceedings of the ACM (SIGCOMM '97)*, pp. 277–288, Cannes, France, September 1997.

[7] 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.

[8] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 62–72, Washington, DC, USA, October 2003.

[9] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing pairwise keys for secure communication in ad hoc networks: a probabilistic approach," in *Proceedings of the 11th IEEE International Conference on Network Protocols*, pp. 326–335, Atlanta, Ga, USA, 2003.

[10] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[11] M. Burmester and Y. Desmedt, "A secure and scalable group key exchange system," *Information Processing Letters*, vol. 94, no. 3, pp. 137–143, 2005.

[12] M. Steiner, G. Tsudik, and M. Waidner, "Diffie-Hellman key distribution extended to group communication," in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, pp. 31–37, New Delhi, India, March 1996.

[13] M. Steiner, G. Tsudik, and M. Waidner, "CLIQUES: a new approach to group key agreement," in *Proceedings of the 18th International Conference on Distributed Computing Systems*, pp. 380–387, May 1998.

[14] Y. Amir, Y. Kim, C. Nita-Rotaru, and G. Tsudik, "On the performance of group key agreement protocols," *ACM Transactions on Information and System Security*, vol. 7, no. 3, pp. 457–488, 2004.

[15] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60–96, 2004.

[16] Y. Kim, A. Perrig, and G. Tsudik, "Group key agreement efficient in communication," *IEEE Transactions on Computers*, vol. 53, no. 7, pp. 905–921, 2004.

[17] B. Wu, J. Wu, and D. Yuhong, "An efficient group key management scheme for mobile ad hoc networks," *International Journal of Security and Networks*, vol. 4, no. 1-2, pp. 125–134, 2009.

[18] P. P. C. Lee, J. C. S. Lui, and D. K. Y. Yau, "Distributed collaborative key agreement and authentication protocols for dynamic peer groups," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 263–276, 2006.

[19] J. Liu, D. Sacchetti, F. Sailhan, and V. Issarny, "Group management for mobile Ad Hoc networks: design, implementation and experiment," in *Proceedings of the 6th International Conference on Mobile Data Management (MDM '05)*, pp. 192–199, Ayia Napa, Cyprus, May 2005.

[20] B. Singh and D. K. Lobiyal, "A novel energy-aware cluster head selection based on particle swarm optimization for wireless sensor networks," *Human-Centric Computing and Information Sciences*, vol. 2, no. 13, pp. 1–18, 2012.

[21] C.-W. Chen, Y.-R. Tsai, and S.-J. Wang, "Cost-saving key agreement via secret sharing in two-party communication systems," *Journal of Convergence*, vol. 3, no. 4, pp. 29–36, 2012.

[22] S. Mohanty and B. Majhi, "A strong designated verifiable dl based signcryption scheme," *Journal of Information Processing Systems*, vol. 8, no. 4, pp. 567–574, 2012.

[23] M. Stevens, A. K. Lenstra, and B. de Weger, "Chosen-prefix collisions for MD5 and applications," *International Journal of Applied Cryptography*, vol. 2, no. 4, pp. 322–359, 2012.